IMT SCHOOL

Embedded system interfacing

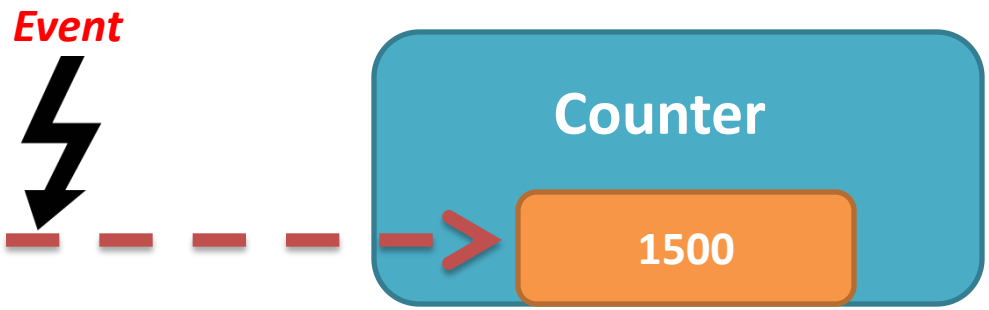
Lecture Nine

Timer & counter

Introduction

What is counter ?

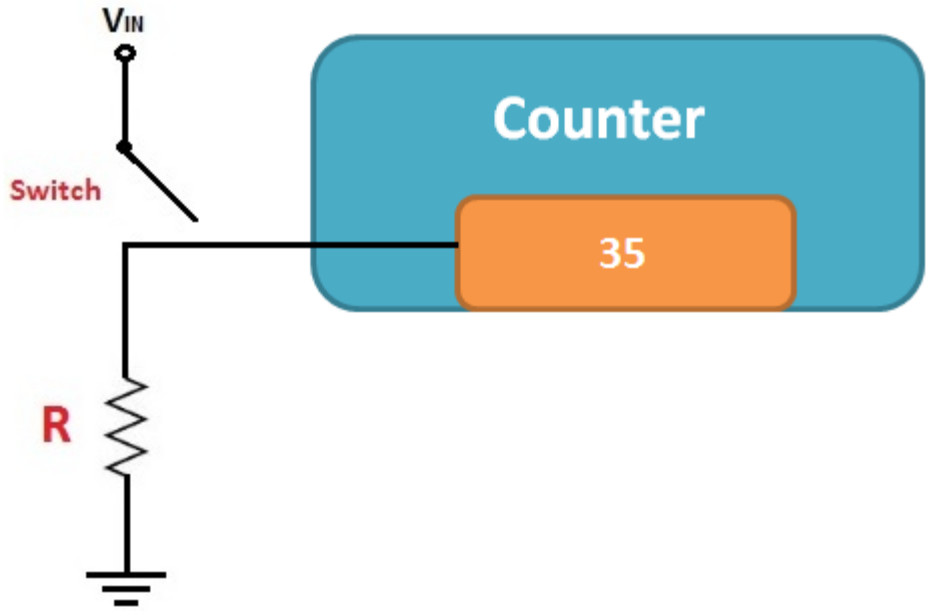
Simply counter is a peripheral that counts ***events*** in a specific register. These events are ***electrical signals*** like ***rising edge*** or ***falling edge***. If these events are periodic (Comes every defined period) then this counter is counting time, hence it is called **Timer.**



Introduction

Example

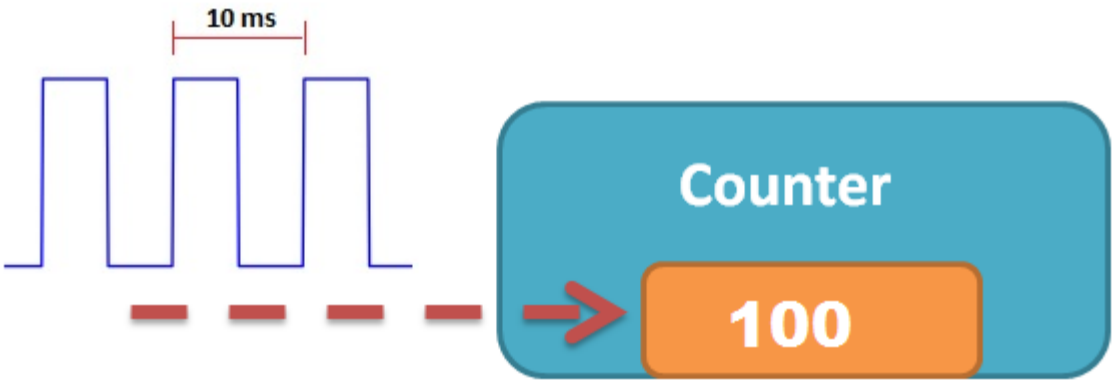
Think of a peripheral that counts a switch presses, The Counter is connected to one terminal of the switch which gives a ***rising edge*** every press. The counter increments its register value by 1 count every press. Then if we read the counter register and found 35, it means that there are 35 switch presses have been done.



