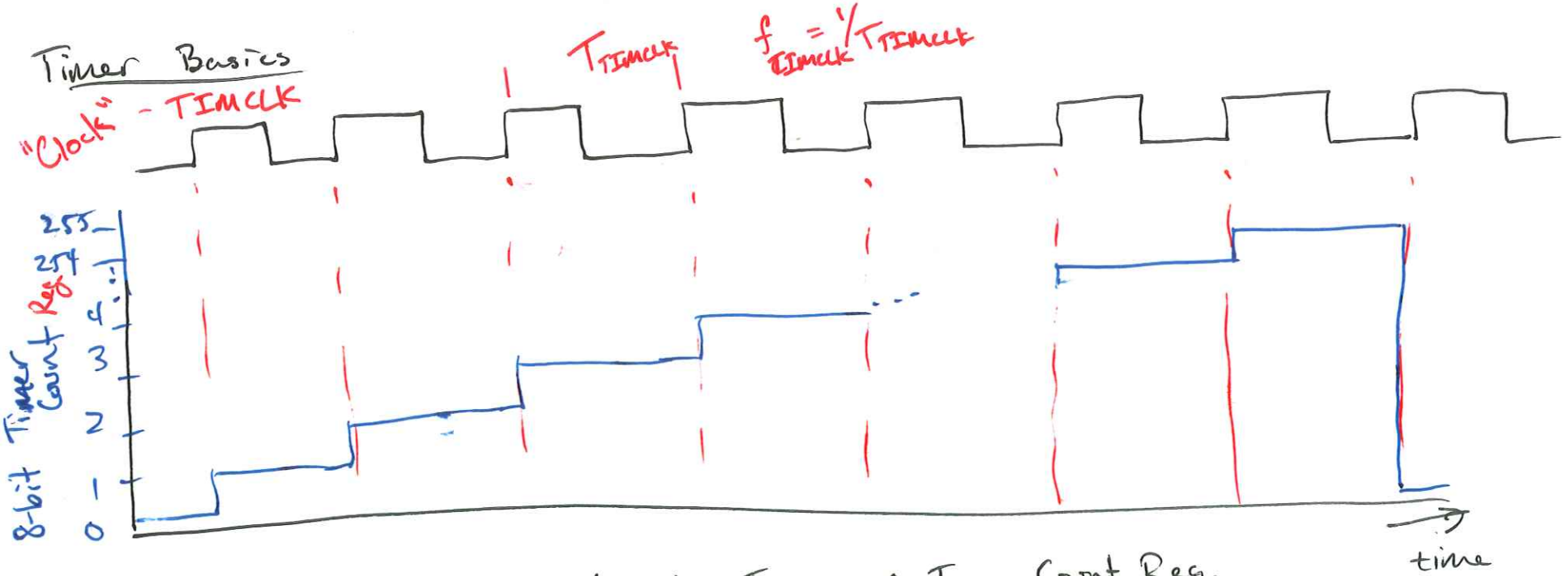
Timer Basics



time elapsed = T_{TIMCLK} • Timer Count Reg.

if $f_{TIMCLK} = 1\,MHz$, then $T_{TIMCLK} = 1\,\mu s$

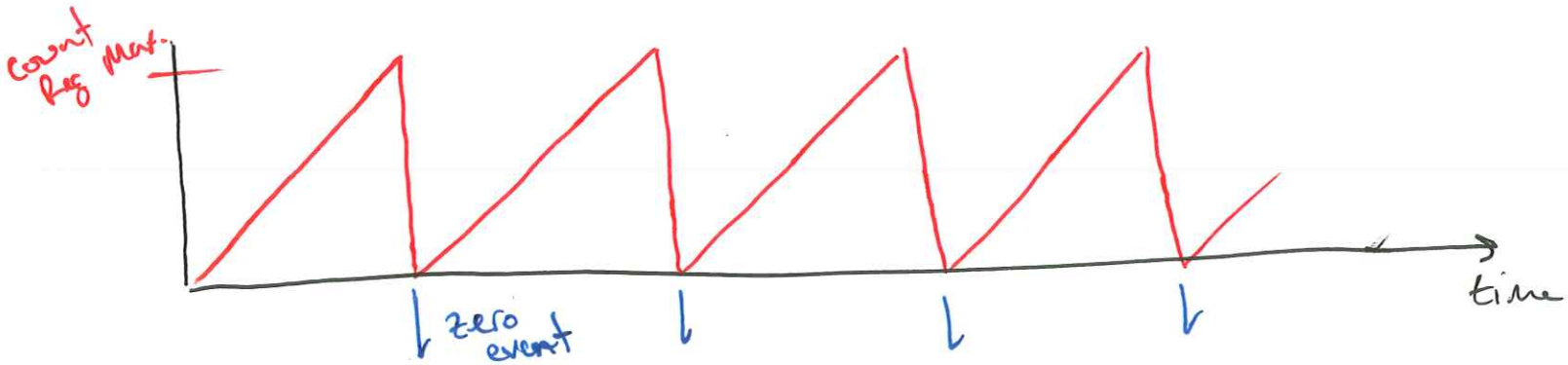
how many counts required for 1s? → 1,000,000

for 8-bit timer, max is $255 \cdot T_{TIMCLK} = 255\,\mu s$

How to increase max?

1) Slow down TIMCLK → Clock divider
2) Increase # of bits in counter
   ex. 16-bit timer at 1MHz → max is 65535 μs
   65.5 ms
3) Track # of timer resets that occur.

