METAS VNA Tools - Data Formats V2.5.4

Michael Wollensack

May 2022

Contents

1 Introduction 4
  1.1 S-Parameter Data ........................................ 4
  1.2 VNA Data .................................................. 5
  1.3 Data Collections ......................................... 5
    1.3.1 S-Parameter Data Collection .......................... 5
    1.3.2 VNA Data Collection .................................. 6
  1.4 Overview .................................................. 6

2 sdatb File Specification 7
  2.1 Binary Structure Version 1 ................................ 7
    2.1.1 Uncertainty Numbers ................................ 7
  2.2 Binary Structure Version 2 ................................ 8
    2.2.1 Flat Vector of Uncertainty Numbers ................. 8
  2.3 Binary Structure Version 3 ................................ 9
    2.3.1 VNA Port Description ................................. 9
  2.4 Binary Structure Version 4 ................................ 10
    2.4.1 Frequency Conversion Subset ........................ 10
  2.5 Binary Structure Version 5 ................................ 10
    2.5.1 Frequency Conversion ................................ 11
  2.6 Example MATLAB Code .................................... 11

3 sdatx File Specification 15
  3.1 XML Schema .............................................. 15
    3.1.1 Frequency and Port Lists ............................. 15
    3.1.2 Uncertainty Numbers ................................ 17
4 sdatcv File Specification
   4.1 Header Lines ................................................. 19
       4.1.1 Port Assignment ........................................ 19
       4.1.2 Reference Impedance .................................... 19
       4.1.3 Data Column Description ................................ 19
   4.2 Data Lines .................................................. 20
       4.2.1 1-Port Example ........................................... 20
       4.2.2 2-Port Examples ......................................... 21
   4.3 Comment Lines ............................................... 22

5 Touchstone V1.x snp File Specification
   5.1 Examples ..................................................... 23

6 Touchstone V2.0 ts File Specification
   6.1 Examples ..................................................... 24

7 vdatb File Specification
   7.1 Binary Structure Version 1 .................................. 25
       7.1.1 VNA Parameter Data ..................................... 25
       7.1.2 VNA Parameter ........................................... 25
       7.1.3 VNA Parameter Examples ................................. 26
   7.2 Binary Structure Version 2 .................................. 26
   7.3 Binary Structure Version 3 .................................. 27
   7.4 Binary Structure Version 4 .................................. 27
   7.5 Binary Structure Version 5 .................................. 28
   7.6 Example MATLAB Code ....................................... 28

8 vdatx File Specification
   8.1 XML Schema .................................................. 32
       8.1.1 VNA Parameter Description .............................. 32

9 vdatcv File Specification
   9.1 Header Lines ................................................ 34
       9.1.1 Port Assignment ........................................ 34
       9.1.2 Reference Impedance .................................... 34
       9.1.3 Data Column Description ................................. 34
   9.2 Data Lines ................................................... 35
       9.2.1 Example .................................................. 35
   9.3 Comment Lines ............................................... 36

10 CITI File Specification
    10.1 Examples .................................................. 37

11 scolb File Specification
    39
## 12 scolcv File Specification

12.1 Header Lines ......................................................... 39
  12.1.1 Number ......................................................... 40
  12.1.2 Name .......................................................... 40
  12.1.3 Port Assignment ............................................... 40
  12.1.4 Reference Impedance ......................................... 40
  12.1.5 Data Column Description ................................. 40
12.2 Data Lines .......................................................... 40
  12.2.1 Example ........................................................ 40
12.3 Comment Lines ....................................................... 41

## 13 vcolb File Specification

14 vcolcv File Specification

  14.1 Header Lines ......................................................... 42
    14.1.1 Number ......................................................... 43
    14.1.2 Name .......................................................... 43
    14.1.3 Port Assignment ............................................... 43
    14.1.4 Reference Impedance ......................................... 43
    14.1.5 Data Column Description ................................. 43
14.2 Data Lines .......................................................... 43
  14.2.1 Example ........................................................ 44
14.3 Comment Lines ....................................................... 45

A ZIP File with Embedded Data Files

B PDF File with Embedded Data Files

  B.1 Example .......................................................... 46

C PTB DCC XML File with Embedded Data Files

  47
1 Introduction

Touchstone and CITI are two well known file formats for storing S-parameters, but both of them don’t support uncertainties with dependencies. Uncertainties with dependencies are a method with low memory consumption for taking correlations into account, see METAS UncLib. The solution is to define new data formats and file types which support uncertainties with dependencies.

The in the following described data and file formats were developed for VNA Tools. Including uncertainties increases the file size drastically. Several file formats which include compression (ZIP) are proposed for this reason.

A file format is a mapping of a data format. Thus the two data formats which are used in VNA Tools are described first. The following sections contain descriptions of the files formats which are derived from the data formats.

1.1 S-Parameter Data

S-Parameter Data is a data format. Most of the calculations in VNA Tools are done with the S-Parameter Data type. It is the main data format in VNA Tools. S-Parameter Data contains the following properties:

- Frequency List (1d array of double)
- Port Assignment (1d array of VnaPortDescription)
- Port Impedance (1d array of ComplexUncNumber) \(^1\)
- Frequency Conversion (1d array of FrequencyConversion) \(^2\)
- Data (3d array of ComplexUncNumber)
  - Index 0: Frequency
  - Index 1: Receiver Port
  - Index 2: Source Port

As can be seen from the properties, this data format is well suited for storing S-parameters. S-Parameter Data supports the following file types:

METAS sdatb is a binary file format which contains the full information,

METAS sdatx is an XML file format which contains the same information as sdatb,

METAS sdatcv is an ASCII text file format which contains only a subset of the information (no correlation between frequency points and different sdatcv files),

Touchstone snp, ts is an ASCII text file format which doesn’t contain uncertainty information.
1.2 VNA Data

VNA Data is another data format in VNA Tools. It is used in the visualization part of VNA Tools to display receiver values and to be compatible with old file formats. VNA Data contains the following properties:

- Frequency List (1d array of double)
- Port Assignment (1d array of VnaPortDescription)
- Port Impedance (1d array of ComplexUncNumber) \(^1\)
- Frequency Conversion (1d array of FrequencyConversion) \(^2\)
- Parameter Data (1d array of VnaParameterData)
  - Parameter (VnaParameter)
  - Data (1d array of ComplexUncNumber)

As can be seen from the properties, this data format is suited for storing arbitrary receiver ratios as well as receiver values. It's more general than S-Parameter Data. VNA Data supports the following file types:

**METAS vdatb** is a binary file format which contains the full information,

**METAS vdatx** is an XML file format which contains the same information as vdatb,

**METAS vdatcv** is an ASCII text file format which contains only a subset of the information (no correlation between frequency points and different files),

**CITI** is an ASCII text file format which contains only a subset of the information (no correlation and no port impedance).

1.3 Data Collections

A single Data Collection file can contain either multiple S-Parameter Data files or multiple VNA Data files.

1.3.1 S-Parameter Data Collection

S-Parameter Data Collection supports the following file types:

**METAS scolb** is a zip file format which contains multiple sdatb files,

**METAS scolcv** is an ASCII text file format which contains multiple standards (correlation between different standards in the same scolcv file) but only a subset of the information (no correlation between frequency points and different scolcv files).

\(^1\) The port impedance is the complex reference impedance. It can be different for each port but not for each frequency. Complex reference impedance in function of frequency needs to be re-normalized. For changing the complex reference impedance see the appendix “Transmission Line Junction” of the METAS VNA Tools - Math Reference.

\(^2\) The frequency conversion property is optional. It can be different for each port.
1.3.2 VNA Data Collection

VNA Data Collection supports the following file types:

**METAS vcolb** is a zip file format which contains multiple vdatb files,

**METAS vcolcv** is an ASCII text file format which contains multiple standards (correlation between different standards in the same vcolcv file) but only a subset of the information (no correlation between frequency points and different vcolcv files).

### 1.4 Overview

Table 1 shows an overview of the different file formats. Each file format can be converted into another from this table.

<table>
<thead>
<tr>
<th>Format</th>
<th>Type</th>
<th>Port Assignment</th>
<th>Port Impedance</th>
<th>S-Parameter</th>
<th>Mixed-Mode P.</th>
<th>Freq. Conversion</th>
<th>Receiver Values</th>
<th>Arbitrary Ratios</th>
<th>Collection</th>
<th>Uncertainties</th>
<th>Correlation</th>
<th>Dependencies</th>
<th>ZIP Compression</th>
<th>No Redundancy</th>
<th>File Size</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>METAS sdatb V1</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS sdatb V2</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS sdatb V3</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS sdatb V4</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS sdatb V5</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS sdatx</td>
<td>xml</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS sdatcv</td>
<td>text</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>Touchstone V1.x snp</td>
<td>text</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>Touchstone V2.0 ts</td>
<td>text</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vdatb V1</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vdatb V2</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vdatb V3</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vdatb V4</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vdatb V5</td>
<td>binary</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vdatx</td>
<td>xml</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vdatcv</td>
<td>text</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>CITI</td>
<td>text</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS scolb</td>
<td>zip</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS scolcv</td>
<td>text</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vcolb</td>
<td>zip</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
<tr>
<td>METAS vcolcv</td>
<td>text</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td>🔴🔴🔴🔴</td>
<td></td>
</tr>
</tbody>
</table>
2  sdatb File Specification

The sdatb-file format is a binary file type developed by METAS. The file can be written in a GZIP file stream to reduce the file size. The byte ordering is little-endian.