Introduction

Example

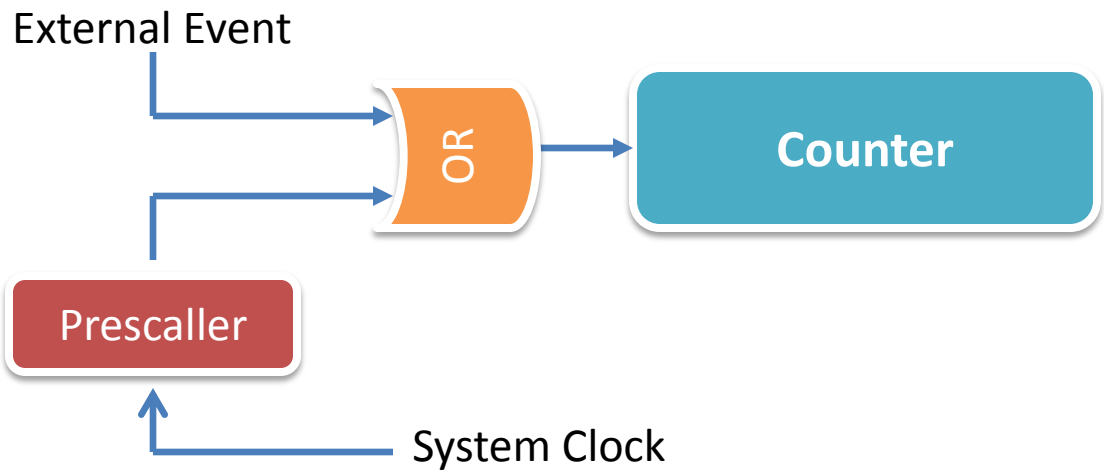
On the other hand, if the counter is connected to ***a periodic signal*** with a periodic time equals to 10 ms. The counter is configured to count every ***rising edge*** as the previous example. If we read the counter register and found 100, it means that 100 rising edge are happened. Because the signal is periodic and the it gives rising edge every 10 ms, then ***we can conclude that 1000 ms have been passed*** from the time we started the counter. This is the timer mode.



Introduction

From previous examples, we can say that a ***timer is a normal counter*** but counting a periodic signal. So that we can calculate the time from the value in the timer register.

In summary, the timer and the counter are the same peripheral, depending on the input we could classify if it is working as timer or as counter.



Timer Terminology

Resolution

Number of bits represents the timer register.

Timer Tick Time

The time between to consecutive events, i.e. time needed to increment the timer register by 1

Timer Count

Number of counts needed by the timer to count from 0 till it reaches 0 again. It shall be equals to $2^{Resolution}$

Timer Overflow

The time needed by the timer to count from 0 until it reaches 0 again. It shall be equals to **Timer Count x Timer Tick Time**

Prescaller

Prescaller is a division factor for the system clock (Processor Clock) before it is applied to the timer.

Timer calculations

- **Timer Clock =** $\dfrac{\text{System Clock}}{\text{Prescaller}}$
- **Timer Tick Time =** $\dfrac{1}{\text{Timer Clock}} = \dfrac{\text{Prescaller}}{\text{System Clock}}$

Example

Assume a microcontroller working on a frequency of 4 MHz, calculate the timer tick time assuming the timer is using a prescaller of 4.

Solution

Timer Tick Time = $\dfrac{4}{4 \times 10^6}$ = 1 micro second.

i.e. this timer would be incremented every 1 micro second.

