

# Pulsing Unit

## Application Note 152

This document describes the setup and usage of a *Pulsing Unit (PU)*. *PU* is a *Software Option* (see [2]). It enables the encoder *Option Modules EN* or *EH* to fire pulses with up to 10 MHz or output a position signal 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	4.6	Signals.....	9
2	Preconditions.....	3	4.7	Remarks.....	10
3	Pinout.....	3	5	Examples.....	11
4	TAM Registers.....	4	5.1	Tama-based FIFO mode.....	11
4.1	Source.....	4	5.2	Pulsing along a multi-dimensional path .....	11
4.2	Output.....	5		Glossary.....	14
4.3	Mode.....	6		References.....	14
4.4	Pulse Configuration for Pulsing.....	8		Revision History.....	15
4.5	Pulse Configuration for Position Output .....	9			

# 1 Purpose and Usage

A Triamec Pulsing Unit has two main use cases. For one, it can control an external device by triggering a process with a digital electrical signal. The triggers are fired with reference to a position information of the motion system at a firing rate of up to 10MHz. The second use is to output a TTL- or RS422-encoder signal to communicate a position information to an external measurement system.

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 (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 (pulsing)	The Pulsing Unit output
Pulse (position output)	A Pulsing Unit event, corresponding to a change of one of the phase signals
Pitch	The distance between two pulses, corresponding to a quarter of the position-output signal period.

