Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Federal Institute of Metrology METAS

METAS VNA Tools - Data Formats V2.8.1

Michael Wollensack

September 2023

Contents

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 1: List of file formats

Format	Type	Port Assignment	Port Impedance	S-Parameter	Mixed-Mode P.	Freq. Conversion	Receiver Values	Arbitrary Ratios	Collection	Uncertainties	Correlation	Dependencies	ZIP Compression	No Redundancy	File Size	Speed
METAS sdatb V1	binary	☺	☺	☺						☺	☺	☺	☺		☺	☺
METAS sdatb V2	binary	☺	☺	☺						☺	☺	☺		☺	☺	☺☺
METAS sdatb V3	binary	☺	☺	☺	☺					☺	☺	☺		☺	☺	☺☺
METAS sdatb V4	binary	☺	☺	☺	☺	☺				☺	☺	☺		☺	☺	☺☺
METAS sdatb V5	binary	☺	☺	☺	☺	☺				☺	☺	☺		☺	☺	☺☺
METAS sdatx	xml	☺	☺	☺	☺	☺				☺	☺	☺	☺		☹	☹
METAS sdatcv	text	☺	☺	☺	☺					☺	☺	☹			☺	☺
Touchstone V1.x snp	text	☹	☺	☺						☹	☹	☹			☺	☺
Touchstone V2.0 ts	text	☺	☺	☺	☺					☹	☹	☹			☺	☺
METAS vdatb V1	binary	☺	☺	☺			☺	☺		☺	☺	☺	☺		☺	☺
METAS vdatb V2	binary	☺	☺	☺			☺	☺		☺	☺	☺		☺	☺	☺☺
METAS vdatb V3	binary	☺	☺	☺	☺		☺	☺		☺	☺	☺		☺	☺	☺☺
METAS vdatb V4	binary	☺	☺	☺	☺	☺	☺	☺		☺	☺	☺		☺	☺	☺☺
METAS vdatb V5	binary	☺	☺	☺	☺	☺	☺	☺		☺	☺	☺		☺	☺	☺☺
METAS vdatx	xml	☺	☺	☺	☺	☺	☺	☺		☺	☺	☺	☺		☹	☹
METAS vdatcv	text	☺	☺	☺	☺		☺	☺		☺	☺	☹			☺	☺
CITI	text	☺	☹	☺			☺	☺		☺	☹	☹			☺	☺
METAS scolb	zip	☺	☺	☺	☺	☺			☺	☺	☺	☺	☺		☺	☺
METAS scolcv	text	☺	☺	☺	☺				☺	☺	☺	☹			☺	☺
METAS vcolb	zip	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺		☺	☺
METAS vcolcv	text	☺	☺	☺	☺		☺	☺	☺	☺	☺	☹			☺	☺

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 version 1 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)

The following enumeration describes the binary structure version 2 of 'UncNumber':

1. Version (7-bit encoded int), value: 2
2. Value (double)
3. Number of Dependencies (7-bit encoded int)
4. Dependencies (DependsOn2[]), size: number of dependencies

The following enumeration describes the binary structure of 'DependsOn2':

1. Input (InputDistribution)
2. Jacobi (double)

2.1.2 Input Distribution

The following enumeration describes the binary structure of 'InputDistribution':

1. Version (7-bit encoded int), value: 2
2. Number of Id Bytes (7-bit encoded int)
3. Id (byte[]), size: number of id bytes
4. Description (string)
5. Distribution (Distribution)

2.1.3 Distributions

The following enumeration describes the binary structure of 'StandardNormalDistribution':

1. Type (7-bit encoded int), value: 0

The following enumeration describes the binary structure of 'NormalDistribution':

1. Type (7-bit encoded int), value: 1
2. mu (double)
3. sigma (double)

The following enumeration describes the binary structure of 'StandardUniformDistribution':

1. Type (7-bit encoded int), value: 2

The following enumeration describes the binary structure of 'UniformDistribution':

