CONVERSION.EXE format conversion rules

In the following the program “CONVERSION.EXE” is simply called “the Program”.

The program is able to manage all the possible conversions (except that writing into pl4 files is presently not implemented) between the following formats:
- Atp (.PL4)
- ASCII PlotXY (.ADF)
- Matlab format (.MAT)
- ASCII COMTRADE format (.CFG and .DAT)

The Program is able to store data in the Matlab 4.0 file format. This can be read seamlessly in any Matlab version.
It is also able to read Matlab 4.0 files. If files to be read by the program are to be written with Matlab versions rel. 5.0 and later, they are to be obtained using the Matlab Save command with the option “V4” included.

The Comtrade format considered by the Program is the IEEE 1999 standard (that should be identical to the IEC 60255-24).

Some of the characteristics of these formats are considered of lesser significance, and therefore they are either not-managed or conventionally managed.
To manage the different ways in which different formats deal with signals, some conventions for converting signal names of different variables are adopted (and described later on).

1. SIGNAL-TYPES

The user of any of the file formats dealt with in this document needs to know not only the signal names and the corresponding numerical data, but also the signal type, if possible.
Different file formats give different possibilities to store the signal type along with the signal names.
In all cases managed by the Program the types and the corresponding units are those indicated in the table I; the table also contains the small strings that the Program uses as a prefix when displaying variable names, to show the corresponding type.

Table I: File types, Units of Measure and prefixes.

Signal Type	*Unit of Measure*	*Prefix*
angle	Degrees (DEG)	"a"
current	Ampere (A)	"c"
energy	Joule (J)	"e"
power	Watt (W)	"p"
voltage	Volt (V)	"v"
TACS	*undefined* (?)	"t"
MODELS	*undefined* (?)	"m"
current (real part)	Ampere (A)	"cr"
current (imaginary part)	Ampere (A)	"ci"
voltage (real part)	Volt (V)	"vr"
voltage (imaginary part)	Volt (V)	"vi"
Universal Machine	*undefined* (?)	"u" or "u#" or "u##" (*)
Synchronous Machine	*undefined* (?)	"s" or "s#" (*)
Digital signal (**)	*pure number* ( ) (**)	"d" (**)
Unidentified	*undefined* (?)	"?"

(*) # indicates a digit (**) Only for Comtrade file formats

2. CONVERSION FROM PL4

2.1. Conversion to ADF and Matlab

PL4 names can be not acceptable as ADF names since:

a) PL4 names can contain blank spaces while ADF ones cannot

In addition they are not as clear as they could be, without some processing, because:

b) the type of PL4 variables are stored outside the corresponding names, while there is not such possibility in ADF files

PL4 names are not acceptable as Matlab names as well, for the same reasons a) and b), and, in addition,

c) several characters are unacceptable within Matlab names.

To solve these problems, some name conversion is needed. The Program , for simplicity, uses the same conversion algorithm for translating PL4 names into ADF or MAT names. These rules are the same as used in the widely diffused Pl42MAT from the same author, and described in detail in Appendix.

2.2. Conversion to Comtrade

Pl4 names are acceptable as Comtrade names.

To identify the type of the stored signals, the Program stores the Units of measure in the corresponding field.

In addition, for more clarity, the Program stores *Prefixes* (see sect. 1) information in the, otherwise unused, "ccbm" field.

3. CONVERSION FROM COMTRADE

While reading Comtrade files the Program tries to understand variable types from their unit of measure, according to the correspondence shown in Table I. In case the Unit of Measure is "?"and the first character of "ccbm" is alphabetical, this character becomes the *prefix* of the variable (cf. again Table I).

The program is able to understand standard modifiers for unit of measure "k" and "m". In the output files the corresponding variables are converted into SI Units. If, for instance, a variable "X" is read from the Comtrade file with unit "kV", in an ADF output file the variable name will be "vX" and will be expressed in V (Volts).

3.1. Conversion to ADF and Matlab

The following two conversion rules apply for conversion into both ADF and Matlab. The third one is performed only for conversion into Matlab format.