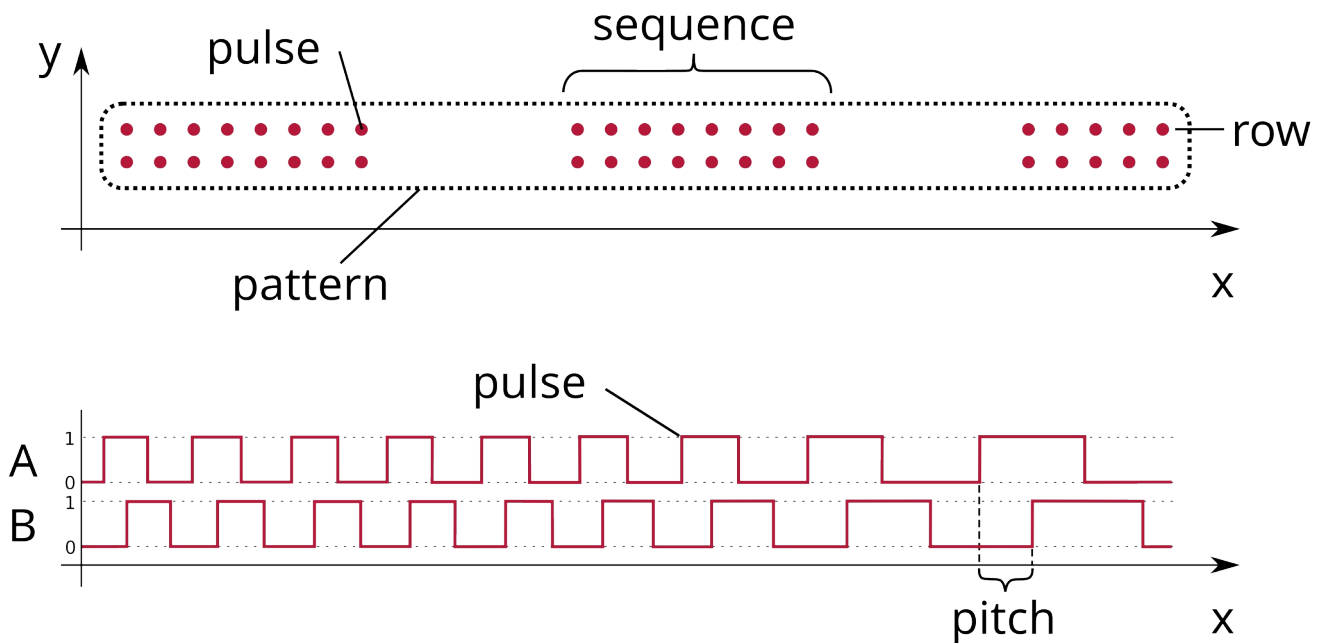


Figure 1: Visualization of Pulsing Unit specific terms for the pulsing use (top) and the position output mode (bottom). A pulsing pattern consists of pulse sequences that themselves consist of equidistant pulses, fired with respect to one position signal (here, x). The position-output signal is made up of two signals A and B that imitate an incremental encoder signal. The pitch of the position output is defined as the distance between two pulse events and is therefore a quarter of the pitch of an equivalent encoder signal.

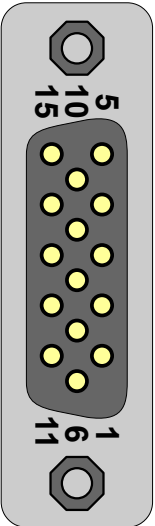
## 2 Preconditions

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. Exception: PU\_Source = PathPlanner.
- Correct encoder settings must be configured, also in case of PU\_Source = PathPlanner.
- Incompatible with EncoderTopology = Standard.
- The command SetPosition should not be used while the Pulsing Unit is active.

## 3 Pinout

The encoder pinout with enabled *Software Option PU* is based on the analog encoder interface. Pins, names and signals in bold signify the different outputs that can be used by the Pulsing Unit. **Pulse (PhX)** in the name of a TTL or RS422 channel identify outputs for the pulsing (position output) modes with the corresponding output types.

Pin Layout X10/X11	Pin	Name	Signals
 <p>15-pin female D-Sub socket</p>	1	+5VDC	Encoder Supply
	2	ChA+	Channel A positive, Cosine 1Vpp
	3	ChB+	Channel B positive, Sine 1Vpp
	4	ChZ+/ <b>PhA+</b>	Index channel positive, RS-422 input / <b>[PosOut] Phase A, RS-422 output</b>
	5	<b>Pulse+/ PhB+</b>	<b>[Pulse] Pulse positive, RS-422 output / [PosOut] Phase B, RS-422 output</b>
	6	Gnd	Supply Ground
	7	ChA-	Channel A negative, Cosine 1Vpp
	8	ChB-	Channel B negative, Sine 1Vpp
	9	ChZ-/ <b>PhA-</b>	Index channel negative, RS-422 input / <b>[PosOut] Phase A-, RS-422 output</b>
	10	<b>Pulse-/ PhB-</b>	<b>[Pulse] Pulse negative, RS-422 output / [PosOut] Phase B-, RS-422 output</b>
	11	EnIn0 <b>PhA-</b>	TTL Level Input No. 0 (max 5VDC Input) / <b>[PosOut] Phase A-, TTL single-ended 3.3V</b>
	12	EnIn1/ <b>PhB-</b>	TTL Level Input No. 1 (max 5VDC Input) / <b>[PosOut] Phase B-, TTL single-ended 3.3V</b>
	13	EnIn2/ <b>PhA+</b>	TTL Level Input No. 2 (max 5VDC Input) / <b>[PosOut] Phase A, TTL single-ended 3.3V</b>
	14	EnIn3/ <b>PulseOut/ PhB+</b>	TTL Level Input No. 3 (max 5VDC Input) / <b>[Pulse] PulseOut, TTL single-ended 3.3V / [PosOut] Phase B, TTL single-ended 3.3V</b>
	15	Gnd	Signal Ground

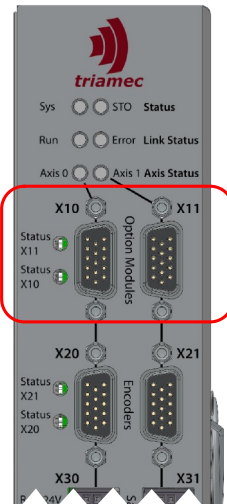


Figure 2: Option Modules jack (X10, X11)

## 4 TAM Registers

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

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

The registers should ideally be set in the order they are listed in the next subsections. First the `PU_Source` and `PU_Output`, then the pulse configuration and the `PU_Mode`. If the *Pulsing Unit* is already set up and a new setup requires a change of the `PU_Source`, set `PU_Output` to `Disabled` before changing the `PU_Source` and then set the `PU_Output` again to correctly reset the Pulsing Unit.