1. Type (7-bit encoded int), value: 3
2. a (double)
3. b (double)

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

1. Type (7-bit encoded int), value: 4
2. a (double)
3. b (double)
4. d (double)

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

1. Type (7-bit encoded int), value: 5
2. a (double)
3. b (double)
4. beta (double)

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

1. Type (7-bit encoded int), value: 6
2. a (double)
3. b (double)

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

1. Type (7-bit encoded int), value: 7
2. a (double)
3. b (double)

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

1. Type (7-bit encoded int), value: 8
2. a (double)
3. b (double)

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

1. Type (7-bit encoded int), value: 9
2. k (int32)

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

1. Type (7-bit encoded int), value: 10
2. mu (double)
3. sigma (double)
4. dof (double)

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

1. Type (7-bit encoded int), value: 11
2. Version (7-bit encoded int), value: 2
3. Number of Samples (7-bit encoded int)
4. Samples (double[]), size: number of samples

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

1. Type (7-bit encoded int), value: 99
2. Version (7-bit encoded int), value: 2
3. Number of Seed Bytes (7-bit encoded int)
4. Seed (byte[]), size: number of seed bytes
5. Number of Samples (7-bit encoded int)
6. Samples (double[]), size: number of samples

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: $2N_{Ports} + 2N_{Freq}N_{Ports}N_{Ports}$

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 version 1 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 version 2 of ‘FlatVectorUncNumbers’:

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

For the definition of ‘InputDistribution’ see section 2.1.2.

The following enumeration describes the binary structure of ‘UncInput’ which was used in version 1 of ‘FlatVectorUncNumbers’:

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: $2N_{Ports} + 2N_{Freq}N_{Ports}N_{Ports}$

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: $2N_{Ports} + 2N_{Freq}N_{Ports}N_{Ports}$

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: $2N_{Ports} + 2N_{Freq}N_{Ports}N_{Ports}$

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:

```
function d = LoadSDATB(filepath)
% Loads VNA Tools II SParamData (*.sdatb) 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, '%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'};
    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
    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);
n = NET.createGeneric('Metas.UncLib.Core.Ndims.RealNArray',
    {'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':

```
<!-- 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:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

3.1.1 Frequency and Port Lists

The following listing shows the XML schema for 'FrequencyList':

```
<!-- definition of FrequencyList -->
<xs:element name="FrequencyList">
  <xs:complexType>
    <xs:sequence>
```

```
        <xs:element maxOccurs="unbounded" name="Frequency"
            type="xs:double" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

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

```
  <!-- definition of 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’:

```
  <!-- definition of VnaPortDescriptionType -->
  <xs:simpleType name ="VnaPortDescriptionType">
    <xs:restriction base="xs:string">
      <xs:pattern
          value="[0-9]+[sdc]?:?M{0,3}(C[MD]|D?C{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’:

```
  <!-- definition of 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’:

```
  <!-- definition of 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’:

```
  <!-- definition of 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" type="InputDistributionType" />
                <xs:element name="Jacobi" type="xs:double" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

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

```
<!-- definition of InputDistributionType -->
<xs:complexType name="InputDistributionType">
  <xs:sequence>
    <xs:element name="Id" type="UncInputIdType" />
    <xs:element name="Description" type="xs:string" />
    <xs:choice>
      <xs:element name="Distribution" type="Distribution" />
      <!-- obsolete -->
      <xs:element name="IDof" type="xs:double" />
    </xs:choice>
    </xs:sequence>
  </xs:complexType>
```