2.1  Binary Structure Version 1

Version 1 of sdatb uses a GZIP file stream to reduce the file size. The following enumeration describes the binary structure of a sdatb-file:

1. Header (string), value: ‘%SDATA’
2. Version (int32), value: 1
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Frequency List (double[]), size: number of frequencies
6. Ports (int32[]), size: number of ports
7. Port Impedance (ComplexUncNumber[]), size: number of ports
8. Data (ComplexUncNumber[,]), size dim 0: number of frequencies, size dim 1 and 2: number of ports.

2.1.1  Uncertainty Numbers

The following enumeration describes the binary structure of ‘ComplexUncNumber’:

1. Version (int32), value: 1
2. Real (UncNumber)
3. Imag (UncNumber)

The following enumeration describes the binary structure of ‘UncNumber’:

1. Version (int32), value: 1
2. Value (double)
3. Version2 (int32), value: 4
4. Number of Dependencies (int32)
5. Dependencies (DependsOn[]), size: number of dependencies

The following enumeration describes the binary structure of ‘DependsOn’:

1. Number of Id Bytes (int32)
2. Input Id (byte[]), size: number of id bytes
3. Input Description (string)
4. Input IDof (double)
5. Jacobi (double)

2.2 Binary Structure Version 2

Version 2 of sdatb is not using a GZIP file stream. The redundancy of the data is removed by storing uncertainty inputs in a look up table. The uncompressed file size of version 2 is comparable to the GZIP compressed file size of version 1 and about four times smaller than the uncompressed file size of version 1. Avoiding the GZIP step and reducing the uncompressed file size speeds up loading and saving of files. The following enumeration describes the binary structure of a sdatb-file:

1. Header (string), value: '%SDATA'
2. Version (int32), value: 2
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Frequency List (double[]), size: number of frequencies
6. Ports (int32[]), size: number of ports
7. Flat Vector (UncNumber[]), size: $2N_{\text{Ports}} + 2N_{\text{Freq}}N_{\text{Ports}}$

The last element contains the port impedance and the data mapped to a flat vector of uncertainty numbers.

2.2.1 Flat Vector of Uncertainty Numbers

The following enumeration describes the binary structure of ‘FlatVectorUncNumbers’:

1. Version (int32), value: 1
2. Length (7-bit encoded int)
3. Values (double[]), size: length
4. Number of Inputs (7-bit encoded int)
5. Inputs (UncInput[]), size: number of inputs
6. Dependencies (UncDependencies[]), size: length

The following enumeration describes the binary structure of ‘UncInput’:

1. Temp (byte), bit 0: same id size, bit 1: empty description, bit 2: zero idof, bit 3-7: 0
2. Id Size (7-bit encoded int), field only present if not same id size
3. Input Id (byte []), size: id size
4. Input Description (string), field only present if not empty description
5. Input IDof (double), field only present if not zero idof

The following enumeration describes the binary structure of ‘UncDependencies’
1. Number of Dependencies (7-bit encoded int)
2. Dependencies (UncDependency[]), size: number of dependencies, pointer to inputs set to 0.

The following enumeration describes the binary structure of ‘UncDependency’
1. Relative Pointer to Inputs (7-bit encoded int)
2. Jacobi (double)

### 2.3 Binary Structure Version 3

Version 3 of sdatb is an extension to version 2 which adds support for mixed-mode S-parameters and port indices. The following enumeration describes the binary structure of a sdatb-file:

1. Header (string), value: ‘%SDATA’
2. Version (int32), value: 3
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Frequency List (double[]), size: number of frequencies
6. Ports (VnaPortDescription[]), size: number of ports
7. Flat Vector (UncNumber[]), size: $2^{N_{Ports}} + 2^{N_{Freq}} N_{Freq} N_{Ports}$

The last element contains the port impedance and the data mapped to a flat vector of uncertainty numbers.
For the definition of ‘Flat Vector of Uncertainty Numbers’ see section 2.2.1.

#### 2.3.1 VNA Port Description

The following enumeration describes the binary structure of ‘VnaPortDescription’:

1. Port (int32)
2. Mode (VnaPortMode)
3. Index (int16)

‘VnaPortMode’ is an enumeration represented by a 16-bit integer where 0 is single-ended ‘s’, 1 is differential mode ‘d’ and 2 is common mode ‘c’. 
2.4 Binary Structure Version 4

Version 4 of sdatb is an extension to version 3 which adds support for frequency converting S-parameters. The following enumeration describes the binary structure of a sdatb-file:

1. Header (string), value: ‘%SDATA’
2. Version (int32), value: 4
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Frequency List (double[]), size: number of frequencies
6. Ports (VnaPortDescription[]), size: number of ports
7. Frequency Conversion List (FrequencyConversionSub[]), size: number of ports
8. Flat Vector (UncNumber[]), size: $2N_{\text{Ports}} + 2N_{\text{Freq}}N_{\text{Ports}}$

The last element contains the port impedance and the data mapped to a flat vector of uncertainty numbers.

For the definition of ‘VnaPortDescription’ see section 2.3.1. For the definition of ‘Flat Vector of Uncertainty Numbers’ see section 2.2.1.

2.4.1 Frequency Conversion Subset

The following enumeration describes the binary structure of ‘FrequencyConversionSub’:

1. Numerator (double)
2. Denominator (double)
3. Offset (double)

2.5 Binary Structure Version 5

Version 5 of sdatb is an extension to version 4 which adds support for frequency converting S-parameters, where the receiver frequency is not equal to the source frequency. The following enumeration describes the binary structure of a sdatb-file:

1. Header (string), value: ‘%SDATA’
2. Version (int32), value: 5
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Frequency List (double[]), size: number of frequencies
6. Ports (VnaPortDescription[]), size: number of ports
7. Frequency Conversion List (FrequencyConversion[]), size: number of ports
8. Flat Vector (UncNumber[]), size: \(2N_{\text{Ports}} + 2N_{\text{Freq}}N_{\text{Ports}}\)

The last element contains the port impedance and the data mapped to a flat vector of uncertainty numbers.

For the definition of ‘VnaPortDescription’ see section 2.3.1. For the definition of ‘Flat Vector of Uncertainty Numbers’ see section 2.2.1.

### 2.5.1 Frequency Conversion

The following enumeration describes the binary structure of ‘FrequencyConversion’:

1. Test Receiver (FrequencyConversionSub)
2. Reference Receiver (FrequencyConversionSub)
3. Source (FrequencyConversionSub)

For the definition of ‘FrequencyConversionSub’ see section 2.4.1.

### 2.6 Example MATLAB Code

The following code shows how to load a sdatb-file in MATLAB:

```matlab
function d = LoadSDATB ( filepath )
% Loads VNA Tools II SParamData (*.sdatb) file
% Michael Wollensack METAS - 22.04.2022

d = {};\n
f3 = OpenFile (filepath);
% Type

type = char(f3.ReadString());
% Version

version = int32 (f3.ReadInt32());

if (strcmp (type, '% SDATA ') && 1 <= version && version <= 5)
% Number of Frequencies

nfreq = f3.ReadInt32();
% Number of Ports

nports = f3.ReadInt32();
% Init

d. Frequency = zeros(1, nfreq);
d. Ports = cell(1, nports);
d. PortZr = LinProp(zeros(1, nports));
d. FrequencyConversions = cell(1, nports);
d. Data = LinProp(zeros([nfreq, nports, nports]));
% Frequency (Hz)

for i = 1:nfreq
    d. Frequency(i) = f3.ReadDouble();
end
% Ports

ModeType = {'' , 'd', 'c'};
IndexType = {'' , ':'I', ':'II', ':'III', ':'IV', ':'V', ':'VI', ':'VII', ':'VIII', ':'IX', ':'X', ':'XI', ':'XII'};
```

Michael Wollensack Page 11 of 47 May 2022
for i = 1:nports
    if (version < 3)
        d.Ports{i} = num2str(f3.ReadInt32());
    else
        d.Ports{i} = [num2str(f3.ReadInt32()) ModeType{f3.ReadInt16() + 1} IndexType{f3.ReadInt16() + 1}];
    end
end

% Frequency Conversions
for i = 1:nports
    if (version < 4)
        d.FrequencyConversions{i}.TestReceiver.Numerator = 1;
        d.FrequencyConversions{i}.TestReceiver.Denominator = 1;
        d.FrequencyConversions{i}.TestReceiver.Offset = 0;
        d.FrequencyConversions{i}.ReferenceReceiver.Numerator = 1;
        d.FrequencyConversions{i}.ReferenceReceiver.Denominator = 1;
        d.FrequencyConversions{i}.ReferenceReceiver.Offset = 0;
        d.FrequencyConversions{i}.Source.Numerator = 1;
        d.FrequencyConversions{i}.Source.Denominator = 1;
        d.FrequencyConversions{i}.Source.Offset = 0;
    elseif (version == 4)
        numerator = f3.ReadDouble();
        denominator = f3.ReadDouble();
        offset = f3.ReadDouble();
        d.FrequencyConversions{i}.TestReceiver.Numerator = numerator;
        d.FrequencyConversions{i}.TestReceiver.Denominator = denominator;
        d.FrequencyConversions{i}.TestReceiver.Offset = offset;
        d.FrequencyConversions{i}.ReferenceReceiver.Numerator = numerator;
        d.FrequencyConversions{i}.ReferenceReceiver.Denominator = denominator;
        d.FrequencyConversions{i}.ReferenceReceiver.Offset = offset;
        d.FrequencyConversions{i}.Source.Numerator = numerator;
        d.FrequencyConversions{i}.Source.Denominator = denominator;
        d.FrequencyConversions{i}.Source.Offset = offset;
    else
        d.FrequencyConversions{i}.TestReceiver.Numerator = f3.ReadDouble();
        d.FrequencyConversions{i}.TestReceiver.Denominator = f3.ReadDouble();
        d.FrequencyConversions{i}.TestReceiver.Offset = f3.ReadDouble();
        d.FrequencyConversions{i}.ReferenceReceiver.Numerator = f3.ReadDouble();
        d.FrequencyConversions{i}.ReferenceReceiver.Denominator = f3.ReadDouble();
        d.FrequencyConversions{i}.ReferenceReceiver.Offset = f3.ReadDouble();
        d.FrequencyConversions{i}.Source.Numerator = f3.ReadDouble();
        d.FrequencyConversions{i}.Source.Denominator = f3.ReadDouble();
        d.FrequencyConversions{i}.Source.Offset = f3.ReadDouble();
    end
end
if (version == 1)
    % Port Zr
    for i = 1:nports
d. PortZr(i) = ReadComplexLinProp(f3);
end

