



Pulsing Unit

Application Note 152

This document describes the setup and usage of a *Pulsing Unit (PU)*. *PU* is a *Software Option* (see [2]), which enables encoder *Option Modules EN* or *EH*, to fire pulses up to 10 MHz in reference to the connected encoder or path planner position. The *Pulsing Unit works only with analog encoders*.

Table of Contents

1 Purpose and Usage.....	2	4.4 Pulse Configuration.....	7
2 Preconditions.....	3	4.5 Signals.....	7
3 Pinout.....	3	5 Examples.....	8
4 TAM Registers.....	3	Glossary.....	8
4.1 Output.....	4	References.....	8
4.2 Source.....	4	Revision History.....	9
4.3 Mode.....	5		

1 Purpose and Usage

A *Triamec Pulsing Unit* usually controls an external device by triggering a process with a digital electrical signal. The triggers are fired in reference to a position information of the motion system at up to 10MHz.

The *Pulsing Unit* is controlled via *Tama* program, *TAM API* or by main controllers via fieldbus. As it is fully controllable with drive registers, simple use cases can also be set up manually via *TAM System Explorer*.

For the explanations on the usage of the *Pulsing Unit*, *Triamec* uses the following terms (Figure 1).

Term	Description
Pattern	Contains all pulses to form the target image or structure.
Row	A pulse pattern is configured by a set of rows. This is because the Pulsing Unit only has position information from one dimension.
Sequence	A row can have one or more sequences of equidistant pulses.
Pulse	The Pulsing Unit output

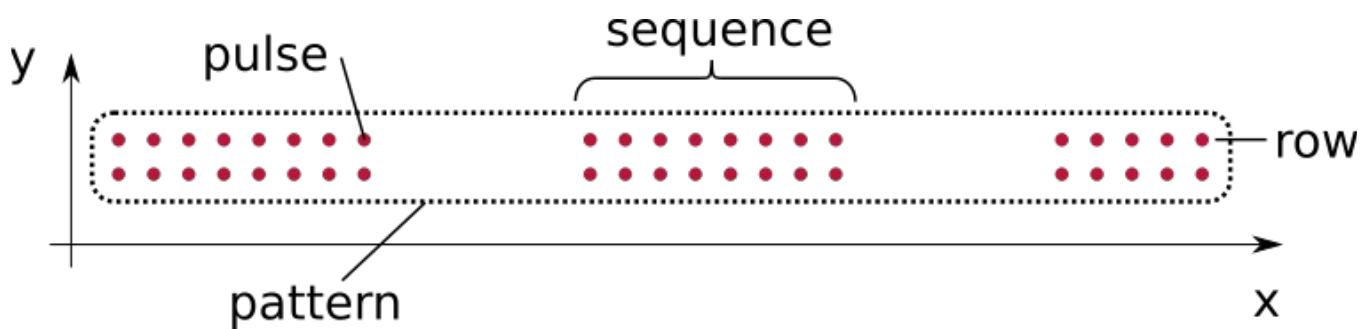


Figure 1: Visualization of Pulsing Unit specific terms

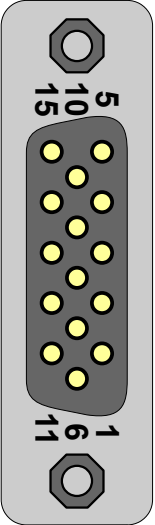
2 Preconditions

The following are mandatory for proper functionality.

- Only available on *Option Module Encoders EN* and *EH* (see [1])
- Only available in combination with analog sin/cos encoders, except using `PU_Source = PathPlanner`.
- Correct encoder settings must be configured, also in case of `PU_Source = PathPlanner`.
- Incompatible with `EncoderTopology = Standard`.

3 Pinout

The encoder pinout with enabled *Software Option PU*, is based on the analog encoder interface and adds the pulse output as *TTL* and *RS422* signals.