The following listing shows the XML schema for the different distributions:

```
  <!-- definition of Distribution -->
  <xs:complexType name="Distribution" abstract="true"/>

  <!-- definition of StandardNormal -->
  <xs:complexType name="StandardNormal">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of Normal -->
  <xs:complexType name="Normal">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="mu" type="xs:double" />
          <xs:element name="sigma" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of StandardUniform -->
  <xs:complexType name="StandardUniform">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of Uniform -->
  <xs:complexType name="Uniform">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="a" type="xs:double" />
          <xs:element name="b" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of CurvilinearTrapezoid -->
  <xs:complexType name="CurvilinearTrapezoid">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="a" type="xs:double" />
          <xs:element name="b" type="xs:double" />
          <xs:element name="d" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of Trapezoidal -->
  <xs:complexType name="Trapezoidal">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="a" type="xs:double" />
          <xs:element name="b" type="xs:double" />
          <xs:element name="beta" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of Triangular -->
  <xs:complexType name="Triangular">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="a" type="xs:double" />
          <xs:element name="b" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of ArcSine -->
  <xs:complexType name="ArcSine">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="a" type="xs:double" />
          <xs:element name="b" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of Exponential -->
  <xs:complexType name="Exponential">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="mu" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of Gamma -->
  <xs:complexType name="Gamma">
```

```
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="a" type="xs:double" />
          <xs:element name="b" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of ChiSquared -->
  <xs:complexType name="ChiSquared">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="k" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of StudentT -->
  <xs:complexType name="StudentT">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="mu" type="xs:double" />
          <xs:element name="sigma" type="xs:double" />
          <xs:element name="dof" type="xs:double" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of StudentTFromSamples -->
  <xs:complexType name="StudentTFromSamples">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
          <xs:element name="Samples">
            <xs:complexType>
              <xs:sequence>
                <xs:element minOccurs="0" maxOccurs="unbounded"
                   name="Sample" type="xs:double" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- definition of RandomChoicesFromSamples -->
  <xs:complexType name="RandomChoicesFromSamples">
    <xs:complexContent>
      <xs:extension base="Distribution">
        <xs:sequence>
```

```
          <xs:element name="Seed" type="UncInputIdType" />
          <xs:element name="Samples">
            <xs:complexType>
              <xs:sequence>
                <xs:element minOccurs="0" maxOccurs="unbounded"
                  name="Sample" type="xs:double" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

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

```
<!-- definition of UncInputIdType -->
<xs:simpleType name="UncInputIdType">
  <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
6 Freq       S[1,1]re   S[1,1]im   CV[1,1]    CV[2,1]    CV[1,2]    CV[2,2]
```

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

Parameter	Index	Parameter	Index	$\cdots$	Parameter	Index
$S_{1,1\,re}$	1	$S_{1,2\,re}$	$2n+1$	$\cdots$	$S_{1,n\,re}$	$2n^2-2n+1$
$S_{1,1\,im}$	2	$S_{1,2\,im}$	$2n+2$	$\cdots$	$S_{1,n\,im}$	$2n^2-2n+2$
$S_{2,1\,re}$	3	$S_{2,2\,re}$	$2n+3$	$\cdots$	$S_{2,n\,re}$	$2n^2-2n+3$
$S_{2,1\,im}$	4	$S_{2,2\,im}$	$2n+4$	$\cdots$	$S_{2,n\,im}$	$2n^2-2n+4$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$S_{n,1\,re}$	$2n-1$	$S_{n,2\,re}$	$4n-1$	$\cdots$	$S_{n,n\,re}$	$2n^2-1$
$S_{n,1\,im}$	$2n$	$S_{n,2\,im}$	$4n$	$\cdots$	$S_{n,n\,im}$	$2n^2$

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):

