STANDARD SINGLE-PURPOSE PERIPHERALS

❖ TIMERS :-

A timer is an peripheral device that can measure time intervals. Timer can be used to

i) Generat events at specific time or to determine the duration between two external events.

eg: Keeping a traffic light green for a specified duration or communicating bits serially between devices at a specific rate.

ii) To determine duration of two external events.

eg: Computing a cars speed by measuring the time the car takes to pass over two separated sensors in a road.

A timer measures time by counting pulses that occur on an input clock signal having a known period.

❖ COUNTERS :-

A counter counts pulses onsome other input signal.

eg: A counter may be used to count the number of cars that pass over a road sensor ot thr number of people that pass through a turnstile.

❖ **TIMERS & COUNTERS** are combined to measure rates.

eg: Counting the number of times the car wheel rotates in one second, in order to determine a cars speed.

❖ Timer Structure :-

The timer structures are

1) Basic Timer
2) A Timer/Counter
3) A Timer with a terminal count
4) A 16/32-bit Timer
5) A Timer with a prescaler

1) Basic Timer :-

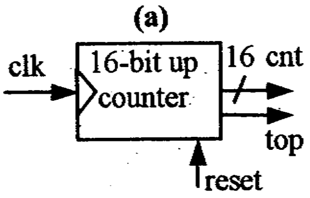


Fig ①: A Basic timer

* The timer has an internal 16 bit upcounter which -increments its value on each clockpulse

* The output value cnt represent the number of pulses sinces 8 the counter was last reset to zero.

* It has additional output top that indicates when the top value of its range has been reached. It is also known as overflow occuring in which case the timer rolls over to zero

* We define a timer range as the maximum time

interval the timer can measure

* Also, a time resolution is the minimum interval it can measure

1) If the given datas are

$f = 100MH_z$. $Cnt = 20,000$.

Find resalution and range.

Soln :-

* Resolution $= \frac{1}{f} = \frac{1}{100MH_z} = 10nsec$

* Range $=$ Cnt × Resolution $= 20,000 \times 10nsec = 200\mu sec$

2) If the given data are $f = 100MH_z$, counter $= 16$-bit find resalution and range.

Soln: 16-bit counter will count from 0 to 65,535

$\therefore$ cnt $= 65,535$

* Resolution $= \frac{1}{f} = \frac{1}{100MH_z} = 10 nsec$

* Range $=$ cnt × Resolution $= 65,535 \times 10nsec = 655.35\mu sec$

2) A Timer/Counte :-

* The mode register holds a bit by which the user can set that uses a 2 X 1 Multiplexer to select the clock input to the internal 16-bit Counter

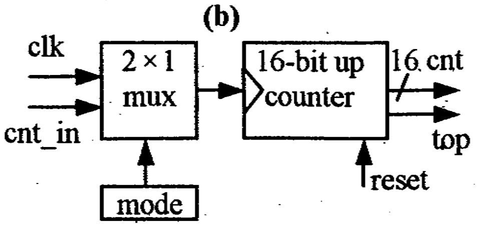


Fig ①: A timer/counter

* The clock input can be the external clock signal, in which case the device acts like a timer.

* The clock input can be the external clock signal, in which case the the device acts like a timer

* The clock input can be the external cnt-in signal ,in which case the device acts like a counter, counting the occurence of pulses on cnt-in.

* Cnt-in would typically be connected to an external Sensor, so pulses would occur at indeterminate intervals.

3) A Timer with a terminal count :-

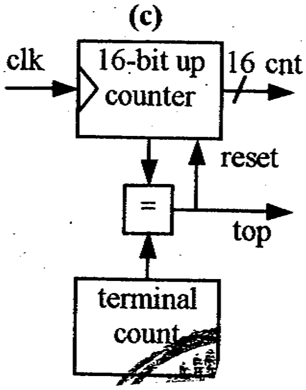


Fig ①: A timer with a terminal count

* A Terminal count register holds a value, which the user sets, indicating the number of clock cycles, in the desired interval & is determined by using simple formulae:

$$\boxed{\text{Number of cycles} = \frac{\text{Desired time interval}}{\text{Clock period.}}}$$

For ex, to obtain a duration of 3 microseconds from a clock cycle of 10nsec ($f = 100MHz$), then

$$\text{No of clock cycles} = \frac{3\times10^{-6}}{10\times10^{-9}} = 300 \text{ cycles.}$$

* The timer structure includes the comparator that asserts the top output (ie top=1) when the terminal count has been reached.

The top output is used to:

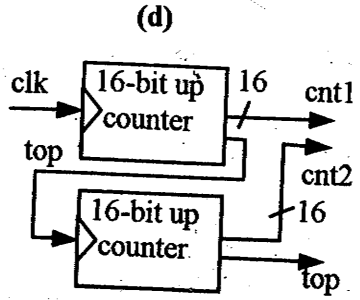
i) Reset the counter to zero and.

ii) To inform the user that the desired time interval has passed.

