Triamec Filesystem

Application Note 124

Version	Date	Editor	Comment
001	2021-04-26	MVX	First release
002	2021-09-07	DG	Renamed ColumnSize to RowSize

Document AN124_Filesystem_EP
Version 002
Source Q:\doc\ApplicationNotes\
Destination T:\doc\ApplicationNotes
Owner mvx

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.

Table of Contents

1 Introduction

We describe how the filesystem of Triamec drives may be accessed. This allows reading and writing large tables for encoder or cogging compensation and reading log files.

2 Preconditions

They require a firmware 4.11.0 or newer and a TAM System Explorer with version 7.15.0 or newer. Some connections require a running TAM System explorer as discussed in chapter 3.

3 Filesystem

The entry point to the filesystem is the web server of the drive. This may be accessed in three ways depending on the connection between drive and PC. The drive can be connected over

- Auxiliary Ethernet: Direct or company Ethernet as described in application note AN123
- USB
- Tria-Link PCI board

Hint: Access over the EtherCAT bus is currently not supported. Use USB or Auxiliary Ethernet for drives with EtherCAT bus.

Hint: If a Tria-Link drive is connected over Auxiliary Ethernet and set to BridgeMode with register General.Parameters.Bridge, it allows accessing all drives within the Tria-Link from Ethernet. This can be used for filesystem access too. Please be aware that the performance of exchanging data this way is limited due to buffer restrictions of the bridge drive.

The most intuitive way of accessing the entry point is using the TAM System Explorer. Open the Explorer and find the context menu of the drive node as shown in Figure 1. Choose the menu item "Browse" and a browser window will open as in Figure 2. This is the entry point of the drive web server and filesystem access (HTTP access).

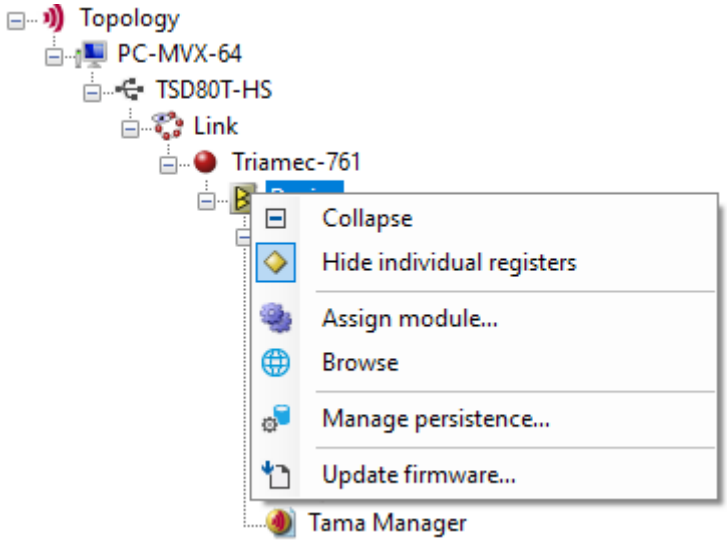

Figure 1: The context menu with the browse entry.

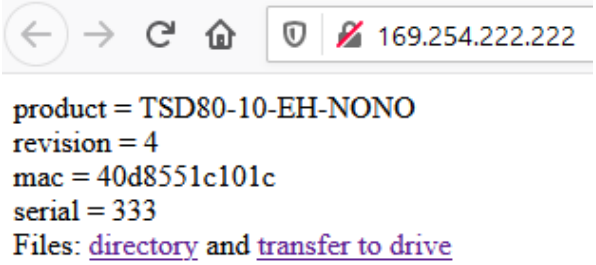

Figure 2: The browser window of the entry point.

This browser page contains two links, which are important for the filesystem: The "directory" and the "transfer to drive". These are discussed in the next two chapters.

Hint: Please note that filesystem access over USB and PCI requires a running TAM System Explorer. See ([1]) for technical details.

1 The IP address of the entry point depends on the connection type. If using USB or PCI, this address is **generated** (!) by the TAM System Explorer and this type of connection is only available as long as the Explorer runs. Over Auxiliary Ethernet, this IP address is independent of the Explorer and is discussed in AN123.

3.1 Directory

Open the directory using the link and the browser page Figure 3 appears.

The left column shows all the files, the drive knows. The second column shows the actual size in bytes. The third column is the maximum size of each file.

- If an entry is marked as a link, the file contains data and may be loaded from the drive to the PC by clicking on this link.
- If the entry is just a plain text (not in link format) the file is empty and the entry is just used as an indication of the maximum size of the file.

