Trajectory Streaming using Tama

Application Note 158

This application note describes how to use Trajectory Streaming to feed an up to 50kHz trajectory from a Tama application.

Table of Contents

1 Introduction....2

2 Simple Trajectory Streaming....2
- 2.1 Prepare Trajectory Streaming....2
- 2.2 Write the position table from Tama....2
- 2.3 The fit polynomial....3
- 2.4 Starting the stream....4
- 2.5 Ending a stream....4

3 Extensions....5
- 3.1 Current FeedForward....5
- 3.2 Custom extensions....5

References....5

Revision History....5

Document	AN158_TrajectoryStreamingTama_EP
Version	002, 2026-07-30
Source	Q:\docRL19\ApplicationNotes\
Destination	T:\Doc\ApplicationNotes
Owner	mvx

1 Introduction

This application note describes how to use Trajectory Streaming to feed an up to 50kHz path trajectory from a Tama application. This feature is available with firmware 4.24 or higher and requires the software feature “TS” to be ordered from Triamec.

2 Simple Trajectory Streaming

Trajectory Streaming allows to feed a table of positions to the 100kHz controller at a desired rate between 0 and 50000 rows/s. Feed rates between 0 and 50000 rows/s are supported. This chapter describes the most simple situation of streaming positions at a constant feed rate.

2.1 Prepare Trajectory Streaming

Data is written into a circular table with a size between 5000 and 1 million points ([1]), which is used like a FIFO. We will write junks of positions and set TableHead to let the drive know, which rows we wrote. The drive will then consume theses points and move its TableTail to indicate, which positions were already consumed.

In the first step, we set the basic properties of the stream. Since we do not intend to end the table so far, we set the end to -1:

```
Register.Axes_0.Commands.PathPlanner.Stream.TableSize = 1000000;
Register.Axes_0.Commands.PathPlanner.Stream.TableEnd = -1;
```

As ExtensionMode is not used for this simple example, we set RowSize to zero (see chapter 3 for more information).

```
Register.Axes_0.Commands.PathPlanner.Stream.ExtensionMode.RowSize = 0;
```

In an error situation, the feature is stopped. The error reaction can be configured using the enumerator:

```
Register.Axes_0.Commands.General.Emergency.TrajectoryStreamError.Reaction
```

to one of the values of ErrorReaction

- None
- Stop — Stop with the standard ramp
- EmergencyStop — Stop with the emergency ramp
- MoveToSafePosition — Move to the safe position specified at Register.Axes_0.Commands.General.Emergency.SafePosition

Now we are ready for the next step.

2.2 Write the position table from Tama

Position data are written into a circular table which can be filled using index `row`=[0 ... TableSize-1]. Start at index `row`=0 and use the following code to write one position

```
Register.Axes_0.Commands.PathPlanner.Stream.Positions[row] = position;
```

1 There is a minimum TableSize of 5000. However, this only defines the wrapping size, not the amount of points sent. The amount of points sent can be smaller or larger than TableSize.

To minimize Tama load we fill a maximum of 6 rows per 100µs cycle of the isochronous Tama. More rows are possible but will increase the execution time of the Tama program. It is also possible to use asynchronous Tama for this task. After filling a junk of positions, set the last written table row as head:

```
Register.Axes_0.Commands.PathPlanner.Stream.TableHead = row;
```

Since the table is circular we must wrap with TableSize and we must ensure, we do not overwrite data that has not been retrieved yet. The signal TableTail shows the last retrieved row, which must not be reached by the TableHead.

The following sample code can be executed cyclically from isochronous Tama and will prepare a cosine trajectory of frequency 1000Hz to be run at 50000 rows/s if

- Doubles[0] is initialized with 0.0
- PhaseAdvance = 2.0 * Math.PI * 1000.0/50000.0

```
static void CalculatePath() {
   int row = Register.Axes_0.Commands.PathPlanner.Stream.TableHead;
   for (int k = 0; k < 6; k++) {
          row += 1;
          if (row >= Register.Axes_0.Commands.PathPlanner.Stream.TableSize) {
              row = 0;
          }
          if (row != Register.Axes_0.Signals.PathPlanner.Stream.TableTail) {
              SavePosition(row);
              Register.Axes_0.Commands.PathPlanner.Stream.TableHead = row;
          } else {
              break;
          }
      }
}
static void SavePosition(int row) {
       var phase = Register.Application.Variables.Doubles[0];
       double pos = 1.0 - Math.Cos(phase);
       Register.Axes_0.Commands.PathPlanner.Stream.Positions[row] = pos;
       phase += PhaseAdvance;
       if (phase >= Math.PI * 2.0) {
          phase -= Math.PI * 2.0;
       }
       Register.Application.Variables.Doubles[0] = phase;
}
```

2.3 The fit polynomial

The Trajectory Streaming function allows to stream arbitrary stream rates of 0 to 50000 rows/s. The position controller on the other side runs at a fix rate of 100000 points/s. Therefore, the desired positions will be at any fractional “row” between the table rows written. This is enabled by fitting a 5^{th} order polynomial into a set of table positions. The streaming rate will then be used to calculate the desired position from the polynomial at the correct location.

