**HAL. We study PWM (PWM). Flashing LEDs smoothly**

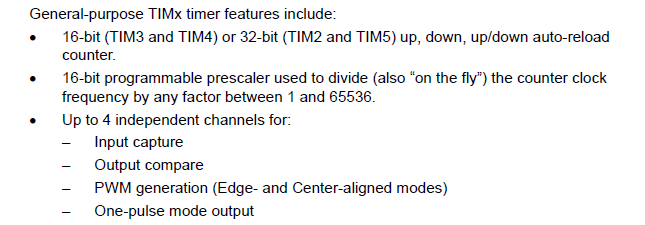
Today we will continue to use the HAL library. Also today we will touch on such technology as **PWM** ( **PWM** ).

The subtleties of this technology are discussed in detail in the lesson on **AVR** controllers . I think it makes no sense to repeat it again.

We will study exactly how PWM is implemented in **STM32** controllers , in particular in **STM32F407** , with which we will continue to work.

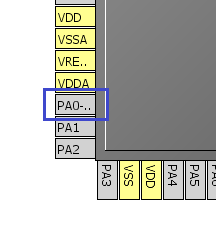
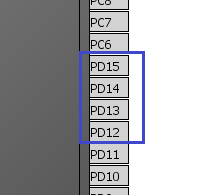
The easiest way to manage the **PWM** technology from these controllers is through timers. We already had a lesson on the timers. What timer does what, we have already briefly become acquainted, so it will be much easier for us.

As a timer for **PWM,** we take timer 4. This timer is 16-bit. Up to 4 separate channels are available. Also in the technical documentation for the controller indicated that the timer can act as a PWM-generator

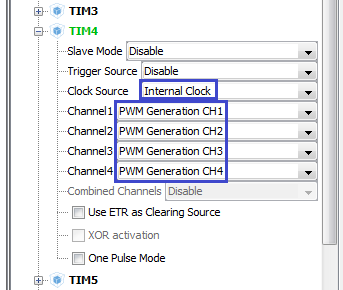


Just as before, the project is created from the very first ( **TEST001** ), call it **MYPWM**

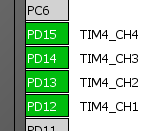
Run the **Cube the MX** , is disabled. firstly all the legs of the LED ports, as well as the button leg



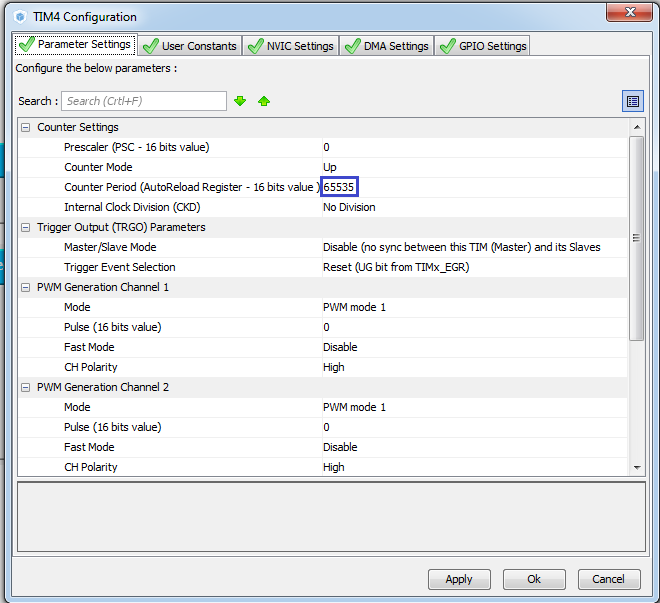
Turn on the timer by selecting the internal clock in the drop-down list and enable the channels in it - all 4, selecting **PWM** there



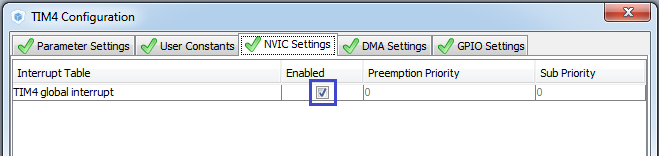
As we can see, the legs of the ports responsible for the LEDs on the board have switched themselves into a certain alternative mode