Pin Layout X10/X11	Pin	Name	Encoder
15-pin female D-Sub socket 	1	+5VDC	Encoder Supply
	2	ChA+	Channel A positive, Cosine 1Vpp
	3	ChB+	Channel B positive, Sine 1Vpp
	4	ChZ+	Index channel positive, RS-422 input
	5	Pulse+	Pulse positive, RS-422 output
	6	Gnd	Supply Ground
	7	ChA-	Channel A negative, Cosine 1Vpp
	8	ChB-	Channel B negative, Sine 1Vpp
	9	ChZ-	Index channel negative, RS-422 input
	10	Pulse-	Pulse negative, RS-422 output
	11	Encln0	TTL Level Input No. 0 (max 5VDC Input)
	12	Encln1	TTL Level Input No. 1 (max 5VDC Input)
	13	Encln2	TTL Level Input No. 2 (max 5VDC Input)
	14	PulseOut0	Pulse output, TTL single-ended 3.3V
	15	Gnd	Signal Ground

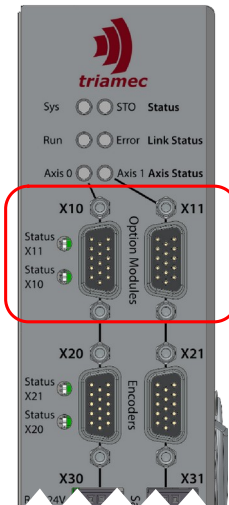


Figure 2: Option Modules jack (X10,X11)

4 TAM Registers

The *Software Option PU* introduces new *TAM Registers* to control the *Pulsing Unit* functionality.

These registers appear in the `Axes[]` node, where the corresponding *Option Module EN* or *EH* is installed. If *Option Module* encoders are installed on both axes, the *Pulsing Unit* will be available on both modules.

4.1 Output

Generated pulses can be routed to either a differential output, or a single ended output (see chapter 3). The corresponding output is configured with the following register. This register can also be used to change the polarity of the desired output and thus also control the static state.

`Axes[].Commands.OptionModule.PU_Output`

Value	Description
Disabled	The outputs are high impedance
RS422	Output as RS-422
RS422inverted	Output as inverted RS-422
TTL	Output as TTL 3V3
TTLinverted	Output as inverted TTL 3V3

4.2 Source

The position used for the pulse generator can be configured with the following register.

`Axes[].Commands.OptionModule.PU_Source`

The sources have different delay times, introduced by different signal paths with filters (Figure 3):

Value	Description	Delay
EncoderFast	Encoder signals with the fast filter.	5.2us*
EncoderSlow	Encoder signals as used by the position controller.	49.3us*
PathplannerAx0	Path planner position of axis 0.	49.3us*
PathplannerAx1	Path planner position of axis 1.	49.3us*

* Delays are perfectly repeatable. More precise values will be given here as more precise measurements of the actual delays become available.

It is recommended to start with `EncoderFast` and try other modes if the results are unsatisfactory. `EncoderSlow` and `Pathplanner` might provide an advantage at closely spaced pulses (e.g. 1nm) at very low speeds (e.g. 1 mm/s).

The delay may be compensated by adding it to the value in `PU_DelayTime`, see Chapter 4.4 Pulse Configuration. `PU_DelayTime` is used to calculate the correct firing time depending on current speed and acceleration, so that the actual position of the pulse is the desired one. If the speed changes during pulsing it is important to correctly configure `PU_DelayTime`, even if only the relative distance of the pulses matters.

If `PathplannerAx0` or `PathplannerAx1` is used as the source for the pulse generator, the parameter `Encoder[].Pitch` of the option module encoder must be configured, even if `Encoder[].Type` is set to `None`.

We recommend $Encoder[()].Pitch > \frac{\sqrt{move\ range}}{5'000'000}$ as the value for Pitch in this use case.

Note The quality of the pulsing unit is strongly related to the quality of the encoder signals. Noisy and nonlinear encoder signals will directly affect the pulse firing position. Therefore the encoder input provides a slight signal filtering in order to reduce the effects of noise. Further the encoder auto calibration removes offset and gain errors from the sin/cos signals, but only if they are smoothly changing.

