PWM generation on STM32 Microcontrollers using HAL

You can use the STM32CubeMX tool to create the necessary config. files to enable PWM output. This works great once set. However, you might want to add an additional PWM output later. And while in mid coding you don't want to generate a new config. file and start all over again.

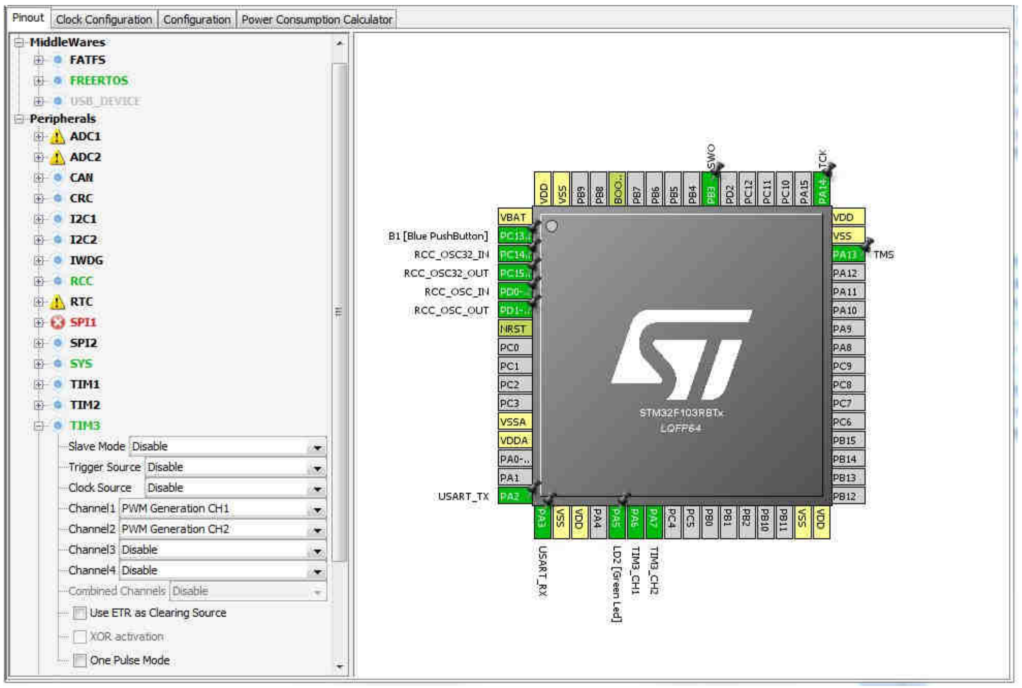
I'm going to show you here, how to change the PWM settings by yourself.

This tutorial uses the following equipment:
- NUCLEO-F103RB Board
- Keil uVision 5 with the necessary packages for Nucleo boards installed
- STLink USB Driver
- STM32CubeMX

STM32CubeMX

Generating the config. files from STM32CubeMX.

1. Open STM32CubeMX and open a new project.
2. Select the Nucleo-F103RB from the Borards tab
3. Enable FreeRTOS
4. Set the RCC (HSE & LSE) to Crystal/Ceramic Resonator
5. Enable the USART2 port in Asynchronous mode
6. Enable Timer 3 Channel 1 and Channel 2 (PWM Generation mode)
7. Go to Project > Generate code
8. Enter a project name and select MDK-ARM V5
9. Generate the code and open the project in Keil uVision



Slave Mode: Disable
Trigger Source: Disable
Clock Source: Disable
Channel1: PWM Generation CH1
Channel2: PWM Generation CH2
Channel3: Disable
Channel4: Disable
Combined Channels: Disable
☐ Use ETR as Clearing Source
☐ XOR activation
☐ One Pulse Mode

Now let's see what the code generator did.

