



Triamec Drive File System

Application Note 124

A description of the file system of Triamec drives and how it is accessed. This allows reading and writing tables, i.e. for compensation data and accessing log files via the drive's web server.

The file system is available in firmware 4.11 and newer and is accessible with a *TAM System Explorer* version 7.15 or newer, or with a browser.

Table of Contents

1	Browser Access.....	2	2.1	TAM API.....	4
1.1	Directory.....	3	2.2	Low level interface.....	4
1.2	Transfer Files to Drive.....	3		References.....	7
2	Software Access.....	4		Revision History.....	7

1 Browser Access

The entry point to the file system is the web server of the drive. For that, a valid connection to the drive has to be set up (see [1]). Be aware that different connection types might not have the same performance (round trip time, etc.).

The most intuitive way of accessing the file system is using the *TAM System Explorer*. Use 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 file system access (HTTP access).

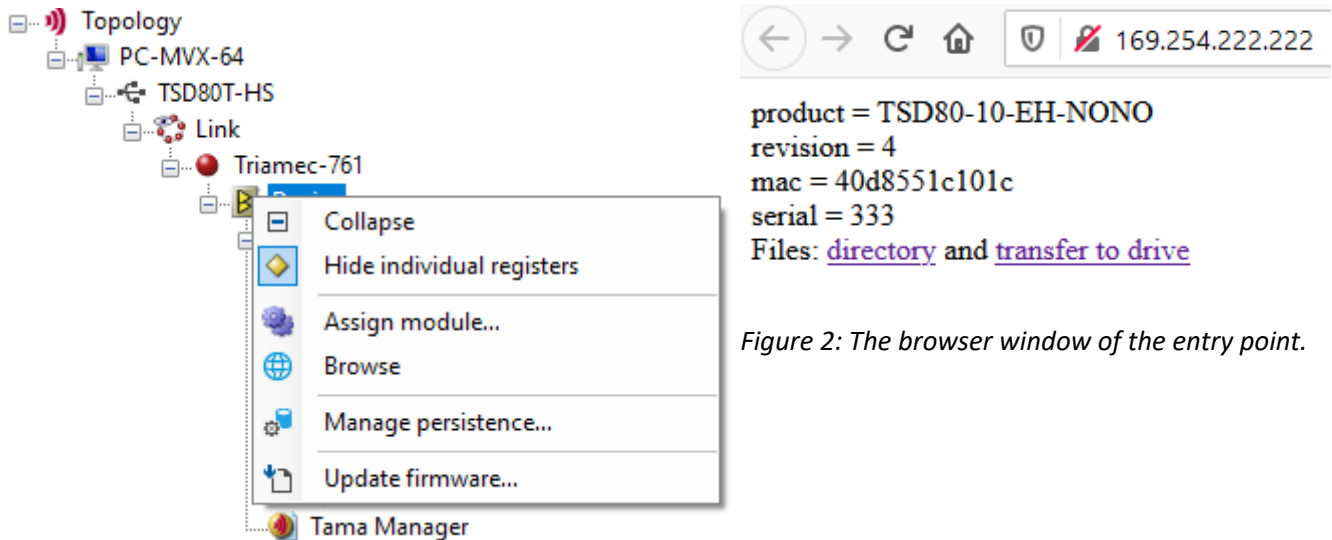


Figure 1: The context menu with the browse entry.

This page contains two links, which are important for the file system: The **directory** and the **transfer to drive**. These are explained in the next two chapters.

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

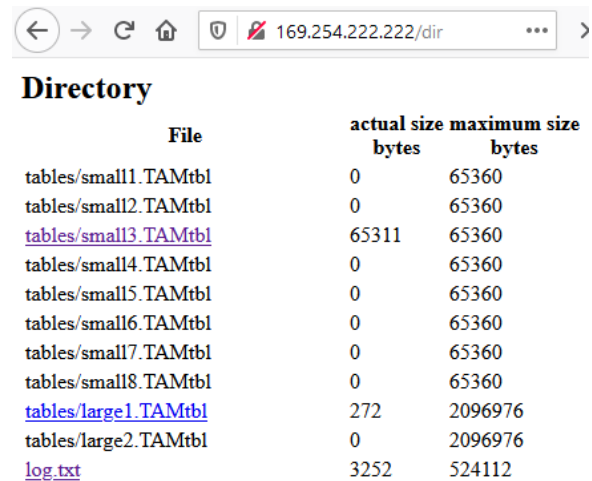
1 The IP address of the entry point depends on the connection type. If using USB or PCI, this address is generated by the *TAM System Explorer* and this type of connection is only available as long as the Explorer runs. Over Ethernet, this IP address is independent of the *TAM System Explorer* and is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

1.1 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 allowed 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 its link.
- If the entry is plain text (without link), the file is empty and the entry is used as an indication of the maximum size of the file.



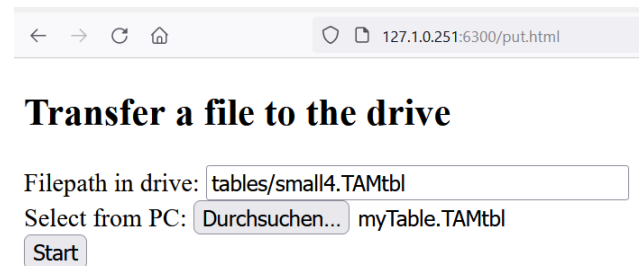
File	actual size bytes	maximum size bytes
tables/small1.TAMtbl	0	65360
tables/small2.TAMtbl	0	65360
tables/small3.TAMtbl	65311	65360
tables/small4.TAMtbl	0	65360
tables/small5.TAMtbl	0	65360
tables/small6.TAMtbl	0	65360
tables/small7.TAMtbl	0	65360
tables/small8.TAMtbl	0	65360
tables/large1.TAMtbl	272	2096976
tables/large2.TAMtbl	0	2096976
log.txt	3252	524112

Figure 3: The directory page of the file system.