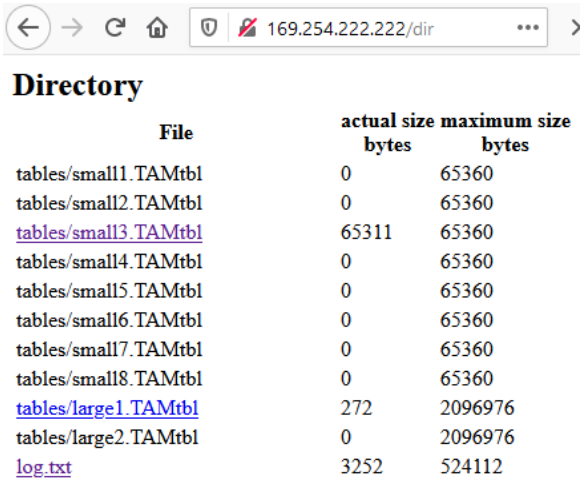

Figure 3: The directory page of the filesystem.

3.2 Transfer a file to the drive

Choose the "transfer to the drive" link in chapter 3 and the browser page Figure 4 opens.

- In the entry "Filepath in drive" enter a path name as shown in the directory Figure 3.
- In the entry "Select from PC" choose the file on your PC that you want to transfer to the drive.
- Then choose "Start" to start transmitting.

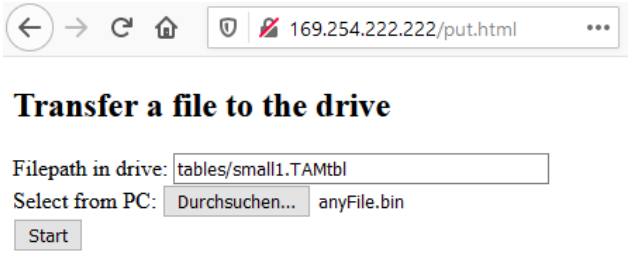

Figure 4: The transfer-a-file page

Onces the browser responses with "upload of tables/small3.bin succeeded", the file has been saved successfully to the drive ram and is accessible from Tama code.

Warning: If a file is a persistent file (see chapter Tables), the internal saving to the permanent memory is not finished at this time. This process starts immediately after the browser finished transmitting and may take up to one second. You can work with the drive as usual and upload other files, but you should not power down the drive during this phase.

4 Tables

Tables can be used in a user real-time application (Tama), e.g., for encoder or cogging compensation. Currently we support 8 small tables (≈16'000 entries) and 2 large tables (≈500'000 entries).

A table contains a header and data. These may be accessed from the TAM System Explorer (chapter 4.1) or a user realtime application (Tama, chapter 4.2) or from the file server (chapter 4.3).

The structure and size of a table (chapter 4.4) is specified with the header as discussed in chapter 4.5 and must be commited. This header also specifies, whether the table is persistent or temporary.

4.1 TAM System Explorer access

To access the table "Small1" for example, use register General.Application.Tables.Small1. This contains the following elements

Command

The command register allows to run one of the following table command:

- *Commit* This command is described below
- *Reload* Reload a persistent table from the persistent memory
- *Erase* Erase the persistent memory of this table and set values to default.

Committing a table calculates the size of the table from the header parameters (see chapter Structure below). After this the table and its header can be read from the filesystem. If the file is persistent (see chapter 4.5) the header and data are saved to the persistent memory.

Warning: Committing a persistent table can wear out the persistent memory. If a certain limit has been reached, committing a table is denied with an error message.

Changing the table header without committing does not change the size of the table as visible from the filesystem nor the header seen from the filesystem nor change persistent memory. Repeat commit after changing the header and the new header will be visible from the filesystem and in persistent memory.

Changing the table data after committing the table changes the temporary memory (RAM) and will be immediately visible over the filesystem but does not update persistent memory (FLASH).

Header

The table header is described in chapter 4.5.

Data

The data entries allow reading one entry of the table.

- Data.Index Specify the Index, at which data is to be read for the entries below
- Data.Float shows the value in float format (32 bits).
- Data.Integer shows the value in Integer format (32 bits).
- Data.Double shows the value in double format (64 bits).

The value cannot be modified. Use Tama or the filesystem to set the values of the Table.

Please note that Double requires two 32bit entries of the table. This means a double value at index i occupies float or integer entries at indices 2*i and 2*i+1.

State

This shows the current state of the table. 3 means the table is ready.

4.2 Tama Code access

From Tama a table is accessed the same as using the TAM System Explorer with two exceptions:

The data entries of a table are arrays that can directly be accessed with theirs indices. To set the float value of table Small1 at index 10000, for example, simply use the code

```
Register.Application.Tables.Small1.Data.Float[10000] = 1.234f;
```

To get the maximum size of a table use

```
int len = Register.Application.Tables.Small1.Data.Float.Length;
```

To save the table persistently call the following **once**. Do not call frequently to prevent flash wear!

```
Register.Application.Tables.Small1.Header.Dim1.Size = size;
Register.Application.Tables.Small1.Header.Dim2.Size = 1;
Register.Application.Tables.Small1.Header.Dim3.Size = 1;
Register.Application.Tables.Small1.Header.RowSize = 1;
Register.Application.Tables.Small1.Header.Persistent = true;
Register.Application.Tables.Small1.Command = TableCommand.Commit;
```

4.3 File server access

Tables are accessed as files using the file server as described in chapters 3.1 and 3.2.