```
SDATCV
Ports
1
Zr[1]re   Zr[1]im
50.0      0.0
Freq      S[1,1]re  S[1,1]im  CV[1,1]   CV[2,1]   CV[1,2]   CV[2,2]
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
Zr[1]re   Zr[1]im   Zr[2]re   Zr[2]im
50.0      0.0       50.0      0.0
Freq      S[1,1]re  S[1,1]im  S[2,1]re  S[2,1]im  S[1,2]re  S[1,2]im
    S[2,2]re  S[2,2]im  CV[1,1]   CV[2,1]   CV[2,2]   CV[3,3]   CV[4,3]
    CV[3,4]   CV[4,4]   CV[5,5]   CV[6,5]   CV[5,6]   CV[6,6]   CV[7,7]
    CV[8,7]   CV[7,8]   CV[8,8]
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   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
    -1.88e-8  3.59e-8   4.72e-8   -1.89e-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):

```
SDATCV
Ports
1         2
Zr[1]re   Zr[1]im   Zr[2]re   Zr[2]im
50.0      0.0       50.0      0.0
Freq      S[1,1]re  S[1,1]im  S[2,1]re  S[2,1]im  S[1,2]re  S[1,2]im
    S[2,2]re  S[2,2]im  CV[1,1]   CV[2,1]   CV[3,1]   CV[4,1]   CV[5,1]
    CV[6,1]   CV[7,1]   CV[8,1]   CV[1,2]   CV[2,2]   CV[3,2]   CV[4,2]
    CV[5,2]   CV[6,2]   CV[7,2]   CV[8,2]   CV[1,3]   CV[2,3]   CV[3,3]
    CV[4,3]   CV[5,3]   CV[6,3]   CV[7,3]   CV[8,3]   CV[1,4]   CV[2,4]
    CV[3,4]   CV[4,4]   CV[5,4]   CV[6,4]   CV[7,4]   CV[8,4]   CV[1,5]
    CV[2,5]   CV[3,5]   CV[4,5]   CV[5,5]   CV[6,5]   CV[7,5]   CV[8,5]
    CV[1,6]   CV[2,6]   CV[3,6]   CV[4,6]   CV[5,6]   CV[6,6]   CV[7,6]
    CV[8,6]   CV[1,7]   CV[2,7]   CV[3,7]   CV[4,7]   CV[5,7]   CV[6,7]
    CV[7,7]   CV[8,7]   CV[1,8]   CV[2,8]   CV[3,8]   CV[4,8]   CV[5,8]
    CV[6,8]   CV[7,8]   CV[8,8]
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  -9.15e-10 -2.38e-10 -1.30e-9
    5.48e-11  -2.13e-8  -4.74e-8  -1.32e-9  7.86e-8   -1.66e-9  -2.15e-9
    -1.91e-9  -2.48e-9  4.47e-8   -2.42e-8  -9.15e-10 -1.66e-9  4.48e-8
    2.69e-8   3.45e-8   2.79e-8   -1.49e-10 -7.61e-9  -2.38e-10 -2.15e-9
    2.69e-8   4.98e-8   2.80e-8   3.97e-8   3.21e-9   -7.84e-9  -1.30e-9
    -1.91e-9  3.45e-8   2.80e-8   4.50e-8   2.70e-8   5.68e-10  -7.06e-9
    5.48e-11  -2.48e-9  2.79e-8   3.97e-8   2.70e-8   5.00e-8   2.55e-9
    -7.22e-9  -2.13e-8  4.47e-8   -1.49e-10 3.21e-9   5.68e-10  2.55e-9
    8.46e-8   4.22e-11  -4.74e-8  -2.42e-8  -7.61e-9  -7.84e-9  -7.06e-9
    -7.22e-9  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 -1.21e-9  -2.87e-10 -1.58e-9
    -9.18e-10 -5.13e-8  2.08e-10  -5.05e-10 7.97e-8   -9.83e-10 2.11e-10
    -4.86e-10 -3.19e-10 -4.38e-9  -5.22e-8  -1.21e-9  -9.83e-10 6.69e-8
    4.46e-9   5.78e-8   4.67e-9   -2.97e-9  -4.73e-9  -2.87e-10 2.11e-10
    4.46e-9   2.12e-8   4.58e-9   1.01e-8   1.03e-9   -3.86e-10 -1.58e-9
    -4.86e-10 5.78e-8   4.58e-9   6.74e-8   4.38e-9   -2.53e-9  -4.32e-9
    -9.18e-10 -3.19e-10 4.67e-9   1.01e-8   4.38e-9   2.15e-8   5.67e-10
    4.68e-11  -5.13e-8  -4.38e-9  -2.97e-9  1.03e-9   -2.53e-9  5.67e-10
    8.06e-8   9.99e-10  2.08e-10  -5.22e-8  -4.73e-9  -3.86e-10 -4.32e-9
    4.68e-11  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  -9.51e-10 1.55e-9   -6.48e-10
    2.26e-9   -4.75e-8  2.02e-8   6.52e-10  1.45e-7   -1.75e-9  6.72e-10
    -2.51e-9  8.33e-10  -2.38e-8  -5.19e-8  -9.51e-10 -1.75e-9  4.72e-8
    -1.88e-8  3.74e-8   -1.98e-8  4.16e-9   -7.01e-9  1.55e-9   6.72e-10
    -1.88e-8  3.59e-8   -1.98e-8  2.55e-8   7.48e-10  6.13e-9   -6.48e-10
    -2.51e-9  3.74e-8   -1.98e-8  4.72e-8   -1.89e-8  3.44e-9   -6.72e-9
    2.26e-9   8.33e-10  -1.98e-8  2.55e-8   -1.89e-8  3.59e-8   3.21e-10
    5.44e-9   -4.75e-8  -2.38e-8  4.16e-9   7.48e-10  3.44e-9   3.21e-10
    1.51e-7   -7.87e-10 2.02e-8   -5.19e-8  -7.01e-9  6.13e-9   -6.72e-9
    5.44e-9   -7.87e-10 1.51e-7
```