### 4.1 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 (see figure 3):

Value	Description	Delay EN*	Delay EH*
EncoderFast	Encoder signals with the fast filter.	5.2 μs	7.6 μs
EncoderSlow	Encoder signals as used by the position controller.	49.3 μs	51.7 μs
PathplannerAx0	Path planner position of axis 0.	49.3 μs	51.7 μs
PathplannerAx1	Path planner position of axis 1.	49.3 μs	51.7 μs
EncoderFastCompensated	Encoder signals with the fast filter and corrected by <code>AxisCompensation</code>	5.2 μs	7.6 μs
EncoderSlowCompensated	Encoder signals as used by the position controller and corrected by <code>AxisCompensation</code>	49.3 μs	51.7 μs

\* 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` >  $\frac{\sqrt{\text{move range}}}{5'000'000}$  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.

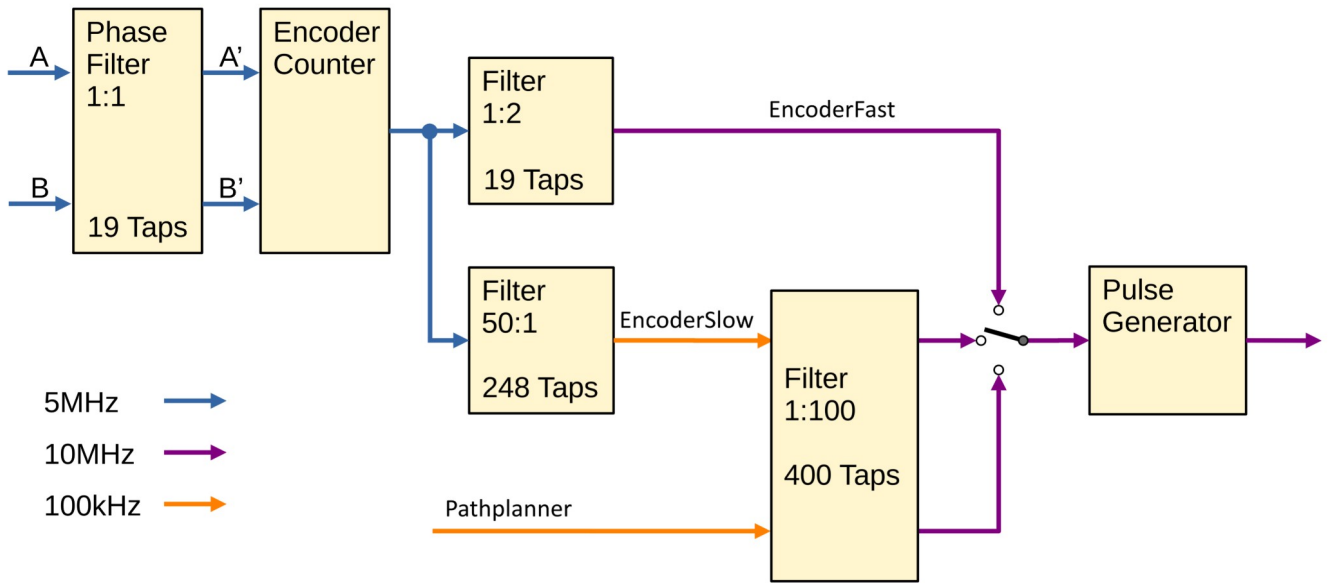


Figure 3: Signal path of the pulse position source

## 4.2 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 (or inverted counting direction for Position Output)
TTL	Output as TTL 3V3
TTLinverted	Output as inverted TTL 3V3 (or inverted counting direction for Position Output)
TTL4 (Position Output)	Output with four TTL 3V3 (phases A/B and inverted phases A/B)
TTL4inverted (Position Output)	Output with four TTL 3V3 (phases A/B and inverted phases A/B) with inverted counting direction.