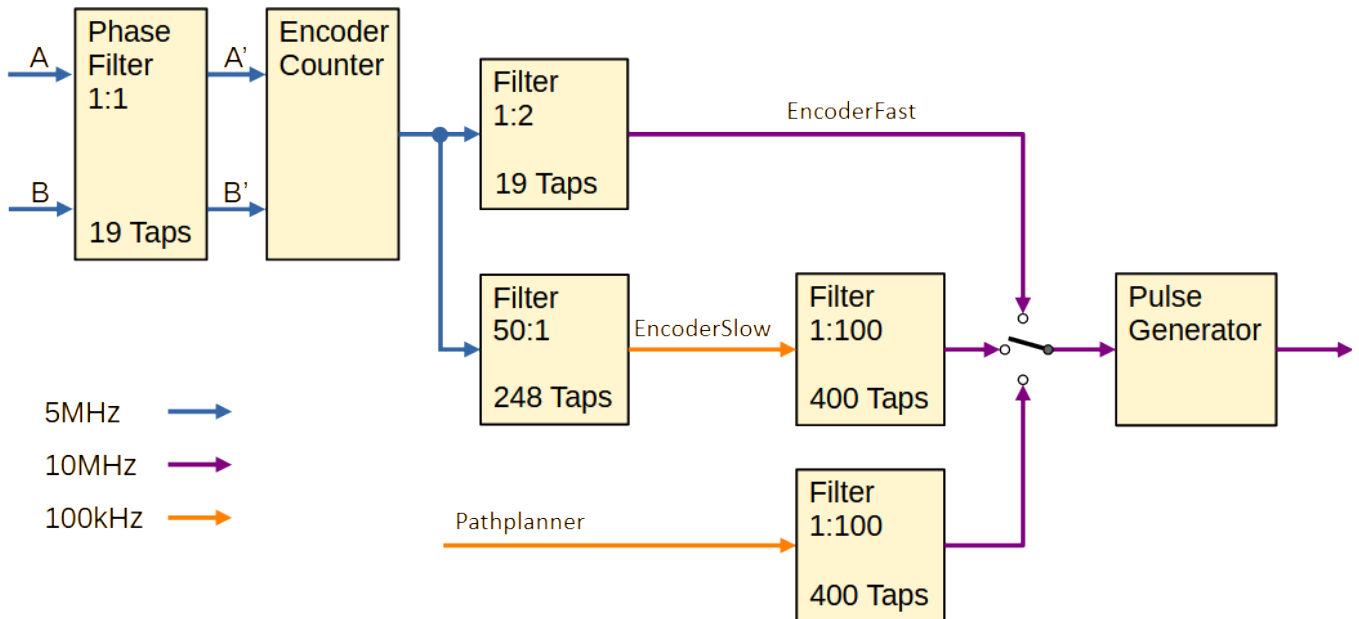


Figure 3: Signal path of the pulse position source

4.3 Mode

The Pulsing Unit can run in different modes. The main difference between modes is how the pulse configuration is applied.

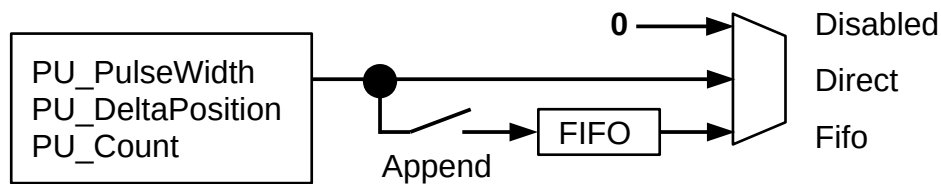


Figure 4: Visualization of Pulsing Unit modes

Axes[()].Commands.OptionModule.PU_Mode

Value	Description
Disabled	Disable the pulse generator.
Direct	Allows instant change of the pulse settings.
Fifo	Run sequences as stored in the FIFO.

The two following register values are written once, when the a mode is activated. While the mode is active, all other configuration changes refer to these values and changes in these two registers have no effect.

- `PU_ReferencePosition` is the absolute position of the first pulse and thus the start of a pattern. Each subsequent pulse position is calculated internally by adding the `PU_DeltaPosition` to the last fired pulse position. See also chapter 4.4.
- `PU_ActualPulseCount` is reset when a mode is disabled. Accordingly, this value is 0 when activating a mode. From this point in time the value in `PU_Count` must be set larger than `PU_ActualPulseCount` to generate new pulses. See also chapter 4.5.