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:

```
# Hz S RI R 50.0
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
```

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

```
# Hz S RI R 50.0
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
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
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  -3.90e-3   6.39e-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
  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
[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 and W2 is the switch term of port 2

Table 3: VNA Parameter

	S11	S21	S12	S22	W1	W2
Numerator Receiver	‘b’	‘b’	‘b’	‘b’	‘a’	‘a’
Numerator Port	1	2	1	2	1	2
Denominator Receiver	‘a’	‘a’	‘a’	‘a’	‘b’	‘b’
Denominator Port	1	1	2	2	1	2
Source Port	1	1	2	2	2	1

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: $2N_{Ports} + 2N_{Parameters}N_{Freq}$

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

For the definition of ‘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:

```
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}.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
        % 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);
        end
        % 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();
end

function p = ReadPort(f3, version)
% Read Port
ModeType = {'', 'd', 'c'};
IndexType = {'', ':I', ':II', ':III', ':IV', ':V', ':VI', ':VII', ':VIII',
    ':IX', ':X', ':XI', ':XII'};
if (version < 3)
    p = num2str(f3.ReadInt32());
else
```

```
    p = [num2str(f3.ReadInt32()) ModeType{f3.ReadInt16() + 1}
        IndexType{f3.ReadInt16() + 1}];
end
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.UncList();
list.BinarySetDataFrom(f3);
n = NET.createGeneric('Metas.UncLib.Core.Ndims.RealNArray',
    {'Metas.UncLib.LinProp.UncNumber'});
n.Init1dData(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':

```
<!-- 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':

```
<!-- definition of VnaParameterType -->
<xs:complexType name="VnaParameterType">
  <xs:sequence>
```

```
    <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>
```

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

```
<!-- definition of ReceiverType-->
<xs:simpleType name ="ReceiverType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="1"/>
    <xs:enumeration value="b"/><!-- Test Receiver -->
    <xs:enumeration value="a"/><!-- Reference Receiver -->
  </xs:restriction>
</xs:simpleType>
```

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/b1,2re  a1/b1,2im  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

Parameter	Description
'S[1,1]'	S-Parameter, reflection at port 1
'S[2,1]'	S-Parameter, transmission from port 1 to port 2
'b1,1'	Test receiver of port 1 when the source is switched to port 1
'b2,1'	Test receiver of port 2 when the source is switched to port 1
'a1,1'	Reference receiver of port 1 when the source is switched to port 1
'a1/b1,2'	Ratio of reference to test receivers of port 1 when the source is switched to port 2

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/b1,2re a1/b1,2im 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 $\frac{a_1}{b_1}$ when the source is switched to port 2.

CV[2,1] describe the covariance between the real and imaginary parts of the receiver ratio $\frac{a_1}{b_1}$ when the source is switched to port 2.

CV[2,2] is the variance of the imaginary part of the receiver ratio $\frac{a_1}{b_1}$ 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/pna/dbcal.html
- http://literature.cdn.keysight.com/litweb/pdf/ads2001/pdf/instrumentio.pdf
- https://na.support.keysight.com/vna/help/latest/S5_Output/SaveRecall.htm#cti
- http://hermes.eee.nott.ac.uk/teaching/ads/doc/cktsim/ck0419.html

10.1 Examples

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