## 4.3 Mode

The Pulsing Unit can run in different modes. The discretely triggered pulses can be realised either with the Direct or Fifo mode, as shown in figure 4. The main difference between the two modes is how the pulse configuration is applied. With the mode PositionOutput, the digital channels are controlled in such a way that they output an encoder-like position signal. The configuration for Direct and Fifo is shown in more detail in chapter 4.4, the configuration for PositionOutput in chapter 4.5.

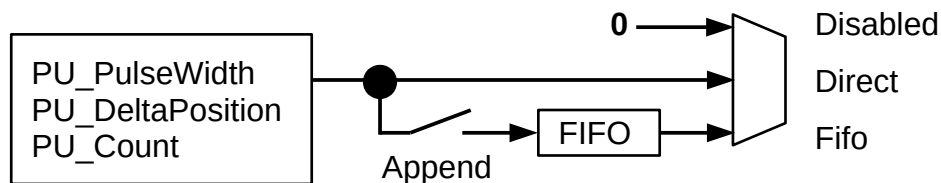


Figure 4: Visualization of Pulsing Unit modes

`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.
PositionOutput	Generates incremental encoder signals with phases A/B or A+/B+/A-/B-

The two following register value is written once, when 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 in modes `Direct` and `Fifo`. Each subsequent pulse position is calculated internally by adding the `PU_DeltaPosition` to the last fired pulse position. See also chapter 4.4. For `PositionOutput`, this parameter is ignored since the reference position for this point is the position at mode activation.

The following signal is reset and set to zero when a mode is activated.

- `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 4.6.

### 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.

The mode `Direct` is not compatible with outputs `TTL4` and `TTL4inverted`.

## 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 chapter 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.

The mode `Fifo` is not compatible with outputs `TTL4` and `TTL4inverted`.

**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.

## PositionOutput

The `PositionOutput` mode works fundamentally differently from the regular pulsing modes. Instead of discrete pulsing events, this mode outputs a continuous encoder-like signal which can be fed to external measurement devices that rely on position information. The supported encoder-signal types are defined by the `PU_Output`. These are single-ended TTL (phase A/B), differential TTL4 (phase A+/B+/A-/B-) and RS422 (phase A+/B+/A-/B-). The corresponding outputs with suffix `*inverted` invert the encoder counting direction and not the individual signals. The configuration of this mode is explained in chapter 4.5.

## 4.4 Pulse Configuration for Pulsing

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	10kHz	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 4.1 Source.