* To top signal is often connected to an interrupt (Pin). The corresponding <u>ISR</u> would include the actions that must be taken at the specified time interval.

* To improve efficiency, a down counter is used rather than an up counter.

4) A 16/32-bit Timer :-



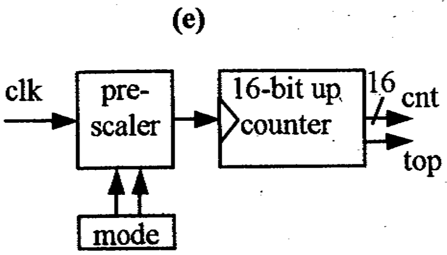
Fig①: A 16/32 bit timer

* Fig-① shows the structure of a timer that can be configured as a 16 or 32 bit timer.

The Timer simply uses the top output of its first 16 bit upcounter as the clock input of its second 16 bit counter.

These are known as cascaded counters.

5) A Timer with a prescaler :-



Fig①: A timer with a prescalar

* A prescaler is essentially a configurable clock divider circuit. Depending on the mode bits being input to the prescaler.

* The prescalar o/p signal might be

i) Same as the input signal frequency

ii) $1/2$ of the frequency.

iii) $1/4$ of the frequency

iv) $1/8$ of the frequency.

Mode bits	o/p frequency
0 0	Same as I/p Signal
0 1	$1/2$ of I/p signal
1 0	$1/4$ of I/p signal
1 1	$1/8$ of I/p signal.

Eg :- consider a timer with a resolution of 10 nsec and a range of 65,535. If the prescalar is configured to divide the clock frequency by 8, then calculated timer resolution.

Sol :-

* Resolution = 10nsec × 8 = 80nsec

* Range = cnt × Resolution = 65,535 × 80 nsec

$$\boxed{\text{Range} = 5.24\ \mu\text{sec}}$$

REACTION Timer :-

Reaction Timer is an application that measures the time a person takes to respond to a visual or audio stimulus.

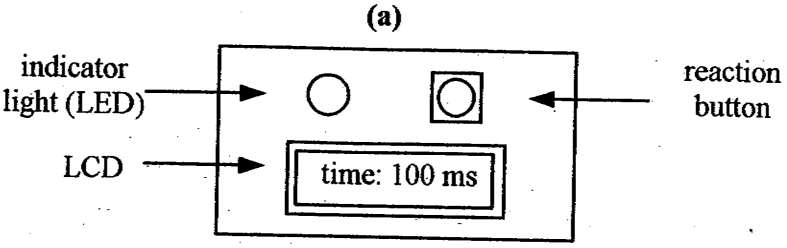


Fig ①: Reaction timer : LED, LCD and button

* The application turns on a LED, then measures the time a person takes to push a button in response, and display this time on an LCD.

Reaction time is expected to be in the order of seconds and displays reaction time to be miliseco-nd precision.

* In this example, we will use a microcontroller with a built in 16-bit timer. The timer is incremented once every instruction cycle for this microcontroller is 6 clock cycles.

* The Timer doesnot have a prescaler or Terminal count register.

* It has a top signal to indicate overflow also allows us to load an initial value for its internal up counter.

* If the clock frequency is 12 MHz, then

$$\text{period} = \frac{1}{f} = \frac{1}{12\text{MHz}} = 83.33\,\text{nsec}$$

* Each instruction has 6 clock cycles

$$\text{Resolution} = 1 \text{ instruction cycle} \times \frac{1}{f}$$

$$\text{Resolution} = 6 \times 83.33\,\text{nsec}$$

$$\boxed{\text{Resolution} = 0.5\,\mu\text{sec}}$$

* The timer is of 16-bit, so maximum value is

$$2^{16} = 65,536$$

$$\therefore \text{Range} = \text{Resolution} \times \text{Maximum value} \quad [0 - 65536 = 65,536]$$

$$= 0.5\,\mu\text{sec} \times 65,536$$

$$\boxed{\text{Range} = 32.77\,\text{m sec}}$$

* We note that this timer's range is smaller than our desired range of several seconds, while its resolution is finer than our required 1msec.

* The initial timer value can be set that the overflow occurs every 1msec. Then we can monitor the top output signal of the timer to activate the code. The code keeps a count of overflows, indicating the number of miliseconds.

