Data Export and Import in TAM System Explorer

Application Note

The TAM System Explorer application ([1], [2]) is the primary service tool for setting up Triamec Motion drives. One primary component is the scope. Acquired data can be persisted by printing or saving an image of the scope to disc. This document focuses on yet a few other possibilities:

- Exporting data as comma-separated values to a text file (CSV, section 1),
- processing *device dump files* (`.TAMdmp`, section 2) and
- streaming (`.TAMpbf`, section 3).

All this data can be imported into the scope.

The data formats are defined separately in section 4.

Table of Contents

Document AN132_TamSystemExplorerDataExchange_EP
Version 007, 2023-08-28
Source Q:\doc\ApplicationNotes\
Destination T:\Doc\ApplicationNotes\
Owner chm

1 Working With Comma Separated Values

Sections 1.1 and 1.2 help in persistency and restoration of acquired data. For an informal specification of the text data format, see section 4.2.

1.1 Save Data

Using the menu **Scope | Save Plot Data…**, export all data to CSV.

A scope configuration companion file is always saved beneath the actual data file. The same name is given, with the extension changed to `.auto.TAMsco`. This companion file isn't necessary for long term storage. However, it supports a round-trip experience when loading the data back into the scope afterwards. Naturally, renaming and re-basing of the data file requires synchronization with the companion file.

Auto-Save

When repeat is enabled, you may opt-in to automatically save a CSV file each time the scope concludes a measurement. 1 shows how to enable the feature.

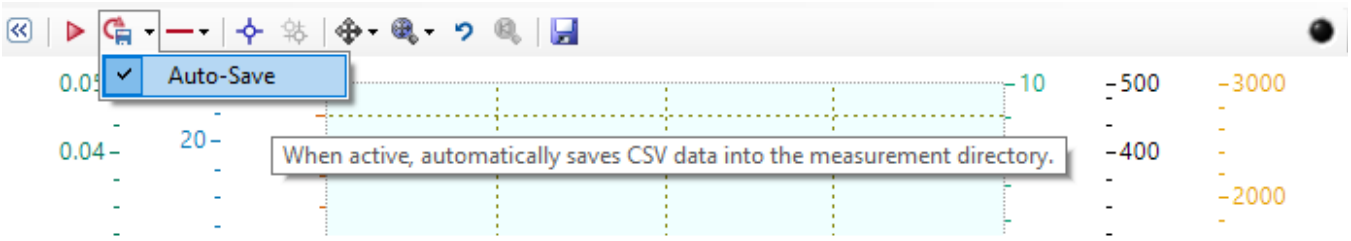


Figure 1: Auto-save is configured via the repeat button in the tool-strip. While enabled, the repeat button receives a disc floppy overlay.

Data is saved in sub-folders of the measurement directory of the current workspace. The sub-folders are named according to the time when the acquisition started, while the individual CSV files get a name according to the start time of the samples they hold.
This directory is most easily located using menu **File | Open Workspace Folder**, and navigating down the `Measurement` directory.

Note	With a larger acquisition duration, the scope doesn't roll to the side as usual when auto-save is enabled.

Auto-save is reset when the application closes.

Caution	The application does not detect the case where disk goes out of space.

Only one scope configuration companion file is saved for one directory.

1.2 Load Data

The menu **Scope |Load Plot Data…** opens the dialog shown in figure 2. Offline data is always shown in a separate window, the *Data Viewer*. This way, several Data Viewer windows might be opened. However, when invoking the menu from the Data Viewer, you have the choice to *append* new data to existing data, or *replace* data.

Several *data sets* can be imported at once or by appending the files incrementally. The *Alignment* option defines how data sets will be associated in the time axis. Choose *Align Left* when you have triggered data where the first sample of each data set shall coincide. Choose *Align Absolute* in order to strictly adhere to the date as specified in the file.

The second option is typically used when combining measurement data taken from different systems simultaneously, where the time bar consists of Tria-Link timestamps. Refer to section 2.3 in the Twincat Library: Diagnostics Application Note [3].