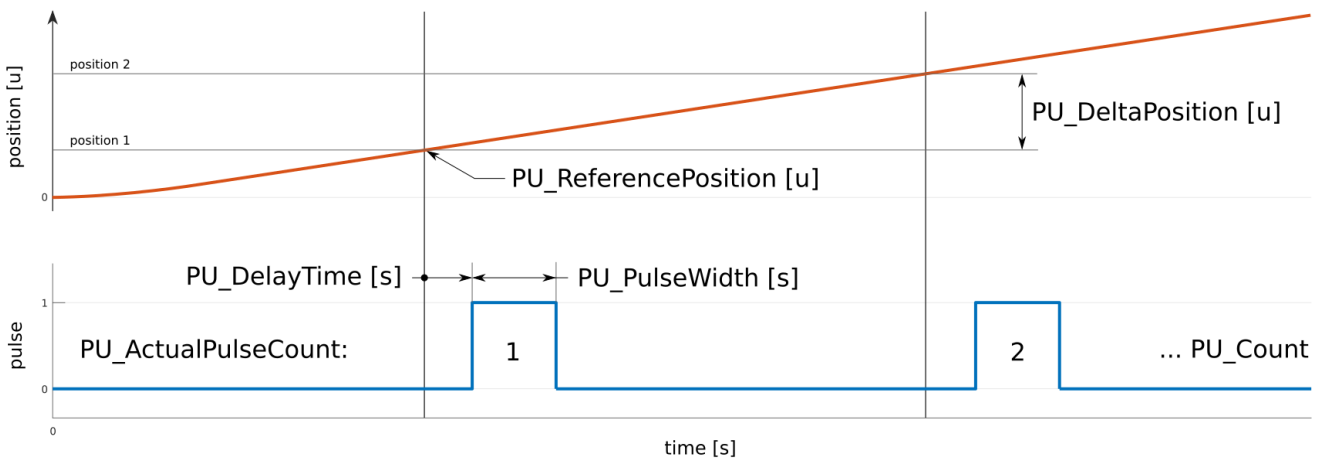


Figure 5: Visualization of Pulsing Unit parameters

**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 Pulse Configuration for Position Output

The following registers in `Axes[].Commands.OptionModule`, define the position-output signal, as visualized in figure 6.

Register	Units	Resolution	Update Rate	Description
PU_PulseWidth	seconds	10ns	10kHz	Minimal time between two pulsing events (switching of a signal channel) and therefore minimal time within a signal quadrant. The actual resolution is the next higher multiple of 100ns. This parameter is important if the position is at standstill near a pulsing position to reduce uncontrolled jitter of the position output. It can be set to zero if the following electronics can handle signal changes faster than 10 MHz.
PU_DeltaPosition	axis units	64bit float	10kHz	Distance between pulses. Only positive values are allowed. Corresponds to a quarter of the resulting encoder-signal pitch.
PU_Count	-	-	-	Ignored in this mode
PU_ReferencePosition	-	-	-	Ignored in this mode
PU_DelayTime	seconds	10ns	10kHz	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 4.1.

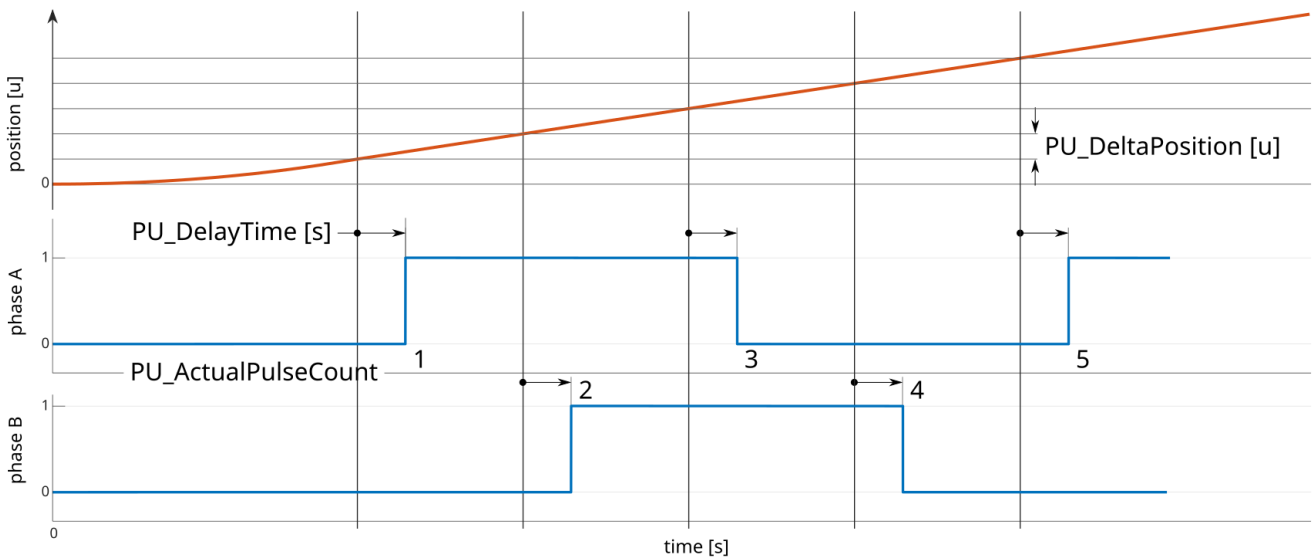


Figure 6: Visualization of Position Output parameters

## 4.6 Signals

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

ming.