% Data
for i1 = 1:nfreq
    for i2 = 1:nports
        for i3 = 1:nports
            d.Data(i1, i2, i3) = ReadComplexLinProp(f3);
        end
    end
end

elseif (2 <= version || version <= 5)
% Flat Vector
v = ReadComplexFlatVectorLinProp(f3);
index = 1;
% PortsZr
for i = 1:nports
    d.PortZr(i) = v(index); index = index + 1;
end
% Data
for i1 = 1:nfreq
    for i2 = 1:nports
        for i3 = 1:nports
            d.Data(i1, i2, i3) = v(index); index = index + 1;
        end
    end
end
f3.Close();
end

function f3 = OpenFile(filepath)
% Open File
NET.addAssembly('System');
% File Stream
f1 = System.IO.FileStream(filepath, System.IO.FileMode.Open);
% Try if Stream is GZIP compressed
try
    f2 = System.IO.Compression.GZipStream(f1, System.IO.Compression.CompressionMode.Decompress);
    f2.ReadByte();
    f1.Position = 0;
    f2 = System.IO.Compression.GZipStream(f1, System.IO.Compression.CompressionMode.Decompress);
    disp('GZIP compressed file')
catch
    f1.Position = 0;
    f2 = f1;
    disp('Uncompressed file')
end
% Binary Reader
f3 = System.IO.BinaryReader(f2);
end

function c = ReadComplexLinProp(f3)
% Read ComplexLinProp using METAS UncLib
n = NET.createGeneric('Metas.UncLib.Core.Complex',
   {'Metas.UncLib.LinProp.UncNumber'});

n.BinarySetDataFrom(f3);
c = LinProp(n);
end

function v = ReadComplexFlatVectorLinProp(f3)

% Read ComplexFlatVectorLinProp using METAS UncLib
list = Metas.UncLib.LinProp.UncList();
list.BinarySetDataFrom(f3);
   {'Metas.UncLib.LinProp.UncNumber'});
n.Init1dData(list.data);
r = LinProp(n);

v = r(1:2:end-1) + 1i.*r(2:2:end);
end
3  sdatx File Specification

The sdatx-file format is an XML file type developed by METAS. It’s described using an XML schema. See https://www.w3schools.com/xml/schema_intro.asp for more details about XML schemas. The file can be written in a GZIP file stream to reduce the file size.

3.1  XML Schema

The following listing shows the XML schema for ‘SParamData’:

```xml
<!-- definition of SParamData -->
<xs:element name="SParamData">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="FrequencyList" />
      <xs:element ref="PortList" />
      <xs:element ref="PortZrList" />
      <xs:element ref="FrequencyConversionList" minOccurs="0" maxOccurs="1" />
      <xs:element name="Data">
        <xs:complexType>
          <!-- Index 0: Frequency -->
          <xs:sequence>
            <xs:element maxOccurs="unbounded" name="Frequency">
              <xs:complexType>
                <!-- Index 1: Receiver Port -->
                <xs:sequence>
                  <xs:element maxOccurs="unbounded" name="ReceiverPort">
                    <xs:complexType>
                      <!-- Index 2: Source Port -->
                      <xs:sequence>
                        <xs:element maxOccurs="unbounded" name="SourcePort" type="ComplexUncNumberType" />
                      </xs:sequence>
                    </xs:complexType>
                  </xs:sequence>
                </xs:complexType>
              </xs:sequence>
            </xs:complexType>
          </xs:sequence>
        </xs:complexType>
      </xs:sequence>
    </xs:complexType>
  </xs:sequence>
</xs:element>
```

3.1.1  Frequency and Port Lists

The following listing shows the XML schema for ‘FrequencyList’:

```xml
<!-- definition of FrequencyList -->
<xs:element name="FrequencyList">
  <xs:complexType>
    <xs:sequence>
      <xs:element>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
The following listing shows the XML schema for ‘PortList’:

```
<xs:element name="PortList">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="Port" type="VnaPortDescriptionType" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The following listing shows the XML schema for ‘VnaPortDescriptionType’:

```
<xs:simpleType name="VnaPortDescriptionType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]+[sdc]?[M]{0,3}[C][MD]{0,3}(C[MD]|D?[CL]{0,3})(X[CL]|L?X{0,3})(I[XV]|V[I]{0,3})" />
  </xs:restriction>
</xs:simpleType>
```

The following listing shows the XML schema for ‘PortZrList’:

```
<xs:element name="PortZrList">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="PortZr" type="ComplexUncNumberType" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The following listing shows the XML schema for ‘FrequencyConversionList’:

```
<xs:element name="FrequencyConversionList">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="FrequencyConversion" type="FrequencyConversionType" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The following listing shows the XML schema for ‘FrequencyConversionType’:

```
<xs:complexType name="FrequencyConversionType">
  <xs:sequence>
    <xs:element name="TestReceiver" type="FrequencyConversionSubType" />
    <xs:element name="ReferenceReceiver" type="FrequencyConversionSubType" />
    <xs:element name="Source" type="FrequencyConversionSubType" />
  </xs:sequence>
</xs:complexType>
```
The following listing shows the XML schema for ‘FrequencyConversionSubType’:

```
<!-- definition of FrequencyConversionSubType -->
<xs: complexType name ="FrequencyConversionSubType">
  <xs: sequence>
    <xs: element name ="Numerator" type ="xs:double" />
    <xs: element name ="Denominator" type ="xs:double" />
    <xs: element name ="Offset" type ="xs:double" />
  </xs: sequence>
</xs: complexType>
```

### 3.1.2 Uncertainty Numbers

The following listing shows the XML schema for ‘ComplexUncNumberType’:

```
<!-- definition of ComplexUncNumberType -->
<xs: complexType name ="ComplexUncNumberType">
  <xs: sequence>
    <xs: element name ="Real" type ="UncNumberType" />
    <xs: element name ="Imag" type ="UncNumberType" />
  </xs: sequence>
</xs: complexType>
```

The following listing shows the XML schema for ‘UncNumberType’:

```
<!-- definition of UncNumberType -->
<xs: complexType name ="UncNumberType">
  <xs: sequence>
    <xs: element name ="Value" type ="xs:double" />
    <xs: element name ="Dependencies">
      <xs: complexType>
        <xs: sequence>
          <xs: element minOccurs ="0" maxOccurs ="unbounded" name ="DependsOn">
            <xs: complexType>
              <xs: sequence>
                <xs: element name ="Input">
                  <xs: complexType>
                    <xs: sequence>
                      <xs: element name ="Id" type ="UncInputIdType" />
                      <xs: element name ="Description" type ="xs:string" />
                      <xs: element name ="IDof" type ="xs:double" />
                    </xs: sequence>
                  </xs: complexType>
                </xs: element>
              </xs: sequence>
            </xs: complexType>
          </xs: element>
        </xs: sequence>
      </xs: complexType>
    </xs: element>
    <xs: element name ="Jacobi" type ="xs:double" />
  </xs: sequence>
</xs: complexType>
```

The following listing shows the XML schema for ‘UncInputIdType’:

```
<!-- definition of UncInputIdType -->
<xs: simpleType name ="UncInputIdType">
</xs: simpleType>
```
<xs:restriction base="xs:string">
  <xs:pattern value="([0-9a-fA-F][0-9a-fA-F][\-])?"/>
</xs:restriction>
</xs:simpleType>
4 sdatcv File Specification

The sdatcv-file format is an ASCII text file type developed by METAS. sdatcv-files consist of a header block followed by one or more sets of S-parameter data. For each frequency there is one set of data. It contains the values of the S-parameters and their covariance matrix. There are some general rules for sdatcv-files:

1. sdatcv-files contain only ASCII characters and the evaluation of sdatcv-files is case-insensitive.
2. Individual entries in a header or data line are separated by tabulator.
3. Header and data lines are terminated by a newline character (CR or CR/LF combination).
4. The decimal symbol for floating point values is the point (.) and not the comma (,), e.g.: 1.234567e-08. Note that digit-grouping symbols like (') are not allowed.
5. By convention, sdatcv-filenames use the file extension ‘sdatcv’.

4.1 Header Lines

Each sdatcv-file must contain a header block. The header block is formatted as follows:

```
1 SDATCV
2 Ports
3 1
4 Zr[1]re  Zr[1]im
5 50.0  0.0
```

Here the first header line defines that it is a sdatcv-file. The other five header lines are described in the following subsections.