Timer calculations

- **Timer Overflow Timer = Timer Count x Timer Tick Time**

$$= 2^{\text{Resolution}} \times \frac{\text{Prescaller}}{\text{System Clock}}$$

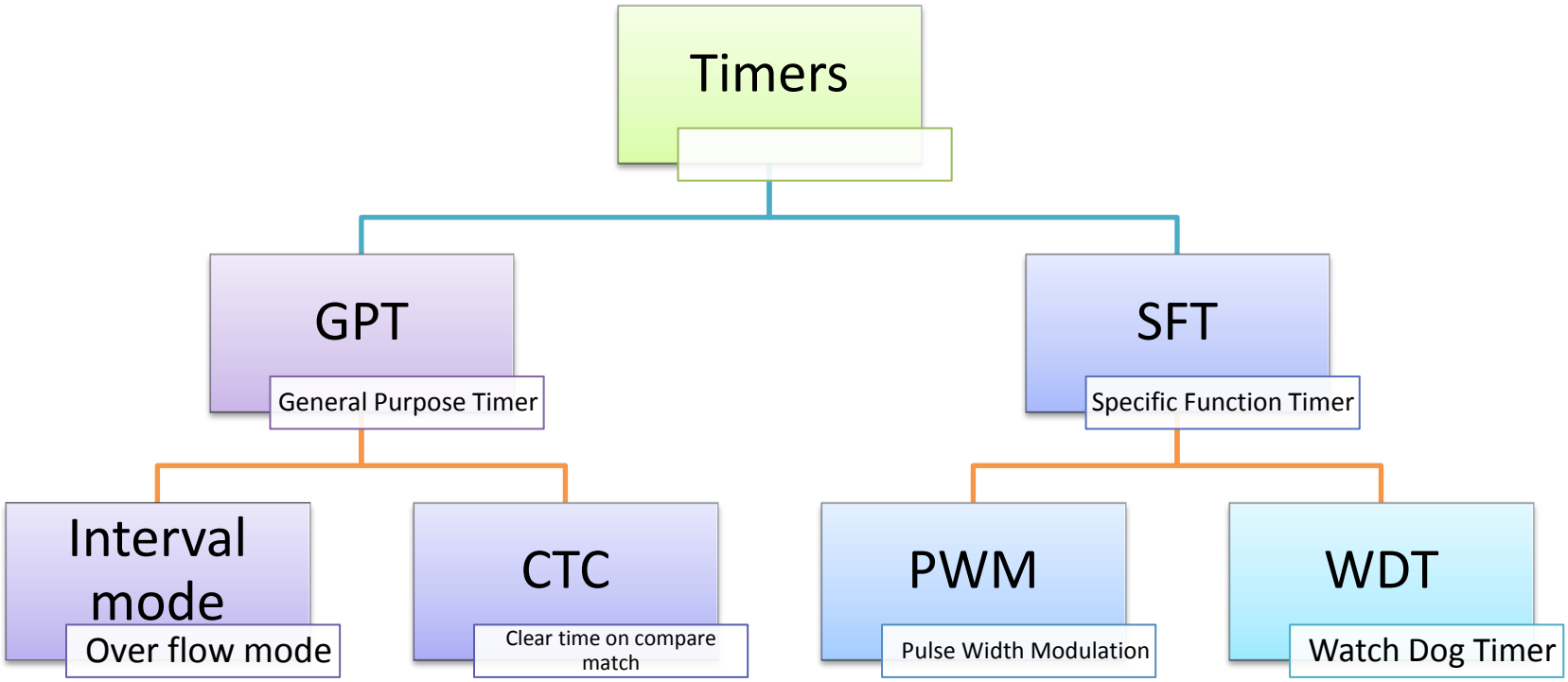
Example

From the previous example, assuming the timer resolution is 8 bit, calculate the over flow time.

Solution

Over flow time = 1 micro second x 2^8 = 256 micro sconds.

Types of timers



Interval Timer Normal Mode

Interval timer is a normal functioning timer that counts until it overflows. On overflow, the timer sets an over flow flag and fires an interrupt if enabled.

How to Calculate a certain time based on an interval timer … ?

Step 1 Calculate the overflow time of the used timer.