- `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. This value also counts pulsing events for the mode `PositionOutput` but can increase uncontrollably fast if the position jitters around a pulsing event.
- `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 for all TTL-output modes can be recorded with the register `Axes[].Signals.General.DigitalInputBits.OptionEncln0-3`.

- When using `PU_Mode = Direct` or `Fifo` with `PU_Output = TTL` or `TTLinverted`, the register `OptionEncln3` shows the output, provided the pulse duration `PU_PulseWidth` is larger than 20  $\mu$ s.
- When using the `PU_Mode = PositionOutput` with `PU_Output = TTL` or `TTLinverted`, the registers `OptionEncln2,3` show the current output.
- When using the `PU_Mode = PositionOutput` with `PU_Output = TTL4` or `TTL4inverted`, the registers `OptionEncln0-3` show the current output.

Check the pin-out in chapter 3 to see which pins correspond to which signal.

## 4.7 Remarks

When activating or deactivating a pulsing unit mode, the following behavior applies:

- When activating `PU_Mode = Direct` or `Fifo` with any `PU_Output = *inverted`, the corresponding outputs switch to high.
- When deactivating `PU_Mode = Direct` or `Fifo` with any `PU_Output = *inverted`, the corresponding outputs remain high.
- When activating `PU_Mode = PositionOutput`, the phases are initialized as  $phA+ = 0$  and  $phB+ = 0$  (and consequently  $phA- = 1$  and  $phB- = 1$  if applicable).

## 5 Examples

This manual covers mostly the fundamentals of configuring the pulsing unit. However, many applications go beyond simple equidistant pulsing in one dimension. For any complex application, Tama programs provide the perfect instance to configure, control and adapt the pulsing unit with maximum flexibility. Tama programs are executed by the drive in every 10 kHz cycle of the firmware and therefore allow fast and deterministic control over the pulsing unit. An introduction to Tama programs is given in the Tama User Guide [3].

In this section, we look at two examples that demonstrate how such complicated applications can be satisfied with the *Pulsing Unit*. The first example covers the automated filling of the FIFO using a Tama program. Since the FIFO can accept a new entry every 10 kHz cycle, Tama is the optimal tool to fill the FIFO and oversee its application. The second example explains how we can pulse along a complex path. This is an important skill for applications where the *Pulsing Unit* triggers cutting lasers. These often rely on equidistant pulses on a complex path in two or more dimensions.

### 5.1 Tama-based FIFO mode

The applications for the pulsing unit can be very dynamic such that an acyclic register access from the Tam API is not sufficiently fast. Also, to use the FIFO to its full extent with a new entry in every 10 kHz cycle, it needs to be commanded from a *Tama* program.

To get started with this topic, *Triamec* provides examples on [GitHub](#) called *PulsingUnit-Barcode*. It describes how the FIFO of the pulsing unit can be filled cyclically with new entries. The information about the individual entries can be either calculated or defined in the Tama itself, loaded to the drive in the form of a table or be written to the Tama application registers from a PLC or TamAPI application. Please visit the GitHub example for further documentation or contact Triamec Motion AG for more information.

### 5.2 Pulsing along a multi-dimensional path

In material processing with lasers, it is often crucial to pulse not along a single dimension, but along a complex path in two or more dimensions. Usually, this path corresponds to the distance traveled by the tool-center point, which needs to be determined from the movement of the involved axes. Figure 7 shows a schematic tool path with pulsing positions and a graph showing the corresponding path. To achieve this, we can use the `PU_Source = PathPlanner` and stream an arbitrarily complex path to the path planner of an axis. Since no motor is being commanded with this position data, it is often necessary to dedicate a drive axis fully to the pulsing unit for this application.