4.1.1 Port Assignment

The keyword ‘Ports’ in header line 2 initiates the port assignment. Header line 3 describes the used ports by a list of VNA port descriptions. A VNA port description consists of an integer port number and an optional letter which describes the port-mode. No letter or ‘s’ denotes single-ended, ‘d’ is differential mode and ‘c’ is common mode.

4.1.2 Reference Impedance

The reference impedance is described in header lines 4 and 5. For each port the reference impedance in Ohm is formatted as a pair of values (real-imaginary).

4.1.3 Data Column Description

Header line 6 describes the data columns. The first column is the frequency column followed by the S-parameter data columns. These are formatted as pairs of values (real-imaginary).
After the S-parameter columns follow the covariance columns. They are as well formatted as pairs of values. It’s possible to specify only certain parts of the covariance matrix. For completing partially given covariance matrices, it’s assumed that the matrix is symmetric. Values which can not be deduced from symmetry are set to zero. The following table describes the order of the S-parameters of a \( n \)-port in the covariance matrix:

### Table 2: Order of the S-parameters in the covariance matrix

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Index</th>
<th>Parameter</th>
<th>Index</th>
<th>\ldots</th>
<th>Parameter</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_{1,1}^{re} )</td>
<td>1</td>
<td>( S_{1,2}^{re} )</td>
<td>( 2n + 1 )</td>
<td>\ldots</td>
<td>( S_{1,n}^{re} )</td>
<td>( 2n^2 - 2n + 1 )</td>
</tr>
<tr>
<td>( S_{1,1}^{im} )</td>
<td>2</td>
<td>( S_{1,2}^{im} )</td>
<td>( 2n + 2 )</td>
<td>\ldots</td>
<td>( S_{1,n}^{im} )</td>
<td>( 2n^2 - 2n + 2 )</td>
</tr>
<tr>
<td>( S_{2,1}^{re} )</td>
<td>3</td>
<td>( S_{2,2}^{re} )</td>
<td>( 2n + 3 )</td>
<td>\ldots</td>
<td>( S_{2,n}^{re} )</td>
<td>( 2n^2 - 2n + 3 )</td>
</tr>
<tr>
<td>( S_{2,1}^{im} )</td>
<td>4</td>
<td>( S_{2,2}^{im} )</td>
<td>( 2n + 4 )</td>
<td>\ldots</td>
<td>( S_{2,n}^{im} )</td>
<td>( 2n^2 - 2n + 4 )</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>( S_{n,1}^{re} )</td>
<td>( 2n - 1 )</td>
<td>( S_{n,2}^{re} )</td>
<td>( 4n - 1 )</td>
<td>\ldots</td>
<td>( S_{n,n}^{re} )</td>
<td>( 2n^2 - 1 )</td>
</tr>
<tr>
<td>( S_{n,1}^{im} )</td>
<td>( 2n )</td>
<td>( S_{n,2}^{im} )</td>
<td>( 4n )</td>
<td>\ldots</td>
<td>( S_{n,n}^{im} )</td>
<td>( 2n^2 )</td>
</tr>
</tbody>
</table>

E.g., the covariance of the real part of \( S_{1,1} \) and the imaginary part of \( S_{2,2} \) would be \( CV[1,8] \) for a two port device.

### 4.2 Data Lines

After the header lines follow the data sets. They contain the S-parameter data. Each data set starts with the frequency in Hz and ends with a newline character (CR or CR/LF combination). After the frequency follow the S-parameter and covariance data. These are formatted as pairs of values (real-imaginary). Each data set has to have as many entries as defined in the data column description. The data set have to be arranged in increasing order of frequency.

#### 4.2.1 1-Port Example

The following example shows a sdatcv-file of a 1-port with a complete covariance matrix (correlation between real and imaginary parts):

```plaintext
SDATCV
1 Ports
1 Zr[1]re Zr[1]im
50.0 0.0
1.00e+9 -9.16e-1 3.91e-1 1.39e-6 3.56e-7 3.56e-7 2.05e-6
2.00e+9 -6.90e-1 7.17e-1 1.98e-6 2.47e-7 2.47e-7 1.96e-6
3.00e+9 -3.55e-1 9.29e-1 2.58e-6 3.88e-7 3.88e-7 1.74e-6
```

\( CV[1,1] \) is the variance of the real part of \( S_{1,1} \).

\( CV[2,1] \) and \( CV[1,2] \) describe the covariance between the real and imaginary parts of \( S_{1,1} \).

\( CV[2,2] \) is the variance of the imaginary part of \( S_{1,1} \).
### 4.2.2 2-Port Examples

The following example shows a sdatcv-file of a 2-port with a reduced covariance matrix (correlation between real and imaginary parts, but no correlation between different S-parameters):

```
SDATCV
Ports
1
2
50.0 0.0 50.0 0.0
Freq
1.00e+9 -3.72e -3 5.39e -3 2.35e -1 -2.13e -1 2.35e -1 -2.14e -1 -3.90e -3 6.39e -3 8.00e -8 -1.32e -9 7.86e -8 4.48e -8 2.69e -8 2.69e -8 4.98e -8 4.50e -8 2.70e -8 5.00e -8 8.46e -8 4.22e -11 4.22e -11 8.55e -8
2.00e+9 -4.99e -4 9.12e -3 3.05e -2 -3.15e -1 3.05e -2 -3.15e -1 1.82e -3 8.80e -3 8.14e -8 -5.05e -10 7.97e -8 6.69e -8 4.46e -9 4.46e -9 2.12e -8 6.74e -8 4.38e -9 4.38e -9 2.15e -8 8.06e -8 9.99e -10 9.99e -10 8.25e -8
3.00e+9 3.81e -3 1.16e -2 -1.89e -1 -2.54e -1 -1.89e -1 -2.54e -1 7.37e -3 7.74e -3 1.46e -7 6.52e -10 1.45e -7 4.72e -8 -1.88e -8 4.72e -8 -1.89e -8 3.59e -8 1.51e -7 -7.87e -10 -7.87e -10 1.51e -7
```

- **CV[1,1]** is the variance of the real part of $S_{1,1}$.
- **CV[2,1]** and **CV[1,2]** describe the covariance between the real and imaginary parts of $S_{1,1}$.
- **CV[2,2]** is the variance of the imaginary part of $S_{1,1}$.
- **CV[3,3]** is the variance of the real part of $S_{2,1}$.
- **CV[4,3]** and **CV[3,4]** describe the covariance between the real and imaginary parts of $S_{2,1}$.
- **CV[4,4]** is the variance of the imaginary part of $S_{2,1}$.
- **CV[5,5]** is the variance of the real part of $S_{1,2}$.
- **CV[6,5]** and **CV[5,6]** describe the covariance between the real and imaginary parts of $S_{1,2}$.
- **CV[6,6]** is the variance of the imaginary part of $S_{1,2}$.
- **CV[7,7]** is the variance of the real part of $S_{2,2}$.
- **CV[8,7]** and **CV[7,8]** describe the covariance between the real and imaginary parts of $S_{2,2}$.
- **CV[8,8]** is the variance of the imaginary part of $S_{2,2}$.

The following example shows a sdatcv-file of a 2-port with a complete covariance matrix (correlation between real and imaginary parts of all S-parameters):
4.3 Comment Lines

One can add comments to a sdatcv-file. Comments are always preceded by a percent sign (%). A comment can be the only entry on a line or can follow the data on any line.
5 Touchstone V1.x snp File Specification

The Touchstone snp-file format is an ASCII text file type developed by the EIA/IBIS Open Forum. For the Touchstone snp file specification see https://ibis.org/connector/touchstone_spec11.pdf.

5.1 Examples

The following example shows a Touchstone s1p-file of a 1-port:

```
1 # Hz S RI R 50.0
2 1.00e+9 -9.16e-1 3.91e-1
3 2.00e+9 -6.90e-1 7.17e-1
4 3.00e+9 -3.55e-1 9.29e-1
```

The following example shows a Touchstone s2p-file of a 2-port:

```
1 # Hz S RI R 50.0
2 1.00e+9 -3.72e-3 5.39e-3 2.35e-1 -2.13e-1 2.35e-1 -2.14e-1 -3.90e-3 6.39e-3
3 2.00e+9 -4.99e-4 9.12e-3 3.05e-2 -3.15e-1 3.05e-2 -3.15e-1 1.82e-3 8.80e-3
4 3.00e+9 3.81e-3 1.16e-2 -1.89e-1 -2.54e-1 -1.89e-1 -2.54e-1 7.37e-3 7.74e-3
```
6 Touchstone V2.0 ts File Specification

The Touchstone ts-file format is an ASCII text file type developed by the EIA/IBIS Open Forum. For the Touchstone ts file specification see https://ibis.org/touchstone_ver2.0/touchstone_ver2.0.pdf.