1. if the names do not contain embedded *separators* (=blanks, commas, tabs), they are left as they are and prefixed with the prefix shown in Tab I.
2. if the names contain embedded separators, they are considered was word separator. Then:
   - each character is converted into lowercase, except the first one that is converted into uppercase
   - all the separators are eliminated
   - prefixes got from the Comtrade file are added at the beginning of the names resulting from the other steps.
3. after the previous steps are completed non-alphabetical characters are converted into '_'.

4. CONVERSION FROM ADF

Naming conversions to Comtrade are not necessary, and not made.

Conversions to Matlab are only made to make the names compatible with Matlab standard:

- if the first character is not a letter, the character 'x' is prefixed
- if some inner character is not a letter, or a digit, or the character '_', it is converted into '_'.

5. CONVERSION FROM MATLAB

Naming conversion

Since Matlab is the standard that poses the more severe limitations to names, there is no problem in converting names into any other format. Therefore while converting from Matlab to ADF or Comtrade, the names remain unchanged.

Matlab files can be of type V5, created using the "save" command of Matlab rel. 5.x or 6.x, or type V4, created by saving with Matlab releases up to 4.0, or with the save command of Matlab 5.x or 6.x, and the option "V4" selected.

The Program is able to read only V4 version of Matlab.

Different kind of arrays can be stored in a Matlab file: matrices of integers, floats, text strings.

However the Program is intended for dealing with files containing output of either measures or simulations. Therefore the files managed by the Program must always contain a "time" (i.e., a quantity monotonically increasing) and some "signals" associated to that time (therefore all the signals should have a number of point equal to that of time).

As a consequence of these characteristics, to be understood by the Program, a mat file has to comply with some specific requirements:

- it must contain only matrices of floats sharing the same number of rows
- it has to contain a "time" variable (i.e., a quantity monotonically increasing). If there is in the file a single column variable having as name "t" it is assumed to be time. Otherwise, the first column of the first variable in the file is assumed to be time.

If some of the variables in the file has multiple columns, it stores multiple signals. In this case the Program adds a unique suffix to the different columns, constituted by a progressive number between parentheses.

APPENDIX: Naming conversion rules from PL4 to ADF and MAT

ADF and MATLAB names are generated by pl42mat according to the following rules:

0. Time is simply indicated as "**`t`**"
1. All the .PL4 node names and TACS, MODELS, Universal Machine and Synchronous Machine variable names are converted into lowercase, except the first characters that are converted into uppercase
2. If the names contain dashes or embedded blanks, they are converted into underscores ('_'); blanks at beginning or end of names are discarded
3.
   - The names for node voltages are obtained adding at the beginning of the node names as they are after step 2 the character '**`v`**'
   - The names for branch voltages and currents are obtained combining the two node names into a unique name and then adding at the beginning of the resulting string the character '**v**' or '**i**' respectively; if one of the nodes is "   ", it is changed into "**Terra** "
   - Names of TACS or MODELS variables are obtained adding at the beginning of the names as they are after step 1 the characters '**`t`**' or '**m**' respectively"
   - Names of Universal Machine and Synchronous Machine variables are obtained adding at the beginning of the names as they are after step 1 the characters '**u**' or '**s**' respectively, followed by one or two digit(s) indicating the corresponding machine number (but up to 9 SMs are supported).

Examples:

(from DC5)

Node of Voltage **`' GENT ': vGent`**

Voltage Difference between **`'TRANFF'`** and **`'OPEN  ': iTranffOpen`**

Current between **`'GEN   '`** and **`' GENT ': iGenGent`**

Current between **`'LOAD  '`** and **`'      ': iLoad`**

(from DCn12)

UM variable **`'UM-1  ' - 'TQGEN': u1Tqgen`**

(other)

Voltage Difference between **`'UMPOS '`** and **`'      ': vUmposTerra`**

Voltage Difference between **`'      '`** and **`'UMNEG ': vTerraUmneg`**

SM variable **`'MACH 1'`** - **`'ID    ': s1Id`**

SM variable **`'MACH 1'`** - **`'TQ GEN': s1Tq_gen`**

SM variable **`'MACH 1'`** - **`'ANG 1 ': s1Ang_1`**

SM variable **`'MACH 2'`** - **`'ANG 1 ': s2Ang_1`**

MODELS variable **`'MODELS'`** - **`'SOC   ': mSoc`**

MODELS variable **`'MODELS'`** - **`'Iw    ': mIw`**