4.3.1 Direct

Use this mode to change the pulse configuration at any time (see Figure 4). Changing a register described in chapter 4.4 takes effect with each 10kHz cycle of the drive. There are two ways of usage:

- `PU_Count = 0`
When activating the `PU_Mode = Direct` with `PU_Count = 0`, pulses will fire as long as the mode is active. The pulses start with crossing the `PU_ReferencePosition` once, and fire each time the `PU_DeltaPosition` is traveled. This is convenient if the pulse count is unknown or not calculated in advance.
A typical use case is pulsing a spiral, where the pulse distance is recalculated for each 10kHz cycle.
- `PU_Count > 0`
When activating the `PU_Mode = Direct` with `PU_Count > 0`, the configured amount of pulses fire while traveling across the positions. Further pulses only fire if the `PU_Count` is increased.
Typical use cases are simple pulse rows, without gaps and equidistant pulse positions.

4.3.2 Fifo

The *Fifo* mode allows to buffer pulse sequences, where one entry represents an equidistant set of pulses as configured per chapter 4.4. Only one FIFO entry can be appended per 10kHz cycle. The register `PU_FIFO` is used for FIFO commands.

This mode is configured as follows:

- Configure a pulse sequence with the registers described in 4.4. Add up `PU_Count` for each sequence.
- Push the sequence to the FIFO, by setting `PU_FIFO = Append`.
- Repeat the above steps to join different sequences, i.e. to create a heterogeneous pulse pattern.

The next sequence will move into the active state as soon as the previous sequence has been fired. The previous sequence is deleted.

Note The FIFO has a size of 512 entries for sequences. The currently active sequence doesn't occupy a place in the FIFO. The amount of pulses within one sequence is not limited.

4.4 Pulse Configuration

The following registers in `Axes[].Commands.OptionModule`, define the pulses, as visualized in Figure 5. Also refer to chapter 5, for application examples.

Register	Units	Resolution	Update Rate	Description
PU_PulseWidth	seconds	10ns	10kHz	Desired ON-time of the pulse.
PU_DeltaPosition	axis units	64bit float	10kHz	Distance between pulses. The sign indicates the crossing direction.
PU_Count	-	32bit int	10kHz	The accumulated number of pulses to fire.
PU_ReferencePosition	axis units	64bit float	at mode activation	Position of the first pulse when a mode is activated.
PU_DelayTime	seconds	10ns	at mode activation	Shift the pulse to compensate propagation delays. Fire early with negative values, or late with positive values. The delay is used to calculate the correct firing time depending on speed, acceleration and the desired position at the time of the pulse, see also Chapter 4.2 Source.