6.1 Examples

The following example shows a Touchstone ts-file of a 1-port:

```
[Version] 2.0
# Hz S RI R 50.0
! Metas.Vna.Tools, 2.1.6907.29753
! Metas.Vna.Data, 2.1.6907.29469
! Created: UTC 2018.11.30 07:25:08
[Number of Ports] 1
[Number of Frequencies] 3
[Reference]
50.0
[Network Data]
! FREQ re:S1,1 im:S1,1
1.00e+9  -9.16e-1  3.91e-1
2.00e+9  -6.90e-1  7.17e-1
3.00e+9  -3.55e-1  9.29e-1
[End]
```

The following example shows a Touchstone ts-file of a 2-port:

```
[Version] 2.0
# Hz S RI R 50.0
! Metas.Vna.Tools, 2.1.6907.29753
! Metas.Vna.Data, 2.1.6907.29469
! Created: UTC 2018.11.30 07:25:20
[Number of Ports] 2
[Two-Port Data Order] 21_12
[Number of Frequencies] 3
[Reference]
50.0 50.0
[Network Data]
! FREQ re:S1,1 im:S1,1 re:S2,1 im:S2,1 re:S1,2 im:S1,2 re:S2,2 im:S2,2
1.00e+9  -3.72e-3  5.39e-3  2.35e-1  -2.13e-1  2.35e-1  -2.14e-1
2.00e+9  -4.99e-3  9.12e-3  3.05e-2  -3.15e-1  3.05e-2  -3.15e-1
3.00e+9  -3.81e-3  1.16e-2  -1.89e-1  -2.54e-1  -1.89e-1  -2.54e-1
[End]
```
7 vdatb File Specification

The vdatb-file format is a binary file type developed by METAS. The file can be written in a GZIP file stream to reduce the file size. The byte ordering is little-endian.

7.1 Binary Structure Version 1

Version 1 of vdatb uses a GZIP file stream to reduce the file size. The following enumeration describes the binary structure of a vdatb-file:

1. Header (string), value: ‘%VDATA’
2. Version (int32), value: 1
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Number of Parameters (int32)
6. Frequency List (double[]), size: number of frequencies
7. Ports (int32[]), size: number of ports
8. Port Impedance (ComplexUncNumber[]), size: number of ports
9. Data (VnaParameterData[]), size: number of parameters

For the definition of ‘ComplexUncNumber’ see section 2.1.1.

7.1.1 VNA Parameter Data

The following enumeration describes the binary structure of ‘VnaParameterData’:

1. Parameter (VnaParameter)
2. Data (ComplexUncNumber[]), size: number of frequencies

7.1.2 VNA Parameter

The following enumeration describes the binary structure of ‘VnaParameter’:

1. Numerator Receiver (ReceiverType)
2. Numerator Port (int32)
3. Denominator Receiver (ReceiverType)
4. Denominator Port (int32)
5. Source Port (int32)

‘ReceiverType’ is an enumeration represented by an integer where 0 is ‘1’, 1 is the test receiver ‘b’ and 2 is the reference receiver ‘a’.
7.1.3 VNA Parameter Examples

Table 3 shows some examples of VNA Parameters. Where W1 is the switch term of port 1

<table>
<thead>
<tr>
<th>Numerator Receiver</th>
<th>S11</th>
<th>S21</th>
<th>S12</th>
<th>S22</th>
<th>W1</th>
<th>W2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerator Port</td>
<td>'b'</td>
<td>'b'</td>
<td>'b'</td>
<td>'b'</td>
<td>'a'</td>
<td>'a'</td>
</tr>
<tr>
<td>Denominator Receiver</td>
<td>'a'</td>
<td>'a'</td>
<td>'a'</td>
<td>'a'</td>
<td>'b'</td>
<td>'b'</td>
</tr>
<tr>
<td>Denominator Port</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Source Port</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

and W2 is the switch term of port 2

7.2 Binary Structure Version 2

Version 2 of vdatb is not using a GZIP file stream. The redundancy of the data is removed by storing uncertainty inputs in a look up table. The uncompressed file size of version 2 is comparable to the GZIP compressed file size of version 1 and about four times smaller than the uncompressed file size of version 1. Avoiding the GZIP step and reducing the uncompressed file size speeds up loading and saving of files. The following enumeration describes the binary structure of a vdatb-file:

1. Header (string), value: '%VDATA'
2. Version (int32), value: 2
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Number of Parameters (int32)
6. Frequency List (double[]), size: number of frequencies
7. Ports (int32[]), size: number of ports
8. VNA Parameters (VnaParameter[]), size: number of parameters
9. Flat Vector (UncNumber[]), size: $2 \cdot N_{Ports} + 2 \cdot N_{Parameters} \cdot N_{Freq}$

The last element contains the port impedance and the data mapped to a flat vector of uncertainty numbers.

For the definition of ‘VnaParameter’ see section 7.1.2. For the definition of ‘Flat Vector of Uncertainty Numbers’ see section 2.2.1.
7.3 Binary Structure Version 3

Version 3 of vdatb is an extension to version 2 which adds support for mixed-mode parameters and port indices. The following enumeration describes the binary structure of a vdatb-file:

1. Header (string), value: ‘%VDATA’
2. Version (int32), value: 3
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Number of Parameters (int32)
6. Frequency List (double[]), size: number of frequencies
7. Ports (VnaPortDescription[]), size: number of ports
8. VNA Parameters (VnaParameter[]), size: number of parameters
9. Flat Vector (UncNumber[]), size: \(2N_{Ports} + 2N_{Parameters}N_{Freq}\)

The last element contains the port impedance and the data mapped to a flat vector of uncertainty numbers.

For the definition of ‘VnaPortDescription’ see section 2.3.1. For the definition of ‘VnaParameter’ see section 7.1.2. For the definition of ‘Flat Vector of Uncertainty Numbers’ see section 2.2.1.

7.4 Binary Structure Version 4

Version 4 of vdatb is an extension to version 3 which adds support for frequency converting parameters. The following enumeration describes the binary structure of a vdatb-file:

1. Header (string), value: ‘%VDATA’
2. Version (int32), value: 4
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Number of Parameters (int32)
6. Frequency List (double[]), size: number of frequencies
7. Ports (VnaPortDescription[]), size: number of ports
8. Frequency Conversion List (FrequencyConversionSub[]), size: number of ports
9. VNA Parameters (VnaParameter[]), size: number of parameters
10. Flat Vector (UncNumber[]), size: \(2N_{Ports} + 2N_{Parameters}N_{Freq}\)
The last element contains the port impedance and the data mapped to a flat vector of uncertainty numbers.

For the definition of ‘VnaPortDescription’ see section 2.3.1. For the definition of ‘FrequencyConversionSub’ see section 2.4.1. For the definition of ‘VnaParameter’ see section 7.1.2. For the definition of ‘Flat Vector of Uncertainty Numbers’ see section 2.2.1.

7.5 Binary Structure Version 5

Version 5 of vdatb is an extension to version 4 which adds support for frequency converting parameters, where the receiver frequency is not equal to the source frequency. The following enumeration describes the binary structure of a vdatb-file:

1. Header (string), value: ‘%VDATA’
2. Version (int32), value: 5
3. Number of Frequencies (int32)
4. Number of Ports (int32)
5. Number of Parameters (int32)
6. Frequency List (double[]), size: number of frequencies
7. Ports (VnaPortDescription[]), size: number of ports
8. Frequency Conversion List (FrequencyConversion[]), size: number of ports
9. VNA Parameters (VnaParameter[]), size: number of parameters
10. Flat Vector (UncNumber[]), size: \(2N_{Ports} + 2N_{Parameters}N_{Freq}\)

The last element contains the port impedance and the data mapped to a flat vector of uncertainty numbers.

For the definition of ‘VnaPortDescription’ see section 2.3.1. For the definition of ‘FrequencyConversion’ see section 2.5.1. For the definition of ‘VnaParameter’ see section 7.1.2. For the definition of ‘Flat Vector of Uncertainty Numbers’ see section 2.2.1.