Step 2 Compare the desired time with the overflow time, and follow one of the following possibilities.

Possibility 1

Desired Time equals to Overflow time

This is the happy scenario possibility, which normal doesn't happen. If you are lucky and the desired time is equals to overflow time, then just enable the overflow interrupt and take the desired action inside the overflow ISR.

```
ISR (OverFlow)
{
    /* Take Desired Action */
}
```

Possibility 2

Desired Time less than Overflow time

Step 1 Calculate the number of counts needed

$$\text{Number of counts needed} = \frac{\text{Desired Time}}{\text{Overflow Time}} \times \text{Over Flow Count}$$

Step 2 Preload the timer register with the following Value:

Preload Value = Over Flow Count – Number of counts needed

Step 3 Enable the timer overflow interrupt and take the desired action

See the following example to understand more …

Example

Assuming 8 bit timer using a prescaller of 4. The system frequency is 4 MHz and an action is needed to be taken after 64 micro second. Do the needed calculations and write the code.

Solution

Over flow time = 256 micro seconds (calculated in previous slides)

The desired time is 64 micro seconds which is less than the overflow time. Then we calculate the needed number of counts:

Number of needed Counts = (64 / 256) x 256 = 64 count.

Preload value = 256 – 64 = 192.

If the timer register is initialized with the preload value (192), then it needs only 64 count to overflow, i.e. the overflow interrupt would fire after only 64 count. Which is the desired time.

Example

Code Solution

At Initialization

```
Timer_Register = 192;
```

Then

```
ISR (OverFlow)
{
    /* Initialize the timer register again to */
    /* keep the timer counting the same value */
    Timer_Register = 192;

    /* Take the desired action */
}
```

Possibility 3 (case 1)

Desired Time more than Overflow time

Step 1 Calculate the number of overflows needed

$$\text{Number of over flows} = \frac{\text{Desired Time}}{\text{Overflow Time}}$$

Assume that the number of the overflows needed is a *decimal value*, then define a variable and increments it inside the overflow interrupt till we reach the desired number of overflows, then take the desired action.

See the following example to understand more …

Example

Assuming 8 bit timer using a prescaller of 4. The system frequency is 4 MHz and an action is needed to be taken after 2.56 milli seconds. Do the needed calculations and write the code.

Solution

Over flow time = 256 micro seconds (calculated in previous slides)

The desired time is 2.56 milli seconds which is more than the overflow time. Then we calculate the needed number of overflows:

Number of overflows = (2560 / 256) = 10 overflows.

Example

Code Solution

At Initialization

```
u8 Overflows_Count = 0;
```

Then

```
ISR (OverFlow)
{
    Overflows_Count++;

    if (Overflows_Count == 10)
    {
        /* Re-initialize the variable to 0 again */
        /* to count the 2560 microseconds again  */
        Overflows_Count = 0;

        /* Take the desired action */
    }
}
```

Possibility 3 (case 2)

Desired Time more than Overflow time

Step 1 Calculate the number of overflows needed

$$\text{Number of over flows} = \frac{\text{Desired Time}}{\text{Overflow Time}}$$

Assume that the number of the overflows has a floating value, then calculate the preload value for the floating part first, then calculate the number of the overflows for the decimal value.

See the following example to understand more …

Example

Assuming 8 bit timer using a prescaller of 4. The system frequency is 4 MHz and an action is needed to be taken after 2.62 milli seconds. Do the needed calculations and write the code.

Solution

Over flow time = 256 micro seconds (calculated in previous slides)

The desired time is 2620 micro seconds which is more than the overflow time. Then we calculate the needed number of overflows:

Number of overflows = (2620/ 256) = 10.25 overflows.

First Calculate the preload value for the floating value,
Preload Value = 256 – (0.25 x 256) = 192 counts.

Second Calculate the number of overflows needed for the decimal value,
Number of overflows = 10 / 1 = 10 overflows.

Example

Code Solution

At Initialization

```
u8 Overflows_Count = 0   ;
Timer_Register     = 192 ;
```

Then

```
ISR (OverFlow)
{
    Overflows_Count++;

    if (Overflows_Count == 11)
    {
        /* Re-initialize the variable to 0 again */
        /* And the timer register to 64          */
        /* to count the 10.5 microseconds again  */
        Overflows_Count = 0;
        Timer_Register     = 192


        /* Take the desired action */
    }
}
```

Can you understand why it is 11 not 10 … ?