Longer time Periods → Tracking timer resets




```
uint64_t resets=0;
while(1){
    if(zero flag){
        zero flag = 0
        resets++
    }
}
```

```
if(resets == 1000){
    TOGGLE LED
    resets = 0
}
```

Timer Clock sources

High-speed CPU clock already exists in uC, eg. CPUCLK, SMCLK, SYSCLK...
BUSCLK

for our purposes, BUSCLK = 32 MHz

Low-speed clock typically exists, eg. LCLK, ACLK ... LFCLK = 32.768 kHz
our chip also has MFCLK = 4 MHz

↑ Pick 1!

→ Modify these clocks using a clock divider... N_{DIV}

$$f_{TIMCLK} = f_{INCLK} / N_{DIV}$$

$$f_{INCLK} = \begin{cases} f_{BUSCLK} \\ f_{MFCLK} \\ f_{LFCLK} \\ \vdots \end{cases}$$

$$T_{TIMCLK} = N_{DIV} T_{INCLK}$$

Timer Math

given f_{INCLK} → $f_{TIMCLK} = f_{INCLK} / N_{DIV}$

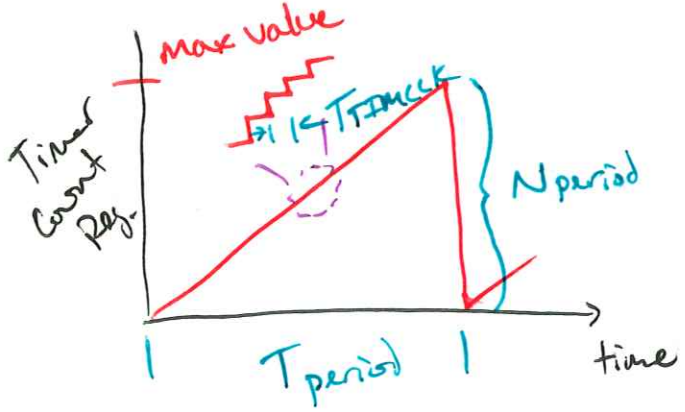
$T_{TIMCLK} = N_{DIV} / f_{INCLK}$

N_{DIV} typically integer with limited size.

$T_{period} = N_{period} \cdot T_{TIMCLK}$

N_{period} = max value + 1

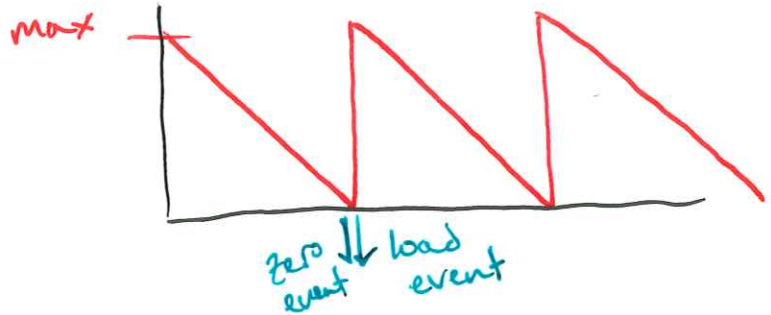
$N_{period} \leq 2^{\#\text{bits in timer}}$



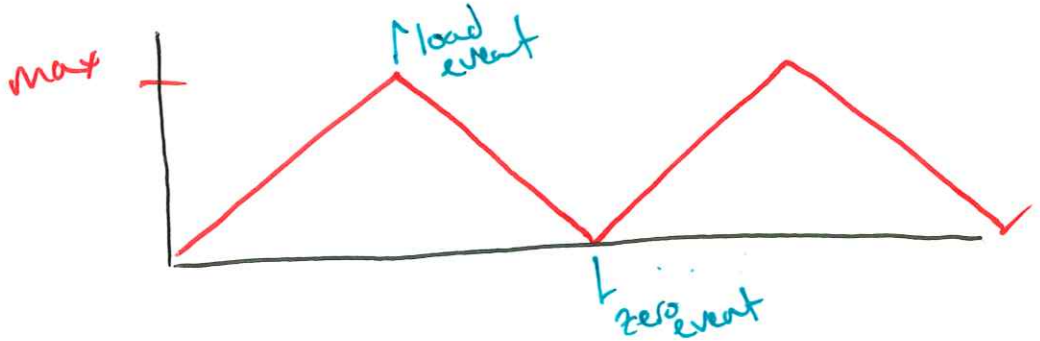
Timer Modes:

UP-Counting (already described)

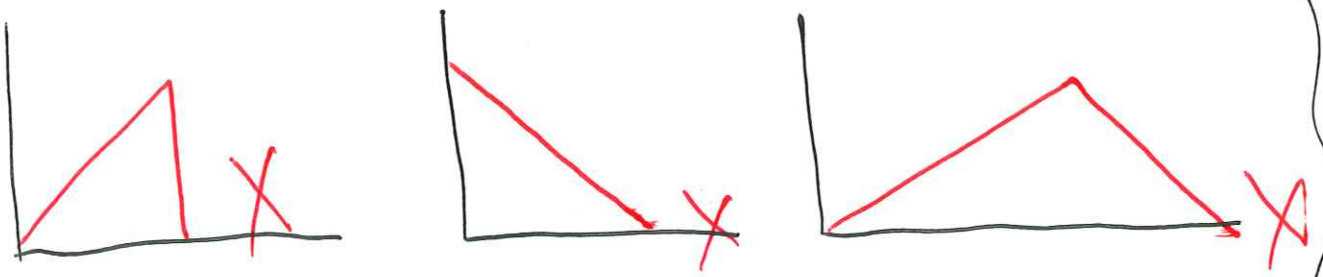
DOWN-counting



UP-DOWN counting



All periodic



ONE SHOT