7.6 Example MATLAB Code

The following code shows how to load a vdatb-file in MATLAB:

```matlab
function d = LoadVDATB(filepath)
% Loads VNA Tools II VnaData (*.vdatb) file
% Michael Wollensack METAS - 22.04.2022

d = {};

f3 = OpenFile(filepath);
% Type
type = char(f3.ReadString());
% Version
version = int32(f3.ReadInt32());
disp(['Type: ' type ', Version: ' num2str(version)])
```
if (strcmp(type, '%VDATA') && 1 <= version && version <= 5)
% Number of Frequencies
nfreq = f3.ReadInt32();
% Number of Ports
nports = f3.ReadInt32();
% Number of Parameters
nparams = f3.ReadInt32();
% Init
d. Frequency = zeros(1, nfreq);
d. Ports = cell(1, nports);
d. PortZr = LinProp(zeros(1, nports));
d. FrequencyConversions = cell(1, nports);
d. VnaParameter = cell(1, nparams);
d. Data = LinProp(zeros(nfreq, nparams));
% Frequency (Hz)
for i = 1:nfreq
  d. Frequency(i) = f3.ReadDouble();
end
% Ports
for i = 1:nports
d. Ports(i) = ReadPort(f3, version);
end
% Frequency Conversions
for i = 1:nports
  if (version < 4)
    d. FrequencyConversions(i).TestReceiver.Numerator = 1;
    d. FrequencyConversions(i).TestReceiver.Denominator = 1;
    d. FrequencyConversions(i).ReferenceReceiver.Numerator = 1;
    d. FrequencyConversions(i).ReferenceReceiver.Denominator = 1;
    d. FrequencyConversions(i).Source.Numerator = 1;
    d. FrequencyConversions(i).Source.Denominator = 1;
  elseif (version == 4)
    numerator = f3.ReadDouble();
    denominator = f3.ReadDouble();
    offset = f3.ReadDouble();
    d. FrequencyConversions(i).TestReceiver.Numerator = numerator;
    d. FrequencyConversions(i).TestReceiver.Denominator = denominator;
    d. FrequencyConversions(i).ReferenceReceiver.Numerator = numerator;
    d. FrequencyConversions(i).ReferenceReceiver.Denominator = denominator;
    d. FrequencyConversions(i).Source.Numerator = numerator;
    d. FrequencyConversions(i).Source.Denominator = denominator;
  else
    d. FrequencyConversions(i).TestReceiver.Numerator = f3.ReadDouble();
    d. FrequencyConversions(i).TestReceiver.Denominator = f3.ReadDouble();
    d. FrequencyConversions(i).TestReceiver.Offset = f3.ReadDouble();
  end
end
d. FrequencyConversions[i].ReferenceReceiver.Numerator = 
f3. ReadDouble();
d. FrequencyConversions[i].ReferenceReceiver.Denominator = 
  f3. ReadDouble();
d. FrequencyConversions[i].ReferenceReceiver.Offset = 
  f3. ReadDouble();
d. FrequencyConversions[i].Source.Numerator = f3. ReadDouble();
d. FrequencyConversions[i].Source.Denominator = f3. ReadDouble();
d. FrequencyConversions[i].Source.Offset = f3. ReadDouble();
end
if (version == 1)
  % Port Zr
  for i = 1:nports
    d. PortZr(i) = ReadComplexLinProp(f3);
  end
  % VnaParameterData
  for i2 = 1:nparams
    % VnaParameter
    d. VnaParameter{i2} = ReadVnaParameter(f3, version);
    % Data
    for i1 = 1:nfreq
      d. Data(i1, i2) = ReadComplexLinProp(f3);
    end
  end
elseif (2 <= version || version <= 5)
  % VnaParameter
  for i2 = 1:nparams
    d. VnaParameter{i2} = ReadVnaParameter(f3, version);
    % Flat Vector
    v = ReadComplexFlatVectorLinProp(f3);
    index = 1;
    % PortsZr
    for i = 1:nports
      d. PortZr(i) = v(index); index = index + 1;
    end
    % Data
    for i2 = 1:nparams
      for i1 = 1:nfreq
        d. Data(i1, i2) = v(index); index = index + 1;
      end
    end
  end
end
f3. Close();
% Read VNA Parameter
RcvType = {1, 'b', 'a'};
p = {};
p.NumRec = RcvType{f3.ReadInt32()} + 1;
p.NumPort = ReadPort(f3, version);
p.DenRec = RcvType{f3.ReadInt32()} + 1;
p.DenPort = ReadPort(f3, version);
p.SrcPort = ReadPort(f3, version);
end

function p = ReadVnaParameter(f3, version)
% Read VNA Parameter
RcvType = {1, 'b', 'a'};
p = {};
p.NumRec = RcvType{f3.ReadInt32()} + 1;
p.NumPort = ReadPort(f3, version);
p.DenRec = RcvType{f3.ReadInt32()} + 1;
p.DenPort = ReadPort(f3, version);
p.SrcPort = ReadPort(f3, version);
end

function f3 = OpenFile(filepath)
% Open File
NET.addAssembly('System');
% File Stream
f1 = System.IO.FileStream(filepath, System.IO.FileMode.Open);
% Try if Stream is GZIP compressed
try
  f2 = System.IO.Compression.GZipStream(f1, System.IO.Compression.CompressionMode.Decompress);
  f2.ReadByte();
  f1.Position = 0;
  f2 = System.IO.Compression.GZipStream(f1, System.IO.Compression.CompressionMode.Decompress);
  disp('GZIP compressed file')
catch
  f1.Position = 0;
  f2 = f1;
  disp('Uncompressed file')
end
% Binary Reader
f3 = System.IO.BinaryReader(f2);
end

function c = ReadComplexLinProp(f3)
% Read ComplexLinProp using METAS UncLib
n = NET.createGeneric('Metas.UncLib.Core.Complex',
  {'Metas.UncLib.LinProp.UncNumber'});
n.BinarySetDataFrom(f3);
c = LinProp(n);
end

function v = ReadComplexFlatVectorLinProp(f3)
% Read ComplexFlatVectorLinProp using METAS UncLib
list = Metas.UncLib.LinProp.Unlist();
list_BINARYSetDataFrom(f3);
  {'Metas.UncLib.LinProp.UncNumber'});
n_INITIDATA(list.data);
r = LinProp(n);
v = r(1:2:end-1) + 1i.*r(2:2:end);
end
8 vdatx File Specification

The vdatx-file format is an XML file type developed by METAS. It’s described using an XML schema. See https://www.w3schools.com/xml/schema_intro.asp for more details about XML schemas. The file can be written in a GZIP file stream to reduce the file size.

8.1 XML Schema

The following listing shows the XML schema for ‘VnaData’:

```xml
<!-- definition of VnaData -->
<xs:element name="VnaData">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="FrequencyList" />
      <xs:element ref="PortList" />
      <xs:element ref="PortZrList" />
      <xs:element ref="FrequencyConversionList" minOccurs="0" maxOccurs="1" />
      <xs:element name="ParameterDataList">
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" name="ParameterData">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Parameter" type="VnaParameterType" />
                  <xs:element name="Data">
                    <xs:complexType>
                      <!-- Index: Frequency -->
                      <xs:sequence>
                        <xs:element maxOccurs="unbounded" name="Frequency" type="ComplexUncNumberType" />
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

For the definition of ‘FrequencyList’, ‘PortList’, ‘PortZrList’, ‘FrequencyConversionList’ and ‘ComplexUncNumberType’ see section 3.1.1 and 3.1.2.

8.1.1 VNA Parameter Description

The following listing shows the XML schema for ‘VnaParameterType’:

```xml
<!-- definition of VnaParameterType -->
<xs:complexType name="VnaParameterType">
  <xs:sequence>
    <xs:element name="Data">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Frequency">
            <xs:complexType>
              <xs:sequence>
                <xs:element type="ComplexUncNumberType" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```
The following listing shows the XML schema for 'ReceiverType':

```xml
<xs:element name="NumRcv" type="ReceiverType" />
<xs:element name="NumPort" type="VnaPortDescriptionType" />
<xs:element name="DenRcv" type="ReceiverType" />
<xs:element name="DenPort" type="VnaPortDescriptionType" />
<xs:element name="SrcPort" type="VnaPortDescriptionType" />
</xs:sequence>
</xs:complexType>
```

For the definition of 'VnaPortDescriptionType' see section 3.1.1.
9 vdatcv File Specification

The vdatcv-file format is an ASCII text file type developed by METAS. vdatcv-files consist of a header block followed by one or more sets of VNA parameter data. For each frequency there is one set of data. It contains the values of arbitrary receiver ratios or single receivers and their covariance matrix. There are some general rules for vdatcv-files:

1. vdatcv-files contain only ASCII characters and the evaluation of vdatcv-files is case-insensitive.
2. Individual entries in a header or data line are separated by tabulator.
3. Header and data lines are terminated by a newline character (CR or CR/LF combination).
4. The decimal symbol for floating point values is the point (.) and not the comma (,), e.g.: 1.234567e-08. Note that digit-grouping symbols like (') are not allowed.
5. By convention, vdatcv-filenames use the file extension ‘vdatcv’.

9.1 Header Lines

Each vdatcv-file must contain a header block. The header block is formatted as follows:

```
1 VDATCV
2 Ports
3 1
4 Zr [1] re Zr [1] im
5 50.0 0.0
6 Freq a1/b1,2 re a1/b1,2 im CV [1,1] CV [2,1] CV [2,2]
```

Here the first header line defines that it is a vdatcv-file. The other five header lines are described in the following subsections.

9.1.1 Port Assignment

The keyword ‘Ports’ in header line 2 initiates the port assignment. Header line 3 describes the used ports by a list of VNA port descriptions. A VNA port description consists of an integer port number and an optional letter which describes the port-mode. No letter or ‘s’ denotes single-ended, ‘d’ is differential mode and ‘c’ is common mode.

9.1.2 Reference Impedance

The reference impedance is described in header lines 4 and 5. For each port the reference impedance in Ohm is formatted as a pair of values (real-imaginary).

9.1.3 Data Column Description

Header line 6 describes the data columns. The first column is the frequency column followed by the VNA parameter data columns. These are formatted as pairs of values (real-imaginary). Table 4 describes some examples of valid VNA parameters.
Table 4: Examples of valid VNA parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘S[1,1]’</td>
<td>S-Parameter, reflection at port 1</td>
</tr>
<tr>
<td>‘S[2,1]’</td>
<td>S-Parameter, transmission from port 1 to port 2</td>
</tr>
<tr>
<td>‘b1,1’</td>
<td>Test receiver of port 1 when the source is switched to port 2</td>
</tr>
<tr>
<td>‘b2,1’</td>
<td>Test receiver of port 2 when the source is switched to port 1</td>
</tr>
<tr>
<td>‘a1,1’</td>
<td>Reference receiver of port 1 when the source is switched to port 1</td>
</tr>
<tr>
<td>‘a1/b1,2’</td>
<td>Ratio of reference to test receivers of port 1 when the source is switched to port 2</td>
</tr>
</tbody>
</table>

After the parameter columns follow the covariance columns. They are as well formatted as pairs of values. It’s possible to specify only certain parts of the covariance matrix. For completing partially given covariance matrices, it’s assumed that the matrix is symmetric. Values which can not be deduced from symmetry are set to zero.

