Study on TIMER, RTC and PWM

STUDY ON TIMERS

A timer (sometimes referred to as a counter) is a special piece of hardware inside many microcontrollers. Their function is simple: they count (up or down, depending on the configuration--we'll assume up for now). For example, an 8-bit timer will count from 0 to 255. Most timers will “roll over” once they reach their max value. So, our 8-bit timer would start over again from 0 once it reaches 255.

The basic timers consist of a 16-bit auto-reload counter driven by a programmable Prescaler. They may be used as generic timers for time-base generation, but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and can drive it through their trigger outputs. The timers are completely independent and do not share any resources.

Basic timer features include:

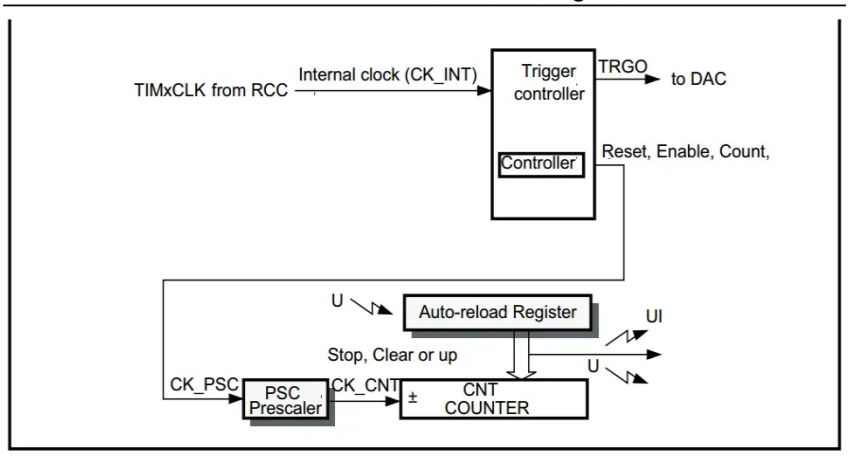
1. 16-bit auto-reload up-counter
2. 16-bit programmable Prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65536
3. Synchronization circuit to trigger the DAC
4. Interrupt/DMA generation on the update event: counter overflow

The main block of the programmable timer is a 16-bit up-counter with its related auto-reload register. The counter clock can be divided by a Prescaler. The counter, the auto-reload register, and the Prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

1. Counter Register (TIMx_CNT)
2. Prescaler Register (TIMx_PSC)
3. Auto-Reload Register (TIMx_ARR)

Basic timer block diagram



The general-purpose timers and use them to generate precise delays and the following are the essential registers to take note of:

1. **Timer Clock Enable (RCC->APBxENR[y]):** For a timer to time correctly, it needs a clock input. By default, our timers use the peripheral bus clock and to enable it for our timers, the specific APBxENR bit must be set. To know the specific bit to set, we can always consult the STM32F411 reference manual
2. **Timer Enable (TIMx->CR1[0]:** As with any other peripheral, our timer modules need to be enabled. Do note that timers can be enabled and disabled at any time.
3. **Timer Count Register (TIMx->CNT):** We can read the current counter value of the timer from this register. It is good practice to set this register to 0 before we enable the counter to ensure that we get our expected delay.
4. **Timer Auto-Reset Register (TIMx->ARR):** When the timer value reaches the value set on the period register, the timer value is reset to 0 and an interrupt is generated, if enabled. This register is also referred to as the timer period register.
5. **Update Generation Bit (TIM2->EGR [0]):** Set this bit to 1 to automatically reset all necessary registers to restart counting.

STUDY ON RTC

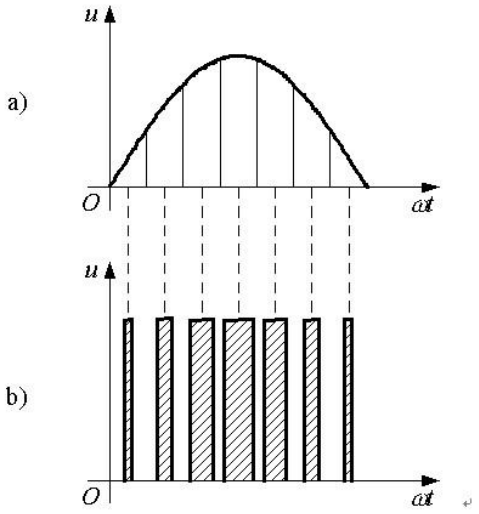
Most of the STM32 devices have RTC (Real Time Clock) built in which can keep track of the current time and date. RTC can be used for chronometers, alarm clocks, watches, small electronic agendas, and many other devices and today we are going to learn HOW to access internal RTC in STM32.

