Introduction

A timer is a free-running counter with a counting frequency that is a fraction of its source clock. The counting speed can be reduced using a dedicated prescaler for each timer.

The most advanced timers in an STM32 microcontroller have several features:

- They can be used as time base generator (which is the feature common to all STM32 timers).
- They can be used to measure the frequency of an external event (input capture mode).
- To control an output waveform, or to indicate when a period of time has elapsed (output compare mode).
- To generate PWM signals in edge-aligned mode or center-aligned mode independently on each channel (PWM mode).

Timer categories

The most important type of timers that can be found on the STM32 are:

- **Basic timers**: timers from this category are the simplest form of timers in STM32 MCUs. They are 16-bit timers used as time base generator, and they do not have output/input pins;

- **General purpose timers**: they are 16/32-bit timers (depending on the STM32-series) providing the classical features that a timer of a modern embedded microcontroller is expected to implement like capture and compare;

- **Advanced timers**: In addition to the features found in a general purpose timer, they include several features related to motor control and digital power conversion applications;

Basic timers

To initialize a basic timer, the structure above is used:

```
typedef struct {
  uint32_t Prescaler;
  uint32_t CounterMode;
  uint32_t Period;
  uint32_t ClockDivision;
  uint32_t RepetitionCounter;
} TIM_Base_InitTypeDef;
```

The structure's members above, are reflected into the configuration that can be setted up from STMCubeIDE.

Then, in order to initialize the timer, the function

```
void HAL_TIM_Base_Init(TIM_HandleTypeDef *htim);
```

and then it is necessary to start the timer, by calling the function:

```
void HAL_TIM_Base_Start(TIM_HandleTypeDef *htim);
```

Using interrupt with basic timers

After having enabled the interrupt source for the selected timer, the function below can be implemented, in order to get notified when the timer reaches the auto-reload register value:

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);
```

To start the basic timer in interrupt mode, it is necessary to call, after having initialized the timer, the function:

```c
void HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim);
```

In interrupt mode, an event is generated according to the following rule:

$$T_{event} = \frac{T_{clk}}{(1 + Prescaler)(1 + Period)}$$

where Prescaler and Period are interpreted as the members of the `TIM_Base_InitTypeDef` structure.

API Documentation

- See `dm00189702-description-of-stm32f7-hal-and-lowlayer-drivers-stmicroelectronics.pdf`, pg. 1017, 1025 to see the basic structure and the functions.
- See `dm00189702-description-of-stm32f7-hal-and-lowlayer-drivers-stmicroelectronics.pdf`, pg 1028 for the callbacks.

Links

Slides on PWM STM application note on timers STM presentation on timers