In the main.c file of the outputted uVision project, on the line 56, a function called

```
56  static void MX_TIM3_Init(void);
```

is initialised. What this function does should be visible somewhere around line 169.

```
168  /* TIM1 init function */
169  void MX_TIM3_Init(void)
170  {
171
172    TIM_MasterConfigTypeDef sMasterConfig;
173    TIM_OC_InitTypeDef sConfigOC;
174
175    htim3.Instance = TIM3;
176    htim3.Init.Prescaler = 0;
177    htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
178    htim3.Init.Period = 0;
179    htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
180    HAL_TIM_PWM_Init(&htim3);
181
182    sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
183    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
184    HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig);
185
186    sConfigOC.OCMode = TIM_OCMODE_PWM1;
187    sConfigOC.Pulse = 0;
188    sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
189    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
190    HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_1);
191
192    HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2);
193  }
194
```

The Struct

```
42  TIM_HandleTypeDef htim3;
```

used here is defined on line 42.

Now pay attention to the listing above. On line 175 the initialisation struct is told, which timer it is assigned to, TIM3. The one statement you should note right now is the one on line 178, which is the period duration of the PWM. Modify this number to 4095, if you want a 12 Bit PWM.

Another thing that happens here, is the initialisation of the PWM on line 180. We'll get to that later.

On line 187 is the second thing you should note, the pulse duration. Modify this number to change the PWM ON-time.
On lines 190 and 192 the two channels of the timer module are initialised, however, this does not affect the GPIO state. The GPIOs have to be declared as output somewhere else.

Now I am referring back to the function call on line 180.
If we look at what this function does (Right-Click, Go To Definition of '`HAL_TIM_PWM_Init()`'), we'll see that in the now open file (stm32f1xx_hal_tim.c) a whole function is present, to initialise things. On line 996 another function (`HAL_TIM_MspInit`) gets called, this is the one we're looking for.

How to add / remove / change PWM channels

Example shows how to add CH3 to TIM3

Open the definition of it. (stm32f1xx_hal_msp.c)
Here the GPIOs are declared as Outputs.

```
    GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

This looks exactly like the output declaration of the LD2_pin in the main header.
So if you wanted to add the Channel 3 on TIM3 you would have to do the following.

The GPIO Pin has to be enabled like the other PWM outputs. For CH3 this would be PB0.
So in this file, where the other outputs are defined, simply add the GPIO pin as a normal push-pull output. In this case, CH3 is not on the same GPIO port as CH1 & CH2, so you need to add a new IO port declaration.

```
    GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);     // Port A

    GPIO_InitStruct.Pin = GPIO_PIN_0;           // CH3 on PB0
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);     // Port B
```

Save and close it and head back to the main fail to the part, where the timer initialisation takes part.
Simply add the channel 3 below the initialisation lines of channel 1 & 2.

```
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_1);

  HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2);

  HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_3);
```

That’s it, this should add channel 3 as a new PWM channel to TIM3.
Note: Initialising the GPIO inside the regular `MX_GPIO_Init()` function won’t work.

Using and changing PWM inside the main routine

Given a function to handle everything.

```c
/* USER CODE BEGIN 4 */
void setPWM(TIM_HandleTypeDef timer, uint32_t channel, uint16_t period,
uint16_t pulse)
{
  HAL_TIM_PWM_Stop(&timer, channel);    // stop generation of pwm
  TIM_OC_InitTypeDef sConfigOC;
  timer.Init.Period = period;           // set the period duration
  HAL_TIM_PWM_Init(&timer);  // reinititialise with new period value

  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = pulse;              // set the pulse duration
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  HAL_TIM_PWM_ConfigChannel(&timer, &sConfigOC, channel);

  HAL_TIM_PWM_Start(&timer, channel);   // start pwm generation
}
/* USER CODE END 4 */

```

Don’t forget to add the function Prototype.

```c
/* USER CODE BEGIN PFP */
/* Private function prototypes ----------------------------------------*/
static void setPWM(TIM_HandleTypeDef, uint32_t, uint16_t, uint16_t);
/* USER CODE END PFP */
```

Example:

```c
for(int i=0; i<256; i++){
  setPWM(htim3, TIM_CHANNEL_1, 255, i);
  osDelay(10);
}
```

Document Created by Simon Burkhardt

This tutorial is very basic and might not show the best way to use the STM32 environment.
It still might help you get into the whole HAL philosophy of STM if you are coming from another platform.

Simon Burkhardt
electronical engineering
simonmartin.ch

History:

V1.0	tested the code, created this document
V1.0.1	added contact info and cc0 notice