Pulse Width Modulation

Prepared by:
Omar Samir

Pulse Width Modulation

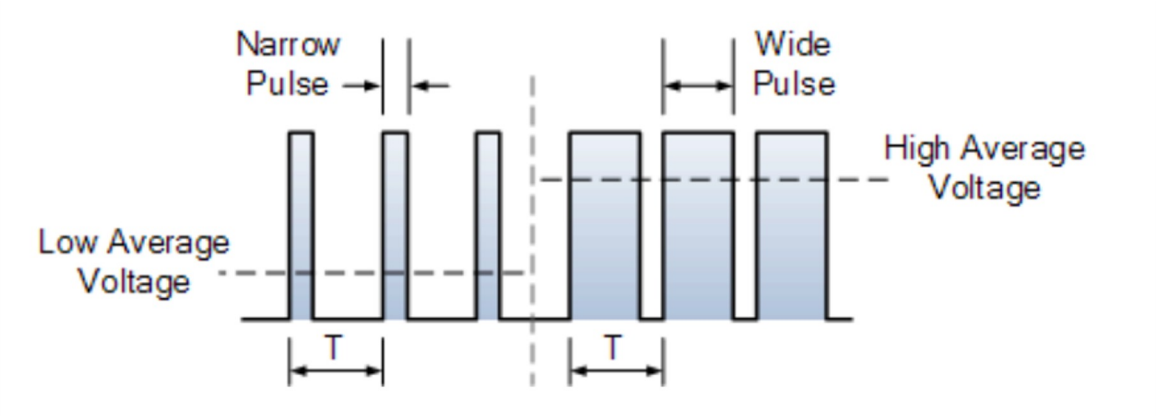
- PWM is a way to control analog devices with a digital output
  - Motors, actuators, speakers, ...
- PWM is not a true output analog signal
- PWM fakes an analog-like result by applying power in pulses

PWM Characteristics

A PWM signal is characterized by:

- Duty Cycle: The percentage of high voltage in a period T
- Frequency: Depends on the application

Average voltage = Duty Cycle x High voltage level



PWM Generation in STM32F40xxx

- PWM signals are generated using the MCU timers
- Our STM32F407VG has a total of 14 timers as follows:
  - 2 advanced timers: (TIM1, TIM8)
  - 10 general purpose timers: (TIM2, 3, 4, 5, 9, 10, 11, 12, 13, 14)
  - 2 basic timers: (TIM6, TIM7)
- Except basic timers, All other timers can be used to generate PWM output
- Each timer has a number of independent channel to be used

PWM Generation in STM32F40xxx

- PWM channels are not dedicated to fixed GPIO port pins

Timer type	Timer	Number of independent channels
Advanced	1, 8	4
General Purpose	2, 3, 4, 5	4
	9, 12	2
	10, 11, 13, 14	1

MikroC PWM Library

- Facilitates the use of timers for PWM generation
- Can be included as we included the Button library before
- Has mainly four functions to deal with timers for PWM
  - PWM_TIMn_Init
  - PWM_TIMn_Set_Duty
  - PWM_TIMn_Start
  - PWM_TIMn_Stop

MikroC PWM Library

unsigned int PWM_TIMn_Init(unsigned long freq_hz);

- Initializes timer n in PWM mode
- Takes as input the PWM frequency in Hz
- Returns the calculated timer period

Example:

```
unsigned int period = 0;

period = PWM_TIM1_Init(25000); // initializes timer 1 in PWM mode with 25 kHz frequency
```

MikroC PWM Library

void PWM_TIMn_Set_Duty(unsigned int duty, char inverted, char channel);

- Changes duty ratio for Timer module in PWM mode for ST MCUs.
- Inputs:
  - duty: PWM duty ratio, takes values from 0 to timer period returned by PWM_TIMn_Init
  - inverted: inverted and non inverted PWM signals
  - channel: desired PWM channel