```
CITIFILE A.01.01
NAME DATA
VAR FREQ MAG 3
DATA S[1,1] RI
DATA U[1,1] RI
VAR_LIST_BEGIN
1.0000000000e+009
2.0000000000e+009
3.0000000000e+009
VAR_LIST_END
BEGIN
-9.1600000000e-001,3.9100000000e-001
-6.9000000000e-001,7.1700000000e-001
-3.5500000000e-001,9.2900000000e-001
END
BEGIN
2.3579652245e-003,2.8635642127e-003
2.8142494559e-003,2.8000000000e-003
3.2124756808e-003,2.6381811917e-003
END
```

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

```
CITIFILE A.01.01
NAME DATA
VAR FREQ MAG 3
DATA S[1,1] RI
DATA U[1,1] RI
DATA S[2,1] RI
DATA U[2,1] RI
DATA S[1,2] RI
DATA U[1,2] RI
DATA S[2,2] RI
DATA U[2,2] RI
VAR_LIST_BEGIN
1.0000000000e+009
2.0000000000e+009
3.0000000000e+009
VAR_LIST_END
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
5.7061365532e-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
    2:S[1,1]im           CV[1,1]    CV[2,1]    CV[3,1]    CV[4,1]    CV[2,2]
    CV[3,2]    CV[4,2]    CV[3,3]    CV[4,3]    CV[4,4]
```

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:

```
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
    2:S[1,1]im          CV[1,1]   CV[2,1]   CV[3,1]   CV[4,1]   CV[2,2]
    CV[3,2]   CV[4,2]   CV[3,3]   CV[4,3]   CV[4,4]
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
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
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.

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:

```
VCOLCV
--------------------------------------------------------------------------------
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
    2:S[1,1]im          CV[1,1]   CV[2,1]   CV[3,1]   CV[4,1]   CV[2,2]
    CV[3,2]   CV[4,2]   CV[3,3]   CV[4,3]   CV[4,4]
```

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

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

14.2.1 Example

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

```
VCOLCV
--------------------------------------------------------------------------------
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
    2:S[1,1]im          CV[1,1]   CV[2,1]   CV[3,1]   CV[4,1]   CV[2,2]
    CV[3,2]   CV[4,2]   CV[3,3]   CV[4,3]   CV[4,4]
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
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
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 6: File types

Description	Extension
S-Parameter Data Binary	(.sdatb)
S-Parameter Data Xml	(.sdatx)
S-Parameter Data Covariance Text	(.sdatcv)
S-Parameter Data Touchstone	(.s*p;.ts)
VNA Data Binary	(.vdatb)
VNA Data Xml	(.vdatx)
VNA Data Covariance Text	(.vdatcv)
VNA Data CITI	(.cti;.citi)

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}
}
% missing /AF entry and /AFRelationship entry, see
% https://tex.stackexchange.com/q/426385/139832


\datafile{Collection/}{Standard_01.sdatb}
\datafile{Collection/}{Standard_02.sdatb}

\begin{document}
See the embedded data files ...
\end{document}
```

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:

```
/dcc:digitalCalibrationCertificate
  /dcc:measurementResults
    /dcc:measurementResult
      /dcc:results
        /dcc:result
          /dcc:data
            /dcc:byteData
              /dcc:filename
```

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

```
  /dcc:digitalCalibrationCertificate
    /dcc:measurementResults
      /dcc:measurementResult
        /dcc:results
          /dcc:result
            /dcc:data
              /dcc:byteData
                /dcc:dataBase64[../dcc:fileName='NAME']
```