1.2 Transfer Files to Drive

Choose the **transfer to the drive** link in chapter 1 and the browser page Figure 4 opens.

- In the entry **Filepath in drive** enter a path and file name as available in the directory (see 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.



Transfer a file to the drive

Filepath in drive:

Select from PC:

Figure 4: The transfer-a-file page

Once the browser responses with **upload of tables/small4.TAMtbl 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, 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.

2 Software Access

2.1 TAM API

Use the *DeviceWebService* from the `Triamec.Tam.Core` NuGet for convenient handling of the file system within .NET.

2.2 Low level interface

External software can access the filesystem using the IP-address shown in chapter 1. This is especially useful if the PC is connected with the drive over its auxiliary Ethernet port. With a TCP connection to port 80, external software can read and write files discussed in this document. The following code snippets show how to read and write a file.

All possible files are listed using “GET /dir”.

Reading a file from the drive

Reading is done with a standard HTTP GET from the drive. To read the table “tables/small1.TAMtbl” use “GET /tables/small1.TAMtbl”. The following samples read the table “small1” from the drive, if the drive is connected using its AUTO-IP address 169.254.222.222.

using C#

```
var FilePath = new Uri("tables/small1.TAMtbl", UriKind.Relative); // URL to file
var baseAddress = new Uri("http://169.254.222.222/"); // drive URL
var client = new HttpClient { BaseAddress = baseAddress };
var response = new HttpResponseMessage(HttpStatusCode.Forbidden);
using (var request = new HttpRequestMessage() {
    RequestUri = FilePath,
    Method = HttpMethod.Get,
}) {
    if (client.BaseAddress != null) {
        response = await client.SendAsync(request).ConfigureAwait(false);
    }
}

if (response.StatusCode == HttpStatusCode.OK) {
    using (var fileStream = await response.Content.ReadAsStreamAsync().ConfigureAwait(false)) {
        // ...add deserialize code here
    }
}
```

using JavaScript

```
const fs = require('fs');
const axios = require('axios');
const FormData = require('form-data');
```

```
async function uploadFile(filePath, driveIP) {
  try {
    const data = await fs.promises.readFile(filePath);
    const formData = new FormData();
    formData.append('filename', 'tables/small1.TAMtbl');
    formData.append('filepath', data, { filename: 'anyname.bin' });
    const response = await axios.post('http://' + driveIP + '/put.html', formData, {
      headers: {
        ...formData.getHeaders(),
        'Content-Length': formData.getLengthSync()
      }
    });
    console.log('Response:', response.data);
  } catch (error) {
    console.error('Error uploading file:', error);
  }
}

uploadFile('dataRead.bin', '169.254.222.222');
```

Writing a file to the drive

Files are written using HTTP POST with a specially formatted MultipartForm object. It must contain a string component "filename" and a stream component "filepath". The following samples write a file "sample.bin" to the drive table "small1", if the drive is connected using its AUTO-IP address 169.254.222.222.

using C#

```
var targetName = "tables/small1.TAMtbl";
var source = new FileStream("sample.bin", FileMode.Open); // stream of sample.bin to be transferred

var uploadSite = new Uri("put.html", UriKind.Relative); // URL for file transfer
var baseAddress = new Uri("http://169.254.222.222/"); // drive URL
var client = new HttpClient { BaseAddress = baseAddress };

using (var content = new MultipartFormDataContent {
    { new StringContent(targetName), "filename" },
    { new StreamContent(source), "filepath" }
})
using (var response = await client.PostAsync(uploadSite, content)
    .ConfigureAwait(continueOnCapturedContext: false))
{
    if (!response.IsSuccessStatusCode)
    {
        throw new HttpRequestException($"Transfer to device at {targetName} failed. {response.ReasonPhrase}");
    }
}
```

using JavaScript

```
// Create a Blob from the binaryData ArrayBuffer
const blob = new Blob([binaryData], { type: 'application/octet-stream' });
// Create a FormData object and append the Blob to it
let formData = new FormData();
const tableSlot = 'tables/small1.TAMtbl';
formData.append('filename', tableSlot);
formData.append('filepath', blob);
// Use the Fetch API to send the FormData to the specified URL
const response = await fetch('http://169.254.222.222/put.html', {
  method: 'POST',
  body: formData,
});
if (response.ok) {
  console.log('Binary data sent successfully.');
```

```
}
else {
  throw new Error(`Failed to send binary data. Status: ${response.status}`);
}
```

References

[1] “Servo Drive Setup Guide”, ServoDrive-SetupGuide_EP027.pdf, Triamec Motion AG, 2024.

Revision History

Version	Date	Editor	Comment
001	2021-04-26	mvx	First release
002	2021-09-07	dg	renamed ColumnSize to RowSize
003	2023-03-02	sm	update template, fix header index 18 type, minor wording changes
004	2023-05-30	sm	Introduce Table Type for firmware contained features.
005	2023-08-16	sm	Fix index of Table Type.
006	2023-11-16	mvx	New interface for reading data of a table in firmware 4.20
007	2024-02-07	dg	Fix table path: /table/small1.TAMtbl → /tables/small1.TAMtbl
008	2024-03-11	ns	Fix JavaScript spelling
009	2024-04-11	dg	Size of /table/large1.TAMtbl increased to 2'000'000
010	2024-08-09	fm	Small review and added reference
011	2025-02-12	ns	Move Table content to AN125

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 © 2025
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 contains proprietary information belonging to Triamec Motion AG and must not be distributed.

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.