

The Most Common Use Case for Timer/Counters

Although timer/counter systems within modern microcontrollers typically support a wide array of functionality, their most common use, at least in this course, is to simply “count” for some specific period of time. For example, if it was desired to keep track of some elapsing number of seconds, it would likely be helpful to configure a timer/counter with a period corresponding to one second, so that once enabled, an “overflow” (or possibly “underflow”, if counting down) occurs every one second.

To configure a period in such a manner, it is first useful to note the following regarding timer/counters available within the ATxmega128A1U. (Most of the given information is also applicable to other microcontrollers.)

- A timer/counter represents time with a digital value, otherwise known as a “count value” or a “tick value,” that either continually increases or continually decreases in a linear fashion (when the timer/counter is enabled). How often a “count” or “tick” is performed by a timer/counter depends primarily on the system clock frequency (strictly speaking, the peripheral system clock frequency) and a selected prescaler value. More specifically, an enabled timer/counter is designed to change its count value upon every elapsed amount of system-clock ticks that is a multiple of the selected prescaler, e.g., if a prescaler value of four is chosen for some timer/counter, this timer/counter will tick upon every four system-clock ticks.
- The period of a timer/counter represents an amount of timer/counter ticks that should occur before the relevant timer/counter overflows or underflows. If a counter with no prescaler counted up to 2, then there would be 3 ticks, one from 0 to 1, one from 1 to 2, and a third from 2 back to 0.
- The chosen prescaler of the timer/counter directly affects what the digital period value represents in regard of time – for example, the digital period value of 65,535 represents 65,536 system-clock ticks for a prescaler value of 1, whereas the same period value for a prescaler value of 1024 represents $65,536 \times 1024 = 67,108,864$ system-clock ticks!
- The availability of clock prescalers within a timer/counter module allow for, and require that, a user perform a tradeoff between the inversely related quantities of (1) *resolution*, defined here to be the smallest change in time representable by the timer/counter, i.e., the amount of time that elapses between one tick of the timer/counter, and (2) the maximum value of time representable by the timer/counter; in particular, a smaller prescaler allows for a higher resolution but a lower maximum time value, whereas a larger prescaler allows for a higher maximum time value but a lower resolution. In this course, the amount of resolution that a timer/counter has will often not be of particular importance, but the maximum time value representable will almost always be; namely, if it is intended that a timer/counter be configured with a period value corresponding to a specific period of time, it should always be determined whether or not there exist some prescaler(s) that allow for this period, and if so, that such an appropriate prescaler be chosen. If no prescaler allows for the desired period of time, it may be possible to slow down the relevant system clock frequency to increase the possible range of timing; otherwise, a good alternative, even when slowing down the system clock frequency is possible, is to instead set the period of the relevant timer/counter to a whole number factor of the desired period and implement control functionality to “count up to the desired period”, e.g., a period value that corresponds to sixty seconds is likely unachievable for the ATxmega128A1U, but setting a period value that corresponds to one second and then keeping track of sixty overflows, or sixty underflows, is normally quite possible.

Now, with an understanding of the above, it can be recognized that to calculate a digital period value corresponding to a specific period of time, one must utilize the relevant relationships between (1) the (peripheral) system clock frequency, (2) the prescaler value of a timer/counter, and (3) time. In particular, one can perform a form of dimensional analysis.

The Most Common Use Case for Timer/Counters

If a timer counter counts up to 1, then it will then go to 0, meaning that the counter counts, 0, 1, 0, 1, ... If the timer-counter counts up to 2, then it will count 0, 1, 2, 0, 1, 2, ... This generalizes such that if the counter counts up to N then there will be N+1 counts in the sequence.

As an example, assuming a system clock (SC) frequency of 2 MHz, a prescaler value of 1024, and a desired period of 10 ms, the following could be realized to calculate a corresponding digital timer/counter (TC) period value:

A TC prescaler of 1024 means that one TC *tick* occurs for every 1024 SC *ticks*. A SC frequency of 2 MHz means that 2×10^6 SC *ticks* occur for every one *second*. A desired 10 ms = 10×10^{-3} s

$$\frac{2 \times 10^6 \text{ SC ticks}}{1 \text{ s}} \times \frac{1 \text{ TC ticks}}{1024 \text{ SC ticks}} \times (10 \times 10^{-3} \text{ s}) = \text{Per} + 1$$
$$\frac{2 \times 10^4 \text{ TC ticks}}{1024} \approx 19.5 \text{ TC ticks} = \text{Per} + 1$$

$$\text{Per} = 18.5 \text{ TC ticks}$$

Generalizing the above strategy, it can be shown that the following formula calculates a *theoretically-corresponding* timer/counter period value for any valid period of time, with *Per* being the digital timer/counter period value, f_{sc} being the relevant system clock frequency, *Prescaler* being the relevant timer/counter prescaler value, and *t* being the specified period of time, in terms of seconds:

$$\text{Per} = t \left(\frac{f_{sc}}{\text{Prescaler}} \right) - 1$$

Note that the “-1” in the above equation is due to the fact that the counter (when counting up) reaches period before resetting to zero on the next count.