* The initial value to be loaded is determined

ie initial value = Maximum value of counter $-\left[\text{Required time} \times \dfrac{\text{Oscillator freq}}{\text{No. of cycles per inst}}\right]$

$$= 65,536 - \left[1\text{msec} \times \frac{12\text{ MHz}}{6 \text{ cycles}}\right]$$

$$= 65,536 - 2000$$

$\therefore \boxed{\text{Initial value} = 63{,}536}$

* The Pseudocode describing the reaction timer implementation is written below.

```
/* main. c*/
#define MS_INIT 63535
void main(void)
{
    int count_miliseconds = 0;
    configure timer mode
    set cnt to MS_INIT
    Wait a random amount of time
    turn ON indicator light
    Start timer
    While (user has not pushed reaction button)
    {   if (top)
        { Stop timer
          set cnt to MS_INIT
          Start timer
          reset top
          count_milliseconds ++;
        }
    }
    turn off indicator light
    printf("Time: %i ms", count, miliseconds);
```

FORMULAE

1) Resolution $= 1$ instruction cycle $\times \frac{1}{f}$

2) Range = Resolution × Maximum value (count)

NOTE :-

Resolution $= \frac{1}{f}$

3) Maximum division needed in a prescalar $= \dfrac{\text{Prescalar measurement}}{\text{Max value of the counter}}$

4) Terminal count $= \dfrac{\text{Desired Time Interval}}{\text{clock period } (T = \frac{1}{f})}$

1) A 16-bit timer operates at a clock frequency of 12MHz. Determine the resolution and range of this timer.

Given :- $f = 12\ MHz$

Timer = 16 - bit

Maximum value $= 2^{16} - 1 = 65,535$

sol :- * Resolution $= \frac{1}{f} = \frac{1}{12\ MHz}$

$$\boxed{\text{Resolution} = 83.33\ nsec}$$

* Range = Resolution × Maximum value

$= 83.33 nsec \times 65,535$

$$\boxed{\text{Range} = 5.5\ msec}$$

2) Determine the range and resolution of a 16-bit timer which operates at a clock frequency of 10Mhz and generate an overflow signal when it reaches FFFF. Calculate the terminal count value for measuring a 3 msec time interval. What is the minimum division needed in a prescaler for measuring 100 msec?.

June-09,6M

Given :-

$f = 10MHz$

16- bit timer

$\therefore$ Maximum value $= 2^{16} - 1 = 65,535$

Time interval $= 3\,msec$

prescalar measurement $= 100\,msec$

Sol:- * Resolution $= \frac{1}{f} = \frac{1}{10MHz}$

$$\boxed{\text{Resolution} = 0.1\,\mu sec}$$

* Range = Resolution × Max value

$= 0.1\,\mu sec \times 6$

$$\boxed{\text{Range} = 6.5535\,msec}$$

* Terminal Count $= \dfrac{\text{Desired time interval}}{\text{period } (T = \frac{1}{f})}$

$= \dfrac{3\,msec}{0.1\,\mu se}$

$$\boxed{\text{Terminal Count} = 30,000 \text{ cycles.}}$$

* Prescalar minimum division $= \dfrac{\text{Prescaler measurement}}{\text{Max value}}$

$= \dfrac{100\,msec}{65,535}$

$$\boxed{\text{Prescaler minimum division} = 1.52 \times 10^{-6}}$$

3) Given a 16-bit timer with 20 Mhz,

i) Determine its range and resolution

ii) Calculate the terminal count value needed to measure 1.5 msec interval

iii) If a prescalar is added, what is the minimum division needed to measure an interval of 50 msec. Determine its range and resolution, if the division value is a power of 2.

Jan-08,6M

Given :- $f = 20 MH_3$, $T = \frac{1}{f} = 50 nsec$

Timer is 16 bit ∴ Maximum Value $= 2^{16} - 1 = 65,536$

soln : (a) Resolution $= \frac{1}{f} = \frac{1}{20 MH_3} = 50 nsec$

Range = Resolution × Maximum value $= 50 nsec \times 65,535$

$$\boxed{\text{Range} = 3.27675 \text{ msec}}$$

(b) Desired time interval $= 1.5 msec$

$$\text{Terminal count} = \frac{\text{Desired time interval}}{\text{period } (T = \frac{1}{f})}$$

$$= \frac{1.5 msec}{50 nsec}$$

$$\boxed{\text{Terminal count} = 30,000}$$

(c) Prescaler measurement interval $= 50 msec$

$$\text{prescaler minimum division} = \frac{\text{Prescaler measurement}}{\text{Maximum value}}$$

$= \frac{50 msec}{65,535}$

$\boxed{\text{prescal minimum division} = 0.762\ \mu sec}$

* Resolution $= \frac{1}{f} = 50 nsec$

Range $= 2^{\textcircled{2}\times 16} \times$ Resolution

$\boxed{\text{Range} = 214.748\ sec}$

Note :-

* The maximum value of timer $= 2^{16}$
* Determine its range & resolution if the division valu is a power of 2.

ie Maximum value of time $= 2^{\textcircled{2}\times16} = 2^{32}$

WATCHDOG TIMER :-

❖ **What is watchdog timer? Explain ATM timeout using a watchdog timer.**

Jan-11,10M

❖ **With a neat diagram, explain functioning of a watch dog timer. Discuss the usage of watch dog timers. Write a Pseudo code for an ATM machine to demonstrate the usage of watch dog timer.**

Jan-10,10M

A Special type of timer is a watchdog timer, which will reset the system after a predefined timeout.