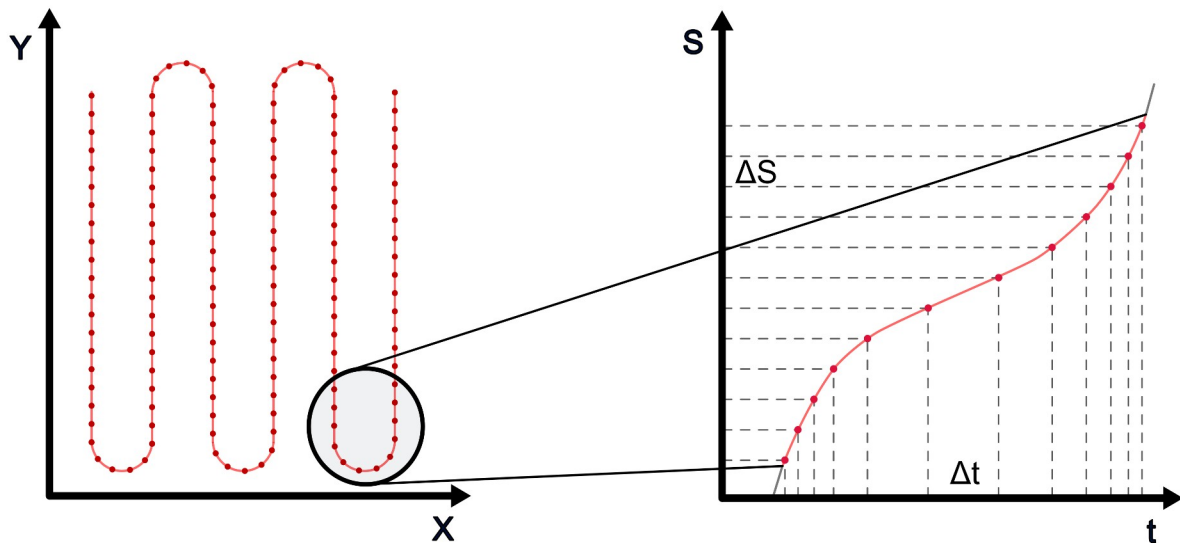


Figure 7: Schematic visualization of a tool path in two dimensions with equidistant pulsing positions. The right graph shows the corresponding path of a section as a function of time. To retain constant distance between pulses, the time between pulses needs to increase in the turning points to adapt for the slower velocity.

For this example, we assume that we want to follow the path traveled by the center of an XY stage. This is the common case for laser cutting, where an equal energy deposition in the material is crucial for an optimal cutting result. We will look at two possible implementations:

1. Controlling the path from the PLC/CNC control
2. Streaming a path with Trajectory Streaming
3. Calculating the path in Tama

Both have the common goal to have the traveled path of the stage as the position information of the path planner and hence as the input to the pulsing unit. Along this path, all the settings of the pulsing unit discussed in this document can be used, such as *PU\_Mode = Direct, FIFO or PositionOutput*.

## Controlling the path from the CNC

The first option is to calculate the path directly in the PLC. In the case of a Beckhoff TwinCAT application, there is a number of possible solutions. For NC applications, the feed can be calculated in the PLC from the current speeds of the underlying axes. The speeds can be read out in the PLC and used to calculate the path speed, as seen in the following code example:

```
// Calculate the current feed with the PLC
// G1, G2 and pathAxis are elements of type AXIS_REF and are linked to the respective NC axes.

// 1. Read out the individual axis velocity and acceleration
velX      := G1.NcToPlc.SetVelo;
accX      := G1.NcToPlc.SetAcc;
velY      := G2.NcToPlc.SetVelo;
accY      := G2.NcToPlc.SetAcc;
pathVelocity := SQRT(velX * velX + velY * velY);

// 2. Read out the PLC cycle time
nCycleTimeNS := TcGetTaskCycleTime();           // cycle time in ns (integer)
fCycleTimeS   := LREAL(nCycleTimeNS) / 1000000000.0; // cycle time in s (float)
```

```
// 3. Calculate X,A for the combined path
pathPosition := PathPosition + PathVelocity * fCycleTimeS;
pathAcceleration := (velX * accX + velY * accY) / pathVelocity;

// 4. Determine the direction
IF pathVelocity = 0 AND pathAcceleration = 0 THEN
  pathDirection := 0; (* stand still *)
ELSIF pathVelocity >= 0 THEN
  pathDirection := 1; (* positive motion *)
ELSE
  pathDirection := -1; (* negative motion *)
END_IF

// 5. Feed the path to an NC axis.
MC_ExtSetPointGenFeed(
  Position := pathPosition,
  Velocity := pathVelocity,
  Acceleraiton := pathAcceleration,
  Direction := pathDirection,
  Axis := pathAxis);
```