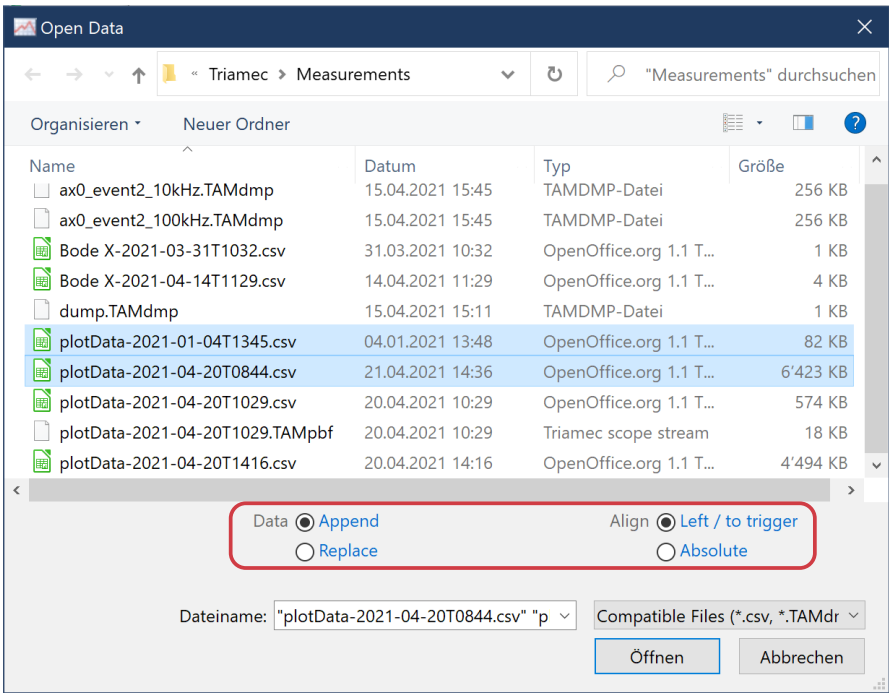


Figure 2: Open Data File Dialog: Select one or several files to import. Note the two radio button pairs adding to the standard elements.

When a companion scope configuration exists, as described in the above section, the configuration is loaded after importing the first data set. In contrast to the primary scope in the TAM System Explorer, existing plots are not replaced with those defined in the configuration. Instead, those plots matching the configuration are just reconfigured. Companion scope configurations of other data sets beyond the first imported one are ignored.

When importing more than one data set, a *template plot* is chosen from existing plots for each of the imported data columns. The template plot needs to have the same name as the imported data column. If none of the existing plots share the same name, an arbitrary existing plot is chosen instead. The new plot is then aligned to the template plot with respect to the alignment option, and assigned the same value axis as the template plot.

You can delete plots which aren't of current interest to you. However, you cannot restore plots without reloading the file.

Loading Auto-Save Data

It can make sense to overlay data from several files produced with Auto-Save activated. This works best when the data was acquired using a trigger.

Loading continuous data is not currently supported, though, and won’t look correct, even when choosing absolute alignment.

2 Working With Device Dump Files

Device dump files (`.TAMdmp`) are a part of the incident log created by devices for extraordinary events. They can be downloaded from the file system of the device, see AN124 [4].

Dumps can then be loaded just the same way as when opening CSV data. (see section 1.2).

The chronological alignment as shown in figure 2 works as follows:

- Align data *to trigger* in order to overlay different incidents of the same type, such that all trigger times coincide in the origin of the time axis. This is the default.
- Align *absolute* in order to preserve the time relation of different incidents. This will only render nicely when the incidents are near together.

3 Working With Streaming Data

Streaming is the tool for saving data from a long test run for later off-line analysis. While the scope can be used to see data spanning maximally 10-20 seconds, streaming duration is only limited by disk space.

Sections 3.1 describes how to stream to disk while plotting. Section 3.2 explains how to read from the persisted stream. For the specification of the streaming format, see section 4.4.

3.1 Streaming to Disk

Establish streaming by going to the Scope | General tab, and there to the category Sophisticated Behavior. Expand the Streaming property as shown in figure 3. Set the Enabled property to True. A default destination will be filled in above. Alternatively, you can specify whatever destination you like.