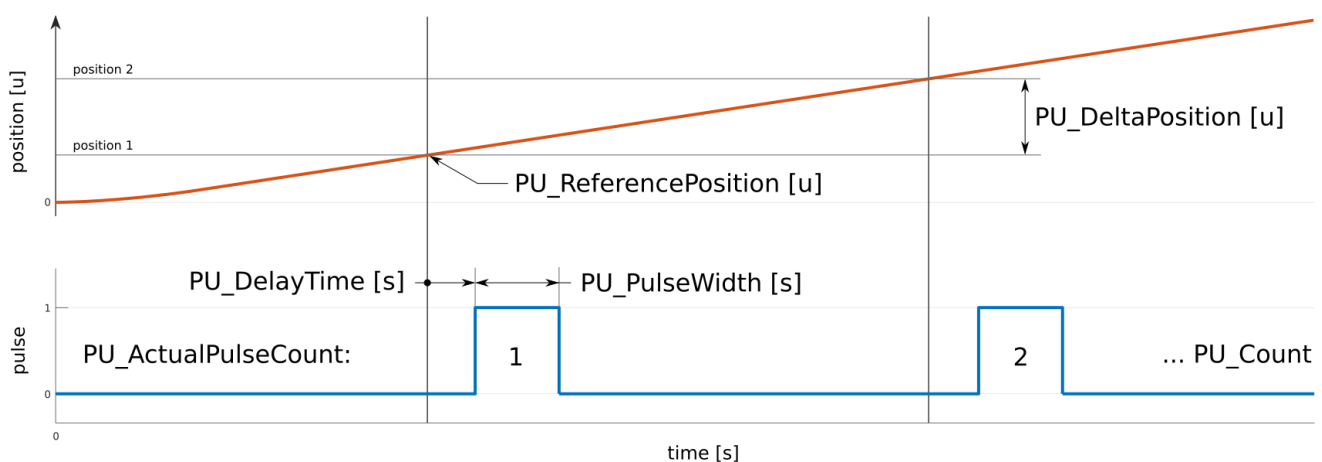


Figure 5: Visualization of Pulsing Unit parameters

Note The `PU_ReferencePosition` is only updated when the *Mode* is activated. To configure a sequence with a new reference position, set `PU_Mode = Disabled`, then configure the new sequence and lastly activate the *Mode*.

4.5 Signals

The following signals are available in `Axes[].Signals.OptionModule` and useful for debugging and programming.

- `PU_ActualPulseCount` indicates the last fired pulse index. If this value does not show the correct amount of pulses, it is usually an indicator for faulty configurations.
- `PU_FreeFifoEntries` indicates the currently available configuration slots. The first sequence pushed

into the FIFO is not reflected by this signal, as it is moved instantly into the active slot. While the value is 0, the FIFO is full and configurations are discarded. The following scenarios can lead to a full FIFO.

- ♦ A pulse configuration with $PU_Count \leq PU_ActualPulseCount$ has been pushed to the FIFO.
- ♦ A pulse configuration has been pushed to the FIFO with a $PU_Count \leq$ a previously pushed configuration. See also chapters 4.3 and 4.4.
- ♦ Continuously pushing to the FIFO, without actually consuming the pulse sequences.
 - The axis is not moving.
 - The axis is moving in the wrong direction, or
 - The pulses are configured for the wrong direction (negative $PU_DeltaPosition$).

Note The signals are delayed by 0.2ms (two cycles at 10kHz). In case of pushing sequences with *Tama*, consider to stop with a safety margin on $PU_FreeFifoEntries$, or count the entries with an own counter variable. Same applies to checks against $PU_ActualPulseCount$.

Using the *TAM System Explorer Scope*, the pulse output can be recorded with the register `Axes[].Signals.-General.DigitalInputBits.OptionEncln3`, if PU_Output is `TTL` or `TTLinverted` and the pulse duration $PU_PulseWidth$ is larger than 20us.

5 Examples

Applications involving a *Pulsing Unit* are very flexible and therefore it is most probably commanded from *Tama* programs. To get started, *Triamec* provides examples on [GitHub](#).

Glossary

FIFO A FIFO is a data structure where the first item added is the first item to be removed.

References

- [1] "Option Modules Manual ", HWTO_OptionModulesManual_EP019.pdf, Triamec Motion AG, 2023.
- [2] "Software Options Overview", SWTO_SoftwareOptions_EP002.pdf, Triamec Motion AG, 2023.

Revision History

Version	Date	Editor	Comment
001	2023-04-27	sm	initial version
002	2023-05-12	Bl, lk	Clarification on pulse frequency and compensation of delays with PU_DelayTime
003	2023-09-27	sm	Add signals delay info, move example to GitHub
004	2024-02-21	ab	Added hints for the use case PathPlanner, Additional debug functionality added

Triamec 中国-销售/技术服务中心

TEL: (0) 13776269231 Email: sales@tyt-motion.com URL: www.triamec.com.cn

Copyright © 2024
Triamec Motion AG
All rights reserved.

Triamec Motion AG
Lindenstrasse 16
6340 Baar / Switzerland

Phone +41 41 747 4040
Email info@triamec.com
Web www.triamec.com

Disclaimer

This document is delivered subject to the following conditions and restrictions:

- This document contains proprietary information belonging to Triamec Motion AG. Such information is supplied solely for the purpose of assisting users of Triamec products.
- The text and graphics included in this manual are for the purpose of illustration and reference only. The specifications on which they are based are subject to change without notice.
- Information in this document is subject to change without notice.