9.2 Data Lines

After the header lines follow the data sets. They contain the parameter data. Each data set starts with the frequency in Hz and ends with a newline character (CR or CR/LF combination). After the frequency follow the parameter and covariance data.

These are formatted as pairs of values (real-imaginary). Each data set has to have as many entries as defined in the data column description. The data set have to be arranged in increasing order of frequency.

9.2.1 Example

The following example shows a vdatcv-file with one parameter and its covariance:

```
VDATCV
Ports
1
Zr[1] re Zr[1] im
50.0 0.0
Freq a1/b1,2 re a1/b1,2 im CV[1,1] CV[2,1] CV[2,2]
1.00e+9 -9.16e-2 3.91e-2 1.39e-6 3.56e-7 2.05e-6
2.00e+9 -6.90e-2 7.17e-2 1.98e-6 2.47e-7 1.96e-6
3.00e+9 -3.55e-2 9.29e-2 2.58e-6 3.88e-7 1.74e-6
```

CV[1,1] is the variance of the real part of the receiver ratio $\frac{a1}{b1}$ when the source is switched to port 2.

CV[2,1] describe the covariance between the real and imaginary parts of the receiver ratio $\frac{a1}{b1}$ when the source is switched to port 2.

CV[2,2] is the variance of the imaginary part of the receiver ratio $\frac{a1}{b1}$ when the source is switched to port 2.
9.3 Comment Lines

One can add comments to a vdatcv-file. Comments are always preceded by a percent sign (%). A comment can be the only entry on a line or can follow the data on any line.
10 Citi File Specification

The Citi-file format is an ASCII text file type developed by Agilent. For the Citi file specification see:

- https://na.support.keysight.com/vna/help/latest/S5_Output/SaveRecall.htm#cti

10.1 Examples

The following example shows a Citi-file of a 1-port:

```
1 CITIFILE A.01.01
2 NAME DATA
3 VAR FREQ MAG 3
4 DATA S[1,1] RI
5 DATA U[1,1] RI
6 VAR_LIST_BEGIN
7 1.0000000000e+009
8 2.0000000000e+009
9 3.0000000000e+009
10 VAR_LIST_END
11 BEGIN
12 -9.1600000000e-001 ,3.9100000000e-001
13 -6.9000000000e-001 ,7.1700000000e-001
14 -3.5500000000e-001 ,9.2900000000e-001
15 END
```

The following example shows a Citi-file of a 2-port:

```
1 CITIFILE A.01.01
2 NAME DATA
3 VAR FREQ MAG 3
4 DATA S[1,1] RI
5 DATA U[1,1] RI
6 DATA S[2,1] RI
7 DATA U[2,1] RI
8 DATA S[1,2] RI
9 DATA U[1,2] RI
10 DATA S[2,2] RI
11 DATA U[2,2] RI
12 VAR_LIST_BEGIN
13 1.0000000000e+009
14 2.0000000000e+009
15 3.0000000000e+009
16 VAR_LIST_END
17 BEGIN
```
-3.7200000000e-003, 5.3900000000e-003
-4.9900000000e-004, 9.1200000000e-003
3.8100000000e-003, 1.1600000000e-002
END
BEGIN
5.6568542495e-004, 5.6071380159e-004
5.706136532e-004, 5.6462376854e-004
7.6419847665e-004, 7.6157671523e-004
END
BEGIN
2.3500000000e-001, -2.1300000000e-001
3.0500000000e-002, -3.1500000000e-001
-1.8900000000e-001, -2.5400000000e-001
END
BEGIN
4.2332020977e-004, 4.4631815719e-004
5.1730068626e-004, 2.9120439557e-004
4.3451121965e-004, 3.7894590643e-004
END
BEGIN
2.3500000000e-001, -2.1400000000e-001
3.0500000000e-002, -3.1500000000e-001
-1.8900000000e-001, -2.5400000000e-001
END
BEGIN
4.2426406871e-004, 4.4721359550e-004
5.1923019943e-004, 2.9325756597e-004
4.3451121965e-004, 3.7894590643e-004
END
BEGIN
-3.9000000000e-003, 6.3900000000e-003
1.8200000000e-003, 8.8000000000e-003
7.3700000000e-003, 7.7400000000e-003
END
BEGIN
5.8172158289e-004, 5.8480766069e-004
5.6780278266e-004, 5.7445626465e-004
7.7717384603e-004, 7.7717423771e-004
END
11 scolb File Specification

The METAS scolb file format is a zip file which contains multiple sdatb files, see section 2.

12 scolcv File Specification

The scolcv-file format is an ASCII text file type developed by METAS. scolcv-files consist of a header block followed by one or more sets of S-parameter data of multiple standards. For each frequency there is one set of data. It contains the values of the S-parameters of multiple standards and their covariance matrix. There are some general rules for scolcv-files:

1. scolcv-files contain only ASCII characters and the evaluation of scolcv-files is case-insensitive.
2. Individual entries in a header or data line are separated by tabulator.
3. Header and data lines are terminated by a newline character (CR or CR/LF combination).
4. The decimal symbol for floating point values is the point (.) and not the comma (,), e.g.: 1.234567e-08. Note that digit-grouping symbols like (′) are not allowed.
5. By convention, scolcv-filenames use the file extension ‘scolcv’.

12.1 Header Lines

Each scolcv-file must contain a header block. The header block is formatted as follows:

```
SCOLCV
Number 1
Name Standard_01
Ports 1
Zr[1] re  Zr[1] im
  50.0  0.0
Number 2
Name Standard_02
Ports 1
Zr[1] re  Zr[1] im
  50.0  0.0
Freq 1:S[1,1] re  1:S[1,1] im  2:S[1,1] re
```

Here the first header line defines that it is a scolcv-file. The other header lines are described in the following subsections.
12.1.1 Number

The keyword ‘Number’ in header line $9(i − 1) + 3$ initiates the number. Header line $9(i − 1) + 4$ describes the current standard $i$.

12.1.2 Name

The name of the current standard is described in header lines $9(i − 1) + 5$ and $9(i − 1) + 6$.

12.1.3 Port Assignment

The keyword ‘Ports’ in header line $9(i − 1) + 7$ initiates the port assignment. Header line $9(i − 1) + 8$ describes the used ports of the current standard by a list of VNA port descriptions. A VNA port description consists of an integer port number and an optional letter which describes the port-mode. No letter or ‘s’ denotes single-ended, ‘d’ is differential mode and ‘c’ is common mode.

12.1.4 Reference Impedance

The reference impedance of the current standard is described in header lines $9(i − 1) + 9$ and $9(i − 1) + 10$. For each port the reference impedance in Ohm is formatted as a pair of values (real-imaginary).

12.1.5 Data Column Description

The last header line describes the data columns. The first column is the frequency column followed by the S-parameter data columns of all standards. These are formatted as pairs of values (real-imaginary).

After the S-parameter columns follow the covariance columns. They are as well formatted as pairs of values. It’s possible to specify only certain parts of the covariance matrix. For completing partially given covariance matrices, it’s assumed that the matrix is symmetric. Values which can not be deduced from symmetry are set to zero.

12.2 Data Lines

After the header lines follow the data sets. They contain the S-parameter data. Each data set starts with the frequency in Hz and ends with a newline character (CR or CR/LF combination). After the frequency follow the S-parameter and covariance data. These are formatted as pairs of values (real-imaginary). Each data set has to have as many entries as defined in the data column description. The data set have to be arranged in increasing order of frequency.

12.2.1 Example

The following example shows a scolcv-file of two 1-port standards and their covariance matrix:

```
1 SCOLCV
2 --------------------------------------------------------------------------------
3 Number
4 1
```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Standard_01</td>
<td></td>
<td>1</td>
<td>50.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Standard_02</td>
<td></td>
<td>1</td>
<td>50.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CV[1,1] is the variance of the real part of S₁₁ of standard 1.

CV[2,1] describe the covariance between the real and imaginary parts of S₁₁ of standard 1.

CV[3,1] describe the covariance between standards 1 and 2 of the real parts of S₁₁.

CV[4,1] describe the covariance between standard 1 real part and standard 2 imaginary part of S₁₁.

CV[2,2] is the variance of the imaginary part of S₁₁ of standard 1.

CV[3,2] describe the covariance between standard 1 imaginary part and standard 2 real part of S₁₁.

CV[4,2] describe the covariance between standards 1 and 2 of the imaginary parts of S₁₁.

CV[3,3] is the variance of the real part of S₁₁ of standard 2.

CV[4,3] describe the covariance between the real and imaginary parts of S₁₁ of standard 2.

CV[4,4] is the variance of the imaginary part of S₁₁ of standard 2.

### 12.3 Comment Lines

One can add comments to a scolcv-file. Comments are always preceded by a percent sign (%). A comment can be the only entry on a line or can follow the data on any line.
13 vcolb File Specification

The METAS vcolb file format is a zip file which contains multiple vdatb files, see section 7.

14 vcolcv File Specification

The vcolcv-file format is an ASCII text file type developed by METAS. vcolcv-files consist of a header block followed by one or more sets of VNA parameter data of multiple standards. For each frequency there is one set of data. It contains the values of arbitrary receiver ratios or single receivers of multiple standards and their covariance matrix. There are some general rules for vcolcv-files:

1. vcolcv-files contain only ASCII characters and the evaluation of vcolcv-files is case-insensitive.
2. Individual entries in a header or data line are separated by tabulator.
3. Header and data lines are terminated by a newline character (CR or CR/LF combination).
4. The decimal symbol for floating point values is the point (.) and not the comma (,), e.g.: 1.234567e-08. Note that digit-grouping symbols like (′) are not allowed.
5. By convention, vcolcv-filenames use the file extension ‘vcolcv’.