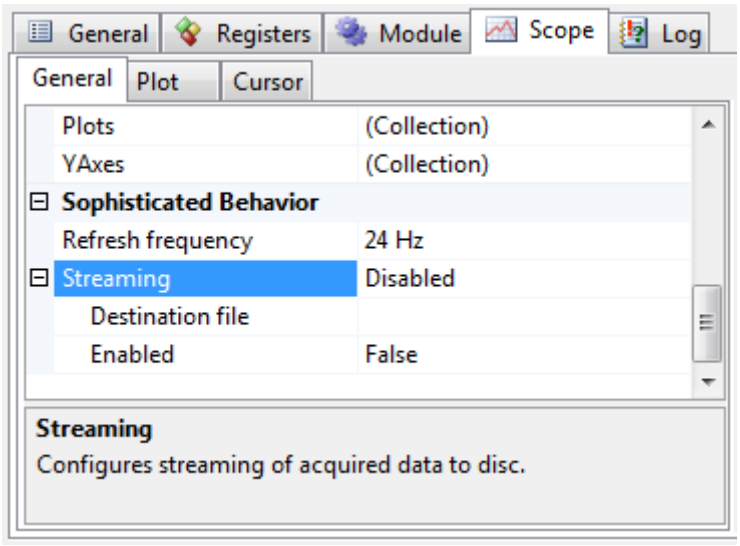


Figure 3: Configuring the scope to stream data to disk

Now, each time the scope is starting, the data stream will be written to the specified location. The stream is taken before triggering, that is, a continuous data stream is produced independently from the data actually shown in the scope. You can safely set any trigger condition without loosing valuable data.

Typically, you will have to enable repeat mode to enable continuous streaming.

Streaming will be more efficient if you lower the Refresh frequency (visible above the Streaming properties in figure 3).

Warning	The destination file is overwritten each time you start the scope.
Warning	When setting the time frame below 1.2 seconds, the scope stops data logging between each take. This will result in gaps within the stream. This might be desired in conjunction with a trigger.
Caution	The application does not detect the case where disk goes out of space.

Similar to saving CSV files, a companion scope configuration is saved, as described in section 1.1 above.

3.2 Load Streaming Data

A stream can be loaded for off-line analysis of measurement data just the same way as when opening CSV data (see section 1.2). The *Data* and *Align* settings are ignored when loading streaming data.

As opposed to loading CSV data, no data is initially plotted. Instead, you need to configure and start the scope yourself. Typically, repeat mode will be off, since the data is streamed much faster than real-time. Be aware that you cannot rewind until the end of the stream is reached or you load the same file again.

See also section 1.1 above regarding the companion scope configuration and deleting plots.

The ScopeStreaming sample [5] shows how to process a streaming file programmatically in C#.

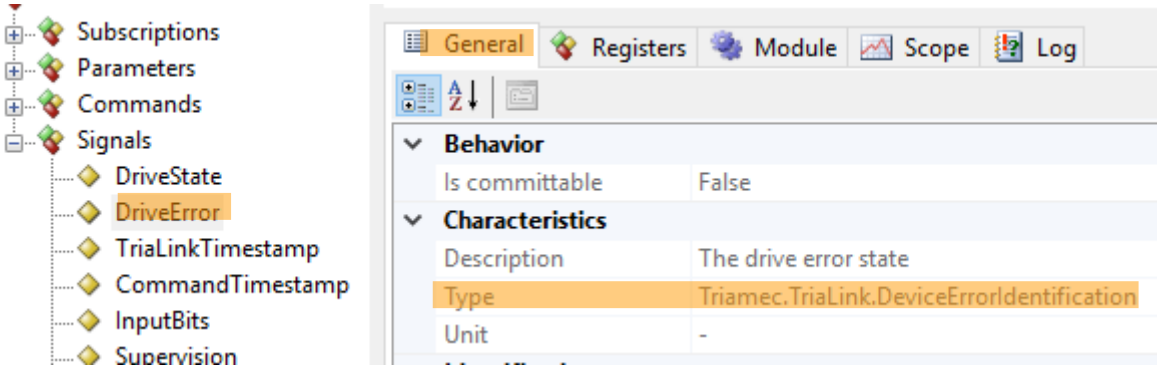
4 Data Formats

Section 4.1 explains how to interpret data from registers of an enumeration type. The remaining content is solely targeted to developers.