Example:
// sets timer 8 duty ratio to 200, non inverted signal, channel 4
PWM_TIM8_Set_Duty(200, _PWM_NON_INVERTED, _PWM_CHANNEL4);

Inverted parameter	
Description	**Predefined library const**
Inverted PWM signal	_PWM_INVERTED
Non-inverted PWM signal	_PWM_NON_INVERTED

Channel parameter	
Description	**Predefined library const**
Channel 1	_PWM_CHANNEL1
Channel 2	_PWM_CHANNEL2
Channel 3	_PWM_CHANNEL3
Channel 4	_PWM_CHANNEL4

MikroC PWM Library

void PWM_TIMn_Start(char channel, const module_Struct *module);

- Starts Timer n in PWM mode
- Inputs:
  - channel: desired PWM channel
  - module: mapping the channel to a GPIO port pin

```
_GPIO_MODULE_TIM2_CH1_PA0
_GPIO_MODULE_TIM2_CH1_PA15
_GPIO_MODULE_TIM2_CH1_PA5
_GPIO_MODULE_TIM2_CH2_PA1
_GPIO_MODULE_TIM2_CH2_PB3
_GPIO_MODULE_TIM2_CH3_PA2
_GPIO_MODULE_TIM2_CH3_PB10
_GPIO_MODULE_TIM2_CH4_PA3
_GPIO_MODULE_TIM2_CH4_PB11
_GPIO_MODULE_TIM3_CH1_PA6
_GPIO_MODULE_TIM3_CH1_PB4
_GPIO_MODULE_TIM3_CH1_PC6
_GPIO_MODULE_TIM3_CH2_PA7
_GPIO_MODULE_TIM3_CH2_PB5
_GPIO_MODULE_TIM3_CH2_PC7
_GPIO_MODULE_TIM3_CH3_PB0
_GPIO_MODULE_TIM3_CH3_PC8
_GPIO_MODULE_TIM3_CH4_PB1
_GPIO_MODULE_TIM3_CH4_PC9
_GPIO_MODULE_TIM4_CH1_PB6
_GPIO_MODULE_TIM4_CH1_PD12
_GPIO_MODULE_TIM4_CH2_PB7
_GPIO_MODULE_TIM4_CH2_PD13
_GPIO_MODULE_TIM4_CH3_PB8
_GPIO_MODULE_TIM4_CH3_PD14
```

Example:

```
// starts timer 3 PWM generation to channel 2 and maps the output to PC7
PWM_TIM3_Start(_PWM_CHANNEL2, &_GPIO_MODULE_TIM3_CH2_PC7);
```

MikroC PWM Library

void PWM_TIMn_Stop(char channel);

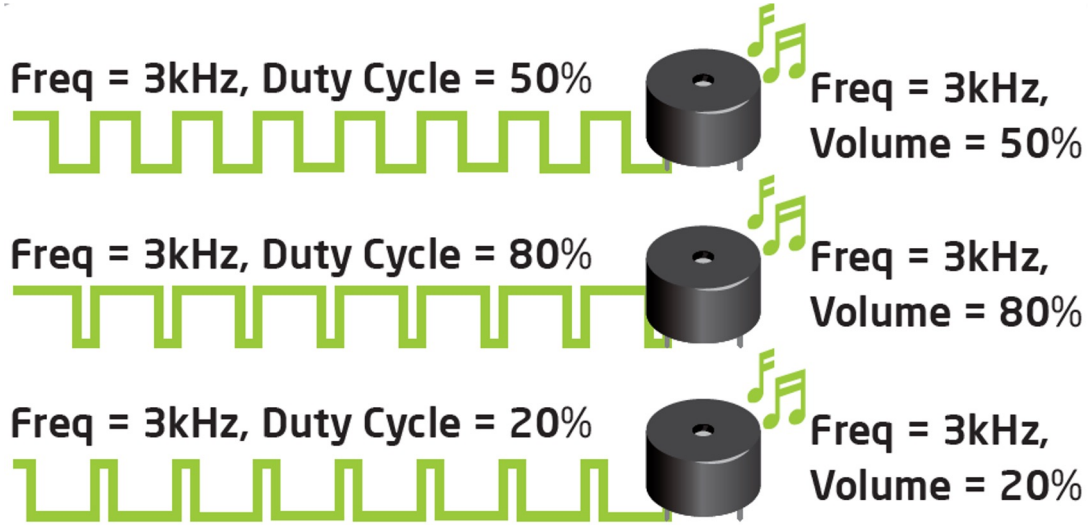
- Stops timer n in PWM mode
- Takes channel as input