Watchdog timer reset timer every X time unit, else timer generates a signal indicating that the system failed.

Common use of watchdog timer is to enable an embedded system to restart itself in case of a failure.

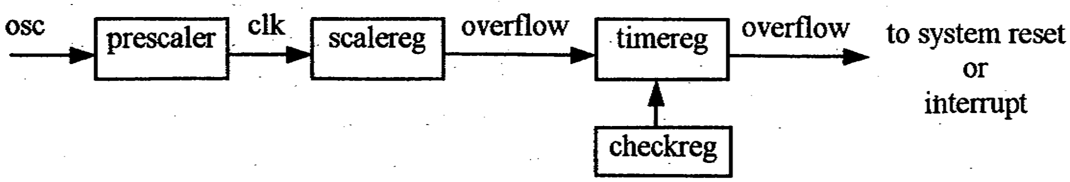
Another common use is to support timeouts in a program while keeping the program structure simple.

Example of ATM timeout using a watchdog timer :-

* In this example, a watchdog timer is used to implement a time out for an automatic teller machine (ATM).

* A normal ATM session involves a user inserting a bank card, typing in a Personal identification number (PIN), and then answering questions about whether to deposit or withdraw money, which account will be involved, how much money will be involved, whether another transaction is desired, and so on.

* We want to design the ATM such that it will terminate the session if at any time the user doesnot press any button for 2 minutes. In this case the ATM will eject the bank card and terminate the session.



fig①: ATM timeout using a watchdog time Structure.

* An oscillator signal OSC is connected to prescaler that divides the oscillator frequency by 12 (OSC/12) to generate a signal clk

* The signal clock is connected to an 11-bit up counter Scalereg. When scalereg overflows, it rolls over to '0', and its overflow output causes the 16 bit up-counter time reg to increment.

* If timereg overflows, it triggers the System reset or an interrupt. To reset the watchdog Timer, checkreg must be enabled. Then a value can be loaded into timereg.

* When a value is loaded into time reg, the checkreg register is automatically reset. If the checkreg register is not enabled, a value cannot be loaded into timereg. This is to prevent erroneous software from unintentionally resetting the watchdog timer.

* Let us determine what value to load in timereg to achieve a timeout of 2 minutes

The osc signal frequency is 12MHz. The timereg is incremented at every 't' seconds, where

$$t = \text{Prescalar} \times \text{Scalereg} \times \frac{1}{\text{osc freq}}$$

$$= 12 \times 2^{11} \times \frac{1}{12 \times 10^{6}}$$

$$t = 0.002 \text{ sec}$$

$\therefore$ $\boxed{\text{Watchdog timer resolution = 2msec}}$

* Since Timereg is a 16-bit register, its range is 0 to 65,535.

$$\therefore \text{Time range} = \text{Max count} \times \text{Resolution}$$
$$= 65,535 \times 2\text{msec}$$

$$\boxed{\text{Timer range} = 131.070 \text{ msec}}$$

(Approximately 2.18 minutes)

* To obtain Timereg value:

$$\text{Timereg value} = 131,070 - X$$
$$X = 131,070 - \text{Timereg value}$$

(WKT timereg value = 2 minutes = 120000 msec)

$$X = 131070 - 120000$$
$$\boxed{X = 11070}$$

* The pseudocode for the main routine and the watchdog reset routine to implement the timeout functionality of the ATM is shown below:

Main pseudo-code:

```
/* main.c */
main ( )
{ wait until card inserted
  call watchdog_reset_routine
  While ( transaction in progress)
  {  if ( button pressed)
     { perform corresponding action
       call watchdog_reset_routine
     }
  /* if watchdog-reset-routine not called every <2 minutes,
  interrupt-service-routine is called */
  }
}
```

Watchdog timer reset routine :-

```
Watchdog-reset_routine ( )
{ /* Checkreg is set so we can load value into Timereg.
  Zero is loaded into scalereg and 11070 is loaded into
  Timoreg */

  checkreg  = 1
  scalereg  = 1
  timreg    = 11070
}
vaid interrupt_service_routine ( )
{   eject card
    reset screen
}
```