To allow fitting the initial polynomial, fill at least 9 rows before proceeding to the next step.

2.4 Starting the stream

Make sure the axis is enabled and the AxisState is in Standstill and the commanded PathPlanner position is identical to the first position in the table.

The feed rate at wich the table is streamed is defined by the polynomial coefficients K0..K3. For this example a constant feed rate of 50000 rows per second is used starting at row 0. Therefore K0 is set to 0 and K1 to 50000.

```
Register.Axes_0.Commands.PathPlanner.Stream.FeedOverride.K0 = 0.0;
Register.Axes_0.Commands.PathPlanner.Stream.FeedOverride.K1 = 50000.0;
Register.Axes_0.Commands.PathPlanner.Stream.FeedOverride.K2 = 0.0;
Register.Axes_0.Commands.PathPlanner.Stream.FeedOverride.K3 = 0.0;
Register.Axes_0.Commands.PathPlanner.Stream.FeedOverride.Continuous = true;
```

Start the trajectory immediately by setting the timestamp trigger to TriaLinkTimestamp using

```
Register.Axes_0.Commands.PathPlanner.Stream.FeedOverride.Timestamp =
Register.General.Signals.TriaLinkTimestamp;
Register.Axes_0.Commands.PathPlanner.Command = TriaLink.PathPlannerCommand.MoveStreamCoupled;
```

The AxisState now changes to StreamCoupledMotion and the trajectory is streamed to the axis.

2.5 Ending a stream

The streamCoupledMotion state may end in various ways.

The first method is using standard PathPlanner commands such as Halt, Stop, MoveAbsolute, etc.

The second method is to mark the table as finished by setting

```
Register.Axes_0.Commands.PathPlanner.Stream.TableEnd = row;
```

The feature will then proceed until TableTail is close to TableEnd and then proceed with a PathPlanner Halt.

Please note that the trajectory might end some rows ahead of TableEnd because ending a trajectory can only take place at a 100µs boundary and feeding might consume up to 5 rows per 100µs at 50000 rows/s. For a feed rate > 10000 rows/s we propose to write the final position multiple times to make sure the position is exactly reached.

The third reason to finish this mode is the error situation: If the feature detects that the TableTail reaches the TableHead, it will throw an error and the error reaction will be started as defined in chapter 2.1.

3 Extensions

3.1 Current FeedForward

The Trajectory Streaming feature can also be used to supply FeedForward currents. Modify the steps in chapter 2 by setting

```
Register.Axes_0.Commands.PathPlanner.Stream.ExtensionMode.RowSize = 1;
Register.Axes_0.Commands.PathPlanner.Stream.ExtensionMode.UseCurrent = true;
Register.Axes_0.Commands.PathPlanner.Stream.ExtensionMode.UseVelocity = false;
```

and in the function `SavePosition(row)` add the code

```
float current = (float)Math.Cos(2.0*phase);
Register.Axes_0.Commands.PathPlanner.Stream.Extensions.Floats[row] = current;
```

The unit of the currents are **A/(row/s)²**. The currents are multiplied by the square of the actual row-rate in the Trajectory Streaming feature. With constant streaming, this is Stream.FeedOverride.K1. This takes into account that at constant row rate the FeedForward currents usually scale with the square of the row rate.
Similar to the case for position streaming, we should make sure that the initial value of the current is equal to the initial element in the table.

```
Register.Axes_0.Commands.PathPlanner.Stream.Extensions.Floats[tableHead] = startCurrent;
```

3.2 Custom extensions

Custom extensions allow data exchange between the streaming function on the pc-side and a Tama program. Based on section 3.1, choose a RowSize larger than 1. Choose a correct TableSize based on the following limitation

```
Stream.TableSize < 2097152 / Stream.ExtensionMode.RowSize;
```

If RowSize is 3, for example, the pc-side streaming feature will write currents and extensions to

```
Register.Axes_0.Commands.PathPlanner.Stream.Extensions[3*row] = current;
Register.Axes_0.Commands.PathPlanner.Stream.Extensions[3*row+1] = extension1;
Register.Axes_0.Commands.PathPlanner.Stream.Extensions[3*row+2] = extension2;
```

It is up to the user, how to use these extensions with Tama code. The TableTail might be a starting point for the index where real-time-data are located, because it indicates the last row consumed by the Trajectory Streaming feature.

Custom extensions are a beta-feature. Contact Triamec Motion AG for more advise.

References

[1] “Tama Real-Time Drive Programming”, SWTAMA_UserGuide_EP004.pdf, Triamec Motion AG, 2025

Revision History

Version	Date	Editor	Comment
001	2025-04-28	mvx, dg	Initial edit
002	2026-07-30	mvx	First Release

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
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.