Example:

```
// stops timer 4 channel 1 from generating PWM output
PWM_TIM4_Stop(_PWM_CHANNEL1);
```

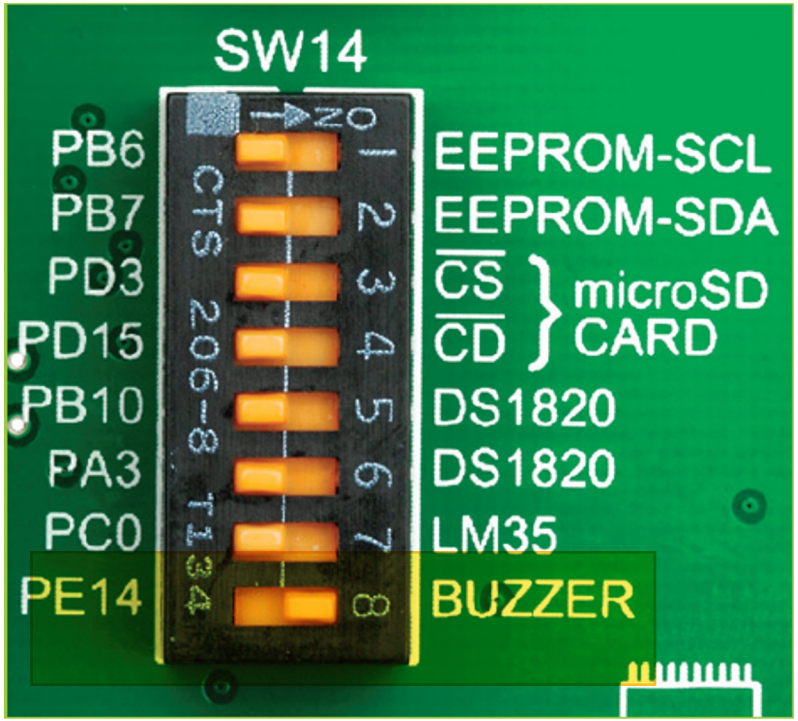
EasyMx Pro Buzzer

- Create sound when provided analog signal
- Connected to pin PE14
- Can be used by generating PWM output to PE14



Enable the Buzzer

To enable the Buzzer you need to push SW14.8 to ON position



Requirements

Requirement 1

It is required to to generate a fixed non inverted 500 Hz PWM signal with 20% duty ratio on PE14 and enable the Buzzer.

Hints:

- Don’t forget to include the PWM library from library manager
- To find out what timer and channel to work with you can use code assistant
  - Type _GPIO_MODULE_TIM
  - Press CTRL + SPACE
  - Search for PE14

Requirement 2

It is required to fade PE14 LED (and hence the Buzzer). Use timer 1 to output a 3.8 kHz PWM signal with a duty ration that goes from 0 to 100% then from 100% to zero and so on.

Hint:

- Use Delay_us(time_in_microseconds) in your loop.

Requirement 3

It is required to use timer 4 to generate 4 kHz PWM signals. Use the 4 channels of timer 4 and map them to PD12, PD13, PD14, PD15. The LEDs should fade from OFF to ON sequentially then from ON to OFF in the same order and so on.