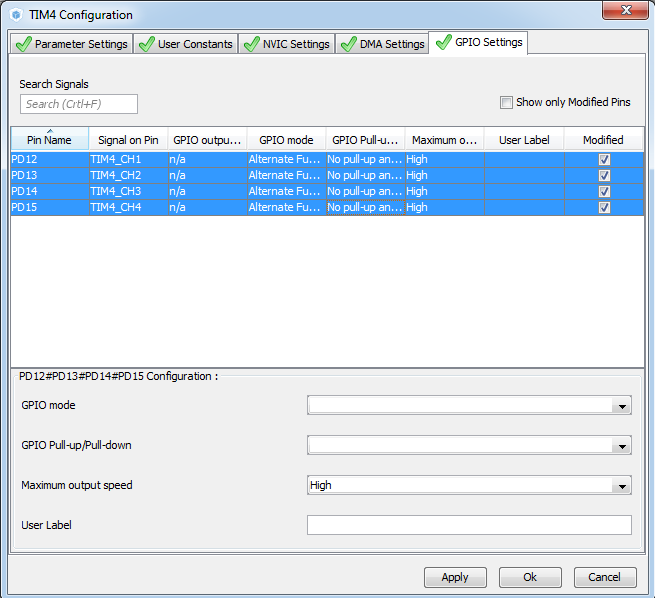


In the timer settings in the *Configuration* parameter, the Counter Period parameter is set to **65535**



Just in case, we'll turn on the interrupts and set all **high** speed ports to all 4 ports . Generating project





We add the initialization code

  / \* Initialize all configured peripherals \* /

  MX\_GPIO\_Init ();

  MX\_TIM4\_Init ();

  / \* USER CODE BEGIN 2 \* /

**HAL\_TIM\_PWM\_Start (& htim4, TIM\_CHANNEL\_1);**

**HAL\_TIM\_PWM\_Start (& htim4, TIM\_CHANNEL\_2);**

**HAL\_TIM\_PWM\_Start (& htim4, TIM\_CHANNEL\_3);**

**HAL\_TIM\_PWM\_Start (& htim4, TIM\_CHANNEL\_4);**

  / \* USER CODE END 2 \* /

In the main function, we also write

  / \* USER CODE BEGIN 1 \* /

**uint32\_t i, d;**

  / \* USER CODE END 1 \* /

And in an infinite cycle we write

/ \* USER CODE BEGIN 3 \* /

**for (i = 0; i <= 524288; i ++)**

**{**

**if (i <65536) TIM4-> CCR1 = i;**

**else if ((i> 65535) && (i <131072)) TIM4-> CCR1 = 131071-i;**

**else if ((i> 131071) && (i <196608)) TIM4-> CCR2 = i-131072;**

**else if ((i> 196607) && (i <262164)) TIM4-> CCR2 = 262164-i;**

**else if ((i> 262163) && (i <327680)) TIM4-> CCR3 = i-262164;**

**else if ((i> 327679) && (i <393216)) TIM4-> CCR3 = 393216-i;**

**else if ((i> 393216) && (i <458752)) TIM4-> CCR4 = i-393216;**

**else TIM4-> CCR4 = 524288-i;**

**for (d = 0; d <300; d ++)**

**{**

**}**

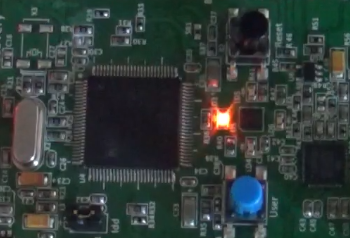
**}**

  }

  / \* USER CODE END 3 \* /

Now we will collect the code, we will sew the controller.

After this we should see the following picture



Our light-emitting diodes are lighted in turn, and they are lit smoothly and drip too smoothly. That's how we felt the effect of pulse-width modulation visually.