AVR timers

In ATMEGA32, we have three different kinds of timers:

TIMER0

- 8-bit timer
- CTC mode
- Phase Correct PWM
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources

TIMER1

- 16-bit timer
- Two Independent Output Compare Units
- One Input Capture Unit
- Phase Correct (PWM)
- Variable PWM Period
- Four Independent Interrupt Sources

TIMER2

- 8-bit timer
- CTC mode
- Phase Correct PWM
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources
- Allows clocking from External 32 kHz

Timer 0 Registers

Refer to datasheet page (78:80) for full description of the registers

Timer/Counter Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
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	

Timer/Counter Register – TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

Output Compare Register – OCR0

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
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 Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
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	

lab 1

Description :

Write a code to toggle a LED every 1 sec using timer 0 “Overflow mode”

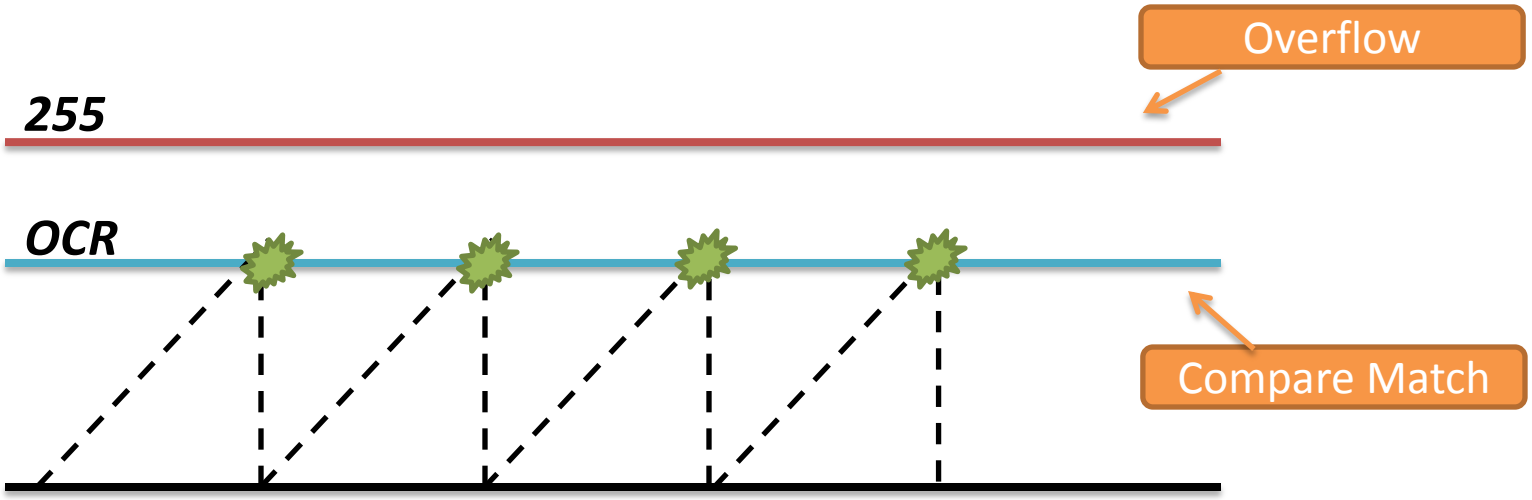
lab 2

Description :

Develop the timer 0 driver in the MCAL

CTC
(clear timer on Compare)

- Compare Match Interrupt is fired by a timer whenever the value of the timer becomes equal to a certain predefined value.
- This predefined value is stored in a register known as the Output Compare Register.



lab 3

Description :

Make a system that generates different tones on a buzzer by changing potentiometer value. The potentiometer is connected to ADC Pin, read the potentiometer and change the OCR value accordingly. In the compare match interrupt toggle a pin that connected to the buzzer. Changing the OCR means changing the frequency of buzzer signal which means changing the buzzer tone.

The End …

IMT SCHOOL

www.imtschool.com

ww.facebook.com/imaketechnologyschool/