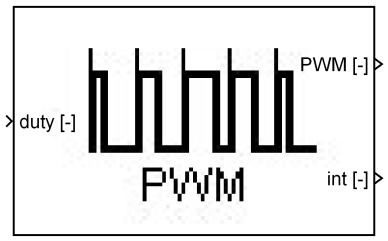
PWM – Pulse Width Modulated Signal (PELOPWM02MSEREF) Documentation

September 27, 2021



Description

This block generates a PWM[1] signal to drive a powerstage.

Features

- Switchable between three types:
  - left
  - right
  - center
- Supplies variable step and discrete solvers
- Can be *turned off*. In this case, the PWM output vries between 0 and 1 (does not work with the *switch model - EMPSMYSW01MSEREF*).

[1] Pulse Width Modulated.

Application area

Model assumptions and limits

The simulation time must start at zero.

Model equations

This block computes *when* (during a cycle period) a branch (half bridge) of the power electronic is set to upper (resp. lower) voltage level. A period has the length T_d. The duty cycle variable *duty* gives *how long* the controlled half bridge is set to each voltage level. (See background documentation for more information.)

If the upper voltage level is required, the PWM output variable is set to 1. Vice versa, it is set to 0 for the lower voltage level. Thus,

- The PMW signal is 1 during the cycle period, if $duty = 1$.
- The PMW signal is 50% 1 and 50% 0, if $duty = 0$.
- The PMW signal is always 0, if $duty = -1$.

and so on.

The block has three operation modes, which are “left”, “right” and “center”. In the left-mode the PWM–signal is (during a cycle period) *first* set to 1 and *then* set to 0. Accordingly, in the right–mode it is first set to 0 and then to 1. And in the center–mode the signal is 1 “in the middle” of a cycle period. This is illustrated in figure 1.

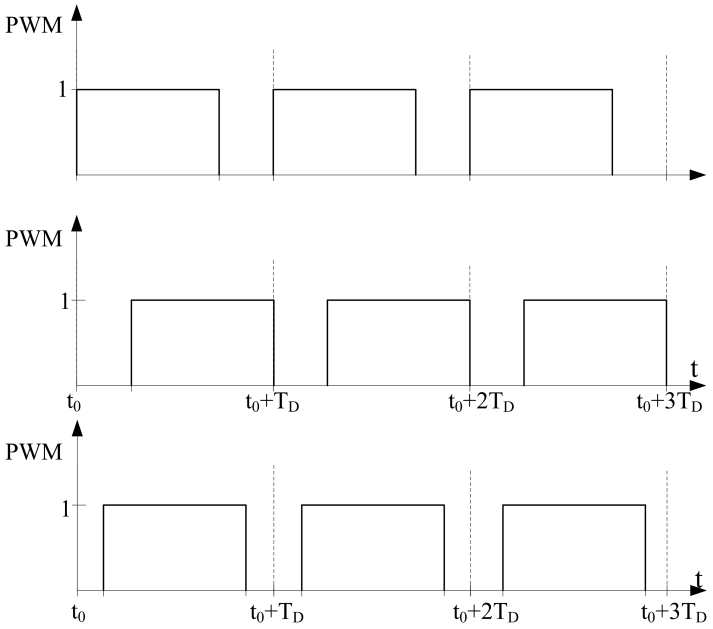


Figure 1: Example of left, right and centered pulse width modulation

Thus, we have: Assuming t_0 is the time at the beginning of the cycle period and $duty(t_0)$ is the value of the duty–signal at t_0. Hence, in mode **left**, we have:

$$\begin{aligned} PWM = 1 \quad &\text{if} \quad t_0 \leq \ t \ \leq t_0 + duty(t_0) \cdot T_d \\ PWM = 0 \quad &\text{if} \quad t_0 + duty(t_0) \cdot T_d < \ t \ < t_0 + T_d \end{aligned}$$

In mode **right** we have:

$$\begin{aligned} PWM = 0 \quad &\text{if} \quad t_0 \leq \ t \ \leq t_0 + (1 - duty(t_0)) \cdot T_d \\ PWM = 1 \quad &\text{if} \quad t_0 + (1 - duty(t_0)) \cdot T_d < \ t \ < t_0 + T_d \end{aligned}$$

Finally in mode **center** we have:

$$\begin{aligned} PWM = 0 \quad &\text{if} \quad t_0 \leq \ t \ \leq t_0 + (1 - duty(t_0)) \cdot T_d/2 \\ PWM = 1 \quad &\text{if} \quad t_0 + (1 - duty(t_0)) \cdot T_d/2 < \ t \ \leq t_0 + (1 - duty(t_0)) \cdot T_d/2 + duty(t_0)T_d \\ PWM = 0 \quad &\text{if} \quad t_0 + (1 - duty(t_0)) \cdot T_d/2 + duty(t_0)T_d < \ t \ < t_0 + T_d \end{aligned}$$

holds.

If turned *off* no PWM is generated. Then, the output varies continuously between 0 and 1.

Continuous solvers/fixed step solver

If the block is simulated by a continuous solver the module *GetTimeOfNextVarHit* is used in order to speed up the simulation. (Since it calculates the next time step at once – without using the *ZeroCrossings* concept.)

However, this module cannot be used with a fixed step solver. Hence, the sample time is set to "inherited" in this case and *GetTimeOfNextVarHit* will not be used. When the option "Discrete" is chosen, the block will be simulated with the specified sample time – independent of the solver mode.

In standard text books, typically the PWM generation is introduced as a comparison of the duty–cycle–signal with a sawtooth (mode "left" and "right") or triangle signal ("center"). This concept takes changes of the duty–cycle–signal into account, which are not considered here.

However there are two substantial arguments for using the MSE–concept: First the duty–cycle–signal changes very slowly compared to pulse frequency and neglecting the changes makes the simulation much more faster, since it makes the *ZeroCrossings* detection dispensable.

The second point is: In a digital environment the duty–cycle–signal does not change during a cycle period anyway, because the duty–cycle comes from a controller which is usually slower clocked than the PWM generation.

Model validation

Code Generation

To inquire if it is principally possible to generate code from this block, please consult the related *Code generation* section in the overview documentation.

Parameters

Type[2]	Name	Description	Symbol	Unit	Default	Values
E	`mode`	Type			middle centered	left aligned right aligned middle centered
F	`Tpl`	Period length	T_{pl}	s	10^{-3}	$[10^{-7}, 10^{3}]$
B	`disc`	Discret			0	true, false
B	`off`	OFF			0	true, false
F	`Td`	Sampling time	T_{d}	s	10^{-6}	$[10^{-12}, 10^{3}]$

[2]B: Boolean parameter, E: Enumerated values parameter, F: Float parameter

Ports

Limits of the duty signal: [-1,1]

Inputs

Direction	Type	Name	Symbol	Description	Unit
input	Float	duty	duty	Length of duty cycle	-

Outputs

Direction	Type	Name	Symbol	Description	Unit
output	Float	PWM	PWM	PWM signal	-
output	Vector/Bus	Internals		Internals output vector	

Internal Variables

Type[3]	Name	Symbol	Description	Unit
II	saturation	$saturation$		
II	status	$status$		
II	num	num		
IF	duty_work	$duty_{work}$		-
IF	tnext1	t_{next}		-
IF	time	$time$		-

[3] II: Internal integer, IF: Internal float