Here are some functions where RTC is used: -

1. **HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR1, 0x32F2);** it is a backup register and is used to store time.
2. **sprintf((char*)time,"%02d:%02d:%02d",gTime.Hours, gTime.Minutes, gTime.Seconds);** This function is used to convert the values into the char so that we can display it on the LCD
3. **sAlarm.AlarmTime.x = (hexa-decimal number);** here x is hours, minutes, seconds, subseconds. This allows you to put alarms. The **HAL_RTC_SetAlarm_IT(&hrtc, &sAlarm, RTC_FORMAT_BCD) != HAL_OK** allows us to set the alarm.
4. **HAL_RTC_AlarmAEventCallback** is called when the current time and date match the alarm time and date. In our example, it is set to 1 second in the future.

STUDY ON PWM

PWM, short for Pulse Width Modulation, is a modulation technique used to control the analog circuits via MCU digital outputs. It is widely applied, ranging from measurement and communication to power control and conversion.



The figure above shows a PWM signal. Figure b) shows the MCU digital signal. And figure a) shows the corresponding analog signal when the digital output is connected to the power device, like motor.

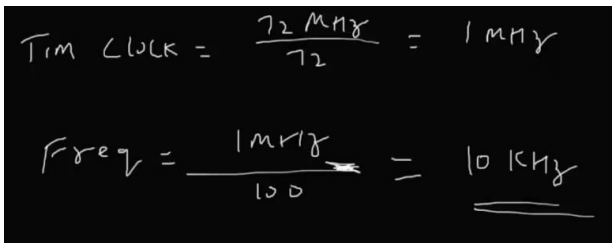
The Pulse width modulation mode allows you to generate a signal with a frequency which is determined by certain registers: -

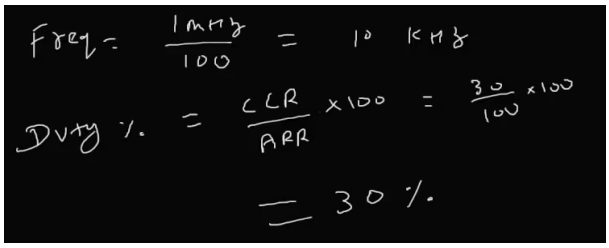
1. The value stored in the TIMx_ARR (auto-reload register) register
2. The duty cycle value which is determined by the value of the TIMx_CCRx (capture/compare register) register.
3. The value stored in the TIMx_CNT (time counter) register

The figure below shows the PWM edge-aligned mode (up-counting configuration). OCxREF is output compare signal, and CCxIF is interrupt flag. For CCRX=4, when the value in TIMx_CCRx is lower than the counter, the output is in high level, or else in low level.

The ARR register stores the value which defines the end of the pulse while the CCR register usually changes the duty cycle of the PWM which is a useful function until it needs to change rapidly.

In PWM mode (1 or 2), and TIMx_CCRx are always compared to determine whether TIMx_CCRx & TIMx_CNT or TIMx_CNT & TIMx_CCRx (depending on the direction of the counter).





From the formulae, the PWM Frequency is found by dividing the Time Clock by its ARR stored value of which Time Clock is found by dividing the set timer frequency by the external crystal with the Prescaler.

A prescaler is an electronic counting circuit used to reduce a high frequency electrical signal to a lower frequency by integer division. It is a way of reducing the PWM frequency to its smallest term. Many of these prescalers are specified for each timer which can be checked in the datasheet. The Prescaler Register and the ARR Registers are set up in a way that they add a 1 to the value. So whatever value we enter for the PSC, 1 will be added to that value. This is why we enter 1 less than the actual value.

The value set in the CCR decides the width of the pulse and its duty cycle. The duty cycle is the percentage of the ratio of the terms in CCR and ARR registers.

Figure 221. Edge-aligned PWM waveforms (ARR=8)