14.1 Header Lines

Each vcolcv-file must contain a header block. The header block is formatted as follows:

```
1 VCOLCV
2 --------------------------------------------------------------------------------
3 Number
4 1
5 Name
6 Standard_01
7 Ports
8 1
10 50.0  0.0
11 --------------------------------------------------------------------------------
12 Number
13 2
14 Name
15 Standard_02
16 Ports
17 1
18 Zr[1]re  Zr[1]im
19 50.0  0.0
20 --------------------------------------------------------------------------------
```
METAS VNA Tools - Data Formats V2.5.4

Here the first header line defines that it is a vcolcv-file. The other header lines are described in the following subsections.

14.1.1 Number

The keyword ‘Number’ in header line $9(i-1)+3$ initiates the number. Header line $9(i-1)+4$ describes the current standard $i$.

14.1.2 Name

The name of the current standard is described in header lines $9(i-1)+5$ and $9(i-1)+6$.

14.1.3 Port Assignment

The keyword ‘Ports’ in header line $9(i-1)+7$ initiates the port assignment. Header line $9(i-1)+8$ describes the used ports of the current standard by a list of VNA port descriptions. A VNA port description consists of an integer port number and an optional letter which describes the port-mode. No letter or ‘s’ denotes single-ended, ‘d’ is differential mode and ‘c’ is common mode.

14.1.4 Reference Impedance

The reference impedance of the current standard is described in header lines $9(i-1)+9$ and $9(i-1)+10$. For each port the reference impedance in Ohm is formatted as a pair of values (real-imaginary).

14.1.5 Data Column Description

The last header line describes the data columns. The first column is the frequency column followed by the VNA parameter data columns of all standards. These are formatted as pairs of values (real-imaginary). Table 5 describes some examples of valid VNA parameters for multiple standards.

After the parameter data columns follow the covariance columns. They are as well formatted as pairs of values. It’s possible to specify only certain parts of the covariance matrix. For completing partially given covariance matrices, it’s assumed that the matrix is symmetric. Values which can not be deduced from symmetry are set to zero.

14.2 Data Lines

After the header lines follow the data sets. They contain the parameter data. Each data set starts with the frequency in Hz and ends with a newline character (CR or CR/LF combination). After the frequency follow the parameter and covariance data. These are formatted as pairs of values (real-imaginary). Each data set has to have as many entries as defined in the data column description. The data set have to be arranged in increasing order of frequency.
Table 5: Examples of valid VNA parameters for multiple standards

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'1:S[1,1]'</td>
<td>S-Parameter, reflection at port 1 of standard 1</td>
</tr>
<tr>
<td>'2:S[1,1]'</td>
<td>S-Parameter, reflection at port 1 of standard 2</td>
</tr>
<tr>
<td>'1:S[2,1]'</td>
<td>S-Parameter, transmission from port 1 to port 2 of standard 1</td>
</tr>
<tr>
<td>'2:S[2,1]'</td>
<td>S-Parameter, transmission from port 1 to port 2 of standard 2</td>
</tr>
<tr>
<td>'1:b1,1'</td>
<td>Test receiver of port 1 when the source is switched to port 1 of standard 1</td>
</tr>
<tr>
<td>'2:b1,1'</td>
<td>Test receiver of port 1 when the source is switched to port 1 of standard 2</td>
</tr>
<tr>
<td>'1:b2,1'</td>
<td>Test receiver of port 2 when the source is switched to port 1 of standard 1</td>
</tr>
<tr>
<td>'2:b2,1'</td>
<td>Test receiver of port 2 when the source is switched to port 1 of standard 2</td>
</tr>
<tr>
<td>'1:a1,1'</td>
<td>Reference receiver of port 1 when the source is switched to port 1 of standard 1</td>
</tr>
<tr>
<td>'2:a1,1'</td>
<td>Reference receiver of port 1 when the source is switched to port 1 of standard 2</td>
</tr>
<tr>
<td>'1:a1/b1,2'</td>
<td>Ratio of reference to test receivers of port 1 when the source is switched to port 2 of standard 1</td>
</tr>
</tbody>
</table>

14.2.1 Example

The following example shows a vcolcv-file of two 1-port standards and their covariance matrix:

```
1 VCOLCV
2
3 Number  1
4 Name     Standard_01
5 Ports  1
6 Zr[1] re Zr[1] im
7   50.0  0.0

8 Number  2
9 Name     Standard_02
10 Ports  1
12   50.0  0.0

13 Freq   1:S[1,1] re 1:S[1,1] im 2:S[1,1] re 2:S[1,1] im
15 1.00E+09 -3.72E-03 5.39E-03 -3.90E-03 6.39E-03 8.00E-08 -1.32E-09 -2.13E-08 -4.74E-08 7.86E-08 4.47E-08 -2.42E-08 8.46E-08 4.22E-11 8.55E-08
16 2.00E+09 -4.99E-04 9.12E-03 1.82E-03 8.80E-03 8.14E-08 -5.05E-10 -5.13E-08 2.08E-10 7.97E-08 -4.38E-09 -5.22E-08 8.06E-08 9.99E-10 8.25E-08
17 3.00E+09 3.81E-03 1.16E-02 7.37E-03 7.74E-03 1.46E-07 6.50E-10 -4.75E-08 2.02E-08 1.45E-07 -2.38E-08 -5.19E-08 1.51E-07 -7.87E-10 1.51E-07
```
CV[1,1] is the variance of the real part of $S_{1,1}$ of standard 1.

CV[2,1] describe the covariance between the real and imaginary parts of $S_{1,1}$ of standard 1.

CV[3,1] describe the covariance between standards 1 and 2 of the real parts of $S_{1,1}$.

CV[4,1] describe the covariance between standard 1 real part and standard 2 imaginary part of $S_{1,1}$.

CV[2,2] is the variance of the imaginary part of $S_{1,1}$ of standard 1.

CV[3,2] describe the covariance between standard 1 imaginary part and standard 2 real part of $S_{1,1}$.

CV[4,2] describe the covariance between standards 1 and 2 of the imaginary parts of $S_{1,1}$.

CV[3,3] is the variance of the real part of $S_{1,1}$ of standard 2.

CV[4,3] describe the covariance between the real and imaginary parts of $S_{1,1}$ of standard 2.

CV[4,4] is the variance of the imaginary part of $S_{1,1}$ of standard 2.

14.3 Comment Lines

One can add comments to a vcolcv-file. Comments are always preceded by a percent sign (%). A comment can be the only entry on a line or can follow the data on any line.
A  ZIP File with Embedded Data Files

Table 6 shows the supported file types for embedded data files in a ZIP file.

<table>
<thead>
<tr>
<th>Description</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-Parameter Data Binary</td>
<td>.sdatb</td>
</tr>
<tr>
<td>S-Parameter Data Xml</td>
<td>.sdatx</td>
</tr>
<tr>
<td>S-Parameter Data Covariance Text</td>
<td>.sdatcv</td>
</tr>
<tr>
<td>S-Parameter Data Touchstone</td>
<td>.s*p:.ts</td>
</tr>
<tr>
<td>VNA Data Binary</td>
<td>.vdatb</td>
</tr>
<tr>
<td>VNA Data Xml</td>
<td>.vdatx</td>
</tr>
<tr>
<td>VNA Data Covariance Text</td>
<td>.vdatcv</td>
</tr>
<tr>
<td>VNA Data CITI</td>
<td>.cti:.citi</td>
</tr>
</tbody>
</table>

B  PDF File with Embedded Data Files

The supported file types for embedded data files in a PDF file are the same like for a ZIP file, see appendix A and table 6.

B.1  Example

The following \LaTeX-code generates an example of a PDF/A-3u with two embedded data files:

```
\documentclass{minimal}
\usepackage[a-3u]{pdfx}
\usepackage{embedfile}
\newcommand{\datafile}[2]
{\embedfile[ filespec={#2},
ucfilespec={#2},
filesystem=URL,
mimetype=application/octet-stream,
desc={#2},stringmethod=escape]{#1#2}}
\datafile{Collection/}{Standard_01.sdatb}
\datafile{Collection/}{Standard_02.sdatb}
\begin{document}
\end{document}
```

See the embedded data files ...
C  PTB DCC XML File with Embedded Data Files

The Digital Calibration Certificate (DCC) is an XML Schema Definition (XSD) developed by Physikalisch-Technische Bundesanstalt (PTB). For the PTB DCC development platform see https://gitlab.com/ptb/dcc/xsd-dcc.

The supported file types for embedded data files in a PTB DCC XML file are the same like for a ZIP file, see appendix A and table 6. The names of the embedded data files are stored under the following XPath:

```
1  /dcc:digitalCalibrationCertificate
2      /dcc:measurementResults
3          /dcc:measurementResult
4              /dcc:results
5                /dcc:result
6                    /dcc:data
7                        /dcc:byteData
8                            /dcc:filename
```

The following XQuery is used to access the embedded data file:

```
1  /dcc:digitalCalibrationCertificate
2      /dcc:measurementResults
3          /dcc:measurementResult
4              /dcc:results
5                /dcc:result
6                    /dcc:data
7                        /dcc:byteData
8                            /dcc:byteData
```

```
9                            /dcc:byteData
10                               [../dcc:filename = ‘NAME’]
```