4.1 Enumerations

Enumeration values are saved in a numerical format. The name-value mapping can be retrieved with the following exemplary procedure:

1. In the TAM System Explorer, select the source register, and bring up the *General* tab.
2. Copy the value of the type, in this case, `Triamec.TriaLink.DeviceErrorIdentification`.



3. Via menu **Help | Documentation**, browse to the *Software* documentation, and there, open the `SWNET_TamApiReference-<version>_EP001.chm` document.
4. Bring up the *Index* tab, paste the type name there and press Enter. You can now see the documentation of the enumeration, including the values for each member.

4.2 CSV Data Format

The format used is comma-separated values. Every column contains the data of one signal. The first column is typically the time vector in the unit of seconds. Since the sampling time may be different from signal to signal, undefined samples are just left out.

Informal Definition

A typical data file, when imported into a spreadsheet, will look like this:

System:	Log-Version	0.1	Lib-Version	4.4
PC:	Host-Name	W7-DESKTOP-1	Time	2014-05-06T 12:53:47.4188998Z
SignalName: time	ActualCurrentQ	ControlOutputVoltageQ	PositionError	
Uri:	tam://localhost/A/L/S/ D/Register/Axes[0]/Sig- nals/CurrentController/ ActualCurrentQ	tam://localhost/A/L/S/ D/Register/Axes[0]/Sig- nals/CurrentController/ ControlOutputVoltageQ	tam://localhost/A/L/ S/D/Register/ Axes[0]/Signals/Posi- tionController/Posi- tionError	
Unit: s	A	V	m or rad	
Sampling [s]: 2E-05	2.00E-005	2.00E-005	2.00E-005	
3166.08977	-0.1171922684	-2.593126297	0	
3166.08979	-0.1197965443	-2.6025352478	-0.0032158147	
3166.08981	-0.1197965443	-2.6452968121	-0.0039303675	
3166.08983	-0.1210986823	-2.6758315563	-0.0046451855	
3166.08985	-0.1237029508	-2.6938037872	-0.0053602685	
3166.08987	-0.1289114952	-2.6660251617	0	

There is a total number of 10 header rows, followed by an arbitrary number of data rows. Each header row starts with a cell which is either empty or has a title text ending with a colon character. The title text defines the content of the cells in that header row.

The semantics of the headers is as follows:

PC

Contains the host name of the measurement system and the time related to the first data row.

SignalName

Contains the name of each signal.

In order for the first column to be recognized as the time bar, it needs to be named *time*. Otherwise, 1ms is assumed as sampling time.

Axis

Contains the index or name of the axis of the signal. This string will be prepended to the signal name.

Uri

Contains the source address of the signal. If the source exists, the synchronize button ➡️ above the plot legend will focus the source register.

Unit

Contains the unit of the signal. A small “u” may denote “m or rad”.

Sampling [s]

This row can be present in order to explicitly specify the sampling time of each signal. If not present, this information is computed from the time bar.

Note that the data is expected to be exactly equidistant. However, the time bar is allowed to specify non-equidistant durations between adjacent samples. The plots will correctly use that information in an x-y manner.

Factor

Specifies a scaling divisor to apply to each sampling point in order to get the actual value as declared by the unit. The plots will show the actual values after that computation. This column helps to save performance in real-time diagnostic tools.

Formal Definition

This section is to be done in general, but already contains some notes refining the informal definition given in the section above.

- The *list separator* is the comma, as defined by the invariant culture of the .NET framework.
- The title text of each header must end with the sequence of a colon and a space. The space needs to be there even if at end of line or before the list separator.
- The PC header needs to be structured exactly as shown in the above table.
- The number of header rows is not specified. However, some simple straight tools might assume a total of ten. Since the format is text, a manual modification towards the required number of rows is easily done.
- An empty header row consists either of an empty line, or an arbitrary number of list separators.

Compatibility

The format doesn't adhere to RFC 4180 [6] in that it typically contains several header lines before actual data lines start.