The file starts with the table header and continues with the table data.

- The header occupies 64 words of 32bits each and is described in chapter 4.5.
- The table data consists of 32bit float or Integer entries or 64bit double entries.

The size of this file in bytes is calculated during Table Commit by

size = 4 * (64 + Header.RowSize * Header.Dim1.Size * Header.Dim2.Size * Header.Dim3.Size)

Note: The table is a binary file with a flat sequence of binary LittleEndian 32bit or 64bit entries without Tabulator or End-of-Line characters.

If a table is transmitted from the PC to the drive, an internal Commit of the Table takes place to provide consistency with the internal register. If the table is marked persistent (see header below) the table will then be saved to the persistent memory.

If a table is loaded from the drive to the PC, the table header is taken from the last commited version. The data part of the file is are always taken from the most recent data even if the table has not been commited since changing data.

4.4 Structure

The structure of the table is flexible. For a standard one dimensional table of Float values set

- Header.RowSize = 1
- Header.Dim1.Size = number of Float values in the table
- Header.Dim2.Size = 1
- Header.Dim3.Size = 1

For a standard one dimensional table of Double values set

- Header.RowSize = 2
- Header.Dim1.Size = number of Double values in the table
- Header.Dim2.Size = 1
- Header.Dim3.Size = 1

For a three dimensional table of Float values set

- Header.RowSize = 1
- Header.Dim1.Size = number of entries in the first dimension
- Header.Dim2.Size = number of loops in the second dimension
- Header.Dim3.Size = number of loops in the third dimension

and generate the values of the table by something like the following code example, which sets the table to a constant value 1.23f at the three dimensional position (pos1, pos2, pos3):

```
int size1 = Register.Application.Tables.Small1.Header.Dim1.Size;
int size2 = Register.Application.Tables.Small1.Header.Dim2.Size;
int size3 = Register.Application.Tables.Small1.Header.Dim3.Size;
for (int k = 0; k < size3; k++) {
   for (int j = 0; j < size2; j++) {
      for (int i = 0; i < size1; i++) {
         int index = i + size1 * (j + size2 * k);
         Register.Application.Tables.Small1.Data.Float[index] = 1.23f;
         float pos1 =  Register.Application.Tables.Small1.Header.Dim1.StartValue +
                       Register.Application.Tables.Small1.Header.Dim1.Distance * (float)i;
         float pos2 =  Register.Application.Tables.Small1.Header.Dim2.StartValue +
                       Register.Application.Tables.Small1.Header.Dim2.Distance * (float)j;
         float pos3 =  Register.Application.Tables.Small1.Header.Dim3.StartValue +
                       Register.Application.Tables.Small1.Header.Dim3.Distance * (float)k;
      }
   }
}
```

4.5 Header

The header fields are shown in the following table.

Word number	Type	Register name	Description
0	Bool	Persistent	0=Table is Volatile, 1=Table is Persistent
1-2	Integer32	-	Must be 0
3	Integer32	checksumMode	{0=Ignore, 1=Check, 2=Calculate}, see chapter 4.6
4-15	Integer32	checksum	The SHA-3-384 checksum with NIST padding, set zero before calculation.
16-17	Integer64	Date	The date in 64 bit POSIX format
18	Integer64	-	Must be 0
19	Integer32	Id	A table ID given by the user
20-35	String	Description	A description string given by the user
36	Integer32	RowSize	The number of words in a row
37-39	Integer32	-	Must be 0
40 41 42 43	Integer32 Integer32 Float32 Float32	Dim1.Size - Dim1.StartValue Dim1.Distance	The size of the table in the first dimension Must be zero The position of the first data point of this dimension The distance between data points in this dimension
44 45 46 47	Integer32 Integer32 Float32 Float32	Dim2.Size - Dim2.StartValue Dim2.Distance	The size of the table in the first dimension Must be zero The position of the first data point of this dimension The distance between data points in this dimension
48 49 50 51	Integer32 Integer32 Float32 Float32	Dim3.Size - Dim3.StartValue Dim3.Distance	The size of the table in the first dimension Must be zero The position of the first data point of this dimension The distance between data points in this dimension
52-63	Integer32	-	Must be 0

4.6 Checksum calculation

A checksum may be attached to the header. This checksum is tested in the drive its checksumMode is "Check". If a file is transmitted to the drive with checksumMode "Calculate", the drive will change the checksumMode to "Check" and then calculate the checksum itself. This is useful if the user does not want to calculate the checksum himself. By reading back this file, he gets a file with the checksum mode enforced.

The checksum is calculated with the SHA-3-384 method. Before calculation, zero the checksum, add NIST-type of padding , then calculate the SHA3-384 hash of the file.