A similar approach is chosen with a CNC application. For such applications, the path feed is available and can be read out from the axis group as

```
pathVelocity = gpCh[channel_idx]^bahn_state.active_feed_r; // given in µm/s
```

The position and acceleration can then be calculated by numerical integration and differentiation, respectively and fed to the NC axis.

For specific implementation options and examples, please contact Triamec Motion AG.

### Streaming the path with Trajectory Streaming

In certain applications, the path is not controlled by a real-time NC or CNC but calculated offline beforehand. This can be beneficial if the density of commanded positions needs to be increased above the possible field-bus-cycle time. Also, it may be useful if a process is repeated many times. In such cases, it is an easy feat to include the path length as an additional dimension of data and feed it to an axis. With Triamec drives, feeding such precalculated data is best done with the Trajectory Streaming (TraSt) feature. It allows the streaming of data with point densities up to 50'000/s via Ethernet to the individual drives, where it is stored in a FIFO table. The path is executed synchronously between all drives as long as they are connected by a field bus.

The process of reading a data file, feeding the data to the drives and executing the movement is controlled with a C# program. Example programs are available upon request from Triamec Motion AG. Please note that TraSt requires an additional software option (TS) additional to the pulsing unit (PU) [2].

### Calculating the path in Tama

Analogously to the PLC example above, the total path traversed can also be calculated in a Tama program. In contrast to the last two sections, where the path length was calculated from the commanded

position, here we can also calculate the path length from the actual encoder feedback. This can be beneficial if the following error of the axes is larger than the required precision. It is also useful if the axis is controlled by point-to-point commands, where the whole path between points is not strictly defined.

The main ingredient for this solution is that at least the position for the involved axes, either commanded or actual, are available as registers on the drive with the pulsing unit. In the case of commanded data, the accuracy of the calculation is improved if the velocity and acceleration are also provided. For this application, we are interested in the information of multiple axes, while at most two axes are available on a single drive. Therefore, we rely on the information being provided either by the machine control or the other drives. Due to its fast cycle time and drive-to-drive communication, Tria-Link is highly recommended for this application. It allows the communication of the actual axis position with only 100  $\mu$ s delay. An implementation with EtherCAT is also possible, but the information will have considerably more delay.

A Tama example to calculate the path distance from two other axes on another drive can be requested from Triamec Motion AG. If the Tama is to be used in combination with a control system, a few precautions need to be taken. For example, the pulsing axis must not be part of the axis group since the Tama feeds the commanded positions to the axis. Further, for proper system integration, axis errors need to be properly communicated between pulsing-unit axis and the axis group with the real axes. Faulty behavior could lead to prolonged pulsing in standstill or pulsing at wrong positions.



## 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.
- [3] “Tama user Guide”, SWTAMA\_UserGuide\_EP004.pdf, Triamec Motion AG, 2025.

## 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
005	2025-07-25	yz, yb	Added information for mode PositionOutput and output TTL4(inverted)
006	2026-06-02	yz	Expanded section Examples

---

Copyright © 2026  
Triamec Motion AG  
All rights reserved.

Triamec Motion AG  
Lindenstrasse 16  
6340 Baar / Switzerland

Phone +41 41 747 4040  
Email [info@triamec.com](mailto:info@triamec.com)  
Web [www.triamec.com](http://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.