Importing data into Microsoft Excel isn't straightforward in German speaking countries. This article [7] provides information for different versions of Microsoft Excel.

Empty values in the data translate to empty cells in spreadsheets, or `NaN` in MATLAB, respectively.

4.3 Device Dump Files

The format of device dump files isn’t officially documented. The `Triamec.Tam.Acquisitions` namespace offers support to process dump files through the `DumpVariable`, `DumpFile` and related classes and the `ITamReadonlyRegister<T>.CreateVariable(DumpSignal)` extension method.

Please contact us if you would like to process device dump files without the TAM API.

4.4 Streaming Data Format

The format is *protobuf* [8] compressed with *gzip* [9]. The following `.proto` file describes the message format.

```
package Triamec.Acquisitions.Serialization;
import "bcl.proto"; // schema for protobuf-net's handling of core .NET types
message Info {
  optional string Address = 1;
  optional string Name = 2;
  optional string Unit = 3;
}
message Segment {
  // The number of 100-nanosecond intervals that have elapsed since
  // 12:00:00 midnight, January 1, 0001.
  required int64 StartTime = 1 [default = 0];
  // The number of 100-nanosecond intervals that make up the sampling time
  required int64 SamplingTime = 2 [default = 0];
  repeated double Data = 3;
  optional Variable Variable = 4;
}
message Variable {
  optional Info Info = 1;
  optional bool IsRegular = 2 [default = false];
  repeated bcl.NetObjectProxy Segments = 3; // reference-tracked Segment
}
message VariableCollection {
  repeated Segment items = 1; // always Variable
}
```

The imported `bcl.proto` looks like this:

```
package bcl;
message NetObjectProxy {
  // for a tracked object, the key of the **first** time this object was seen
  optional int32 existingObjectKey = 1;
  // for a tracked object, a **new** key, the first time this object is seen
  optional int32 newObjectKey = 2;
  // for dynamic typing, the key of the **first** time this type was seen
  optional int32 existingTypeKey = 3;
  // for dynamic typing, a **new** key, the first time this type is seen
  optional int32 newTypeKey = 4;
  // for dynamic typing, the name of the type (only present along with newTypeKey)
  optional string typeName = 8;
  // the new string/value (only present along with newObjectKey)
  optional bytes payload = 10;
}
```

Multiple messages are collated with 32-bit little-endian length prefixes.

Before implementing an application using the above information, consider using the `Triamec.Acquisitions.Protobuf` class, as demonstrated in [5].

Glossary

CSV Refers to the Comma-separated values file format.

References

[1] TAM System Explorer software web-page, 2019
https://www.triamec.com/de/tam-system-explorer.html

[2] “Drive Setup Guide”, `SW_TSD-TSP360-TSP710-Setup-Guide_EP001.pdf`, Triamec Motion AG, 2019

[3] “Twincat Library: Diagnostics Application Note”, `AN100_TwinCAT-Diagnostics_EP003.pdf`, Triamec Motion AG, 2019

[4] “Triamec File System: Application Note 124”, AN124_Filesystem_EP001.pdf, Triamec Motion AG, 2021

[5] “ScopeStreaming”, Triamec Motion AG, 2023
https://github.com/Triamec/ScopeStreaming

[6] “RFC 4180”, Internet Engineering Task Force, 2005
http://tools.ietf.org/html/rfc4180

[7] “Excel: CSV-Datei importieren”, PCtipp, 2017
https://www.pctipp.ch/tipps-tricks/kummerkasten/office/artikel/excel-csv-datei-oeffnen-bzw-importieren-27911/

[8] protobuf – Protocol Buffers – Google's data interchange format, 2014
https://github.com/protocolbuffers/protobuf

[9] GNU Gzip, 2019, http://www.gnu.org/software/gzip/

Revision History

Version	Date	Editor	Comment
001	2014-06-06	chm	Initial edit
002	2017-06-16	chm	Add reference to Matlab script
003	2018-09-14	chm	Cope with Enums
004	2019-01-14	chm	Introduce CSV auto-save
006	2021-04-26	chm	Introduce device dump files
007	2023-08-28	chm	Document streaming with C# sample

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.