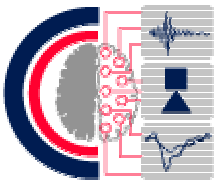
EEProbe™
Reference Manual

complete signal analysis for psychophysiological research
(ERP processing for EEG/MEG)



ANT Software BV, Enschede, The Netherlands
Max Planck Institute of Cognitive Neuroscience, Leipzig, Germany

A.N.T. Software BV
Colosseum 2
7521 PT Enschede
The Netherlands

e-mail eeprobe@ant-software.nl
phone +31 (0)53-4365175
fax +31 (0)53-4303795
internet www.ant-software.nl

Printed March 18, 2003

DISCLAIMER

We have attempted to write this manual as accurately as possible. However, mistakes are bound to occur, and we reserve the right to make changes to EEProbe, which may render parts of this manual invalid. We assume no liability as a result of the use or application of this product.
No part of this manual may be copied or reproduced without the express permission of the authors.

NOTICE

EEProbe, the incorporated EEP modules and any auxiliary scripts are research software. It is not to be used as the sole or partial basis for clinical diagnosis or treatment.

Table of Contents

1 EEPROBE

Contents and Overview

ANT Software b.v.
Max-Planck-Institute of Cognitive Neuroscience
ANT / MPI

1.1 Introduction

EEP 3.1 is a Unix software for Event Related Potential studies. The fileformats are partly compatible with the MS-DOS based Software EEP 2.0 by Erdmut Pfeifer.
The package consists of several programs (*modules*) to process digitized EEG/MEG records(cnt-files) and the ERP's(avr-files) derived from these records.
Each module is started by a shell-command and is controlled by command line arguments and, optionally, by a configuration file. Protocol output is directed into the standard output channel. Status and error messages are directed into the standard error channel. After successful operation a value of 0 (zero) is returned to the calling process, otherwise a different value.
EEP is not a general purpose signal processing toolkit. It offers only the most frequently needed facilities for ERP studies in its highly specialized modules.
New users should be able to work with the Unix operating system (file management, view/edit textfiles, shell) and should know the fundamentals of the ERP methodology. Then they should start by browsing this page as well as the "Usage" page to get a first overview.
Normally, EEP is installed ready-to-use. No per-user configuration is required. Some useful sample data files are available in the EEP directory:

```
cfg          - sample configurations
fir          - sample filter coeffs
app-defaults - resource templates for Motif apps
doc          - the HTML documentation
```

1.2 Modules

The following tables list the EEP modules and point to their individual reference pages. For some simple programs there is only the synopsis. Calling these modules with the `--help` argument and the configuration file comments should give you sufficient information.
The modules are classified according to its main purpose into Converter, Viewer, Data Management and Signal Processing. Of course, this classification is not perfect. For example, you can find signal processing options in the viewer programs or management functionality in the converters.

The tables also list the available computer architectures on which the programs can be run. However, this list was assembled at the Max-Planck Institute, Leipzig. Only the 'Linux i386' is fully supported, and 'IRIX mips' partly supported by ANT Software b.v. Please contact us on any of the other versions.

1.2.1 Converter

Name	Purpose	OSF1 alpha	Linux "i386"	IRIX mips	SunOS sparc	DOS "i386"
ns2riff	NeuroScan SynAmps EEG Import	x	x	x		x
bti2riff	BTi MEG Import				x(1)	
refa2cnt	TMS PORTI-32/MREFA EEG Import		x(1)			
pegasus2cnt	Pegasus 2 Import		x			
egiraw2cnt	Electrical Geodesics 'Simple Binary Format'		x			
dig2cnt	DIG EEG Import (UCSD)	x	x	x		
wg2cnt	Walter Graphtek Raw Data Import	x	x	x		
wasc2cnt	ASCII Raw Data ("Bonn-style") Import	x	x	x		
sig2cnt	BrainLab Raw Data Import	x	x	x		
bv2cnt	BrainProducts EEG Import	x	x	x		
eep2riff	EEP 2.0 Format Import	x	x	x		x
riff2eep	EEP 2.0 Format Export	x	x	x		x
edf2cnt	European Data Format Import		x			
cnt2edf	European Data Format Export		x			
cnt2asc	ASCII Raw Data Export	x	x	x		
avr2asc	ASCII ERP Data Export	x	x	x		
raw2cnt	Generic Raw Data Import Filter	x	x	x		
mex funcs	Data Exchange with Matlab/Octave	x	x	x		
Scripts	(require 'bash', 'awk', 'perl')					
cntextract	extract ascii trials based on trg file	x	x	x		
asa2avr	converts asa average file-format into EEP avr file-format	x	x	x		x
avr2asa	converts EEP avr file-format into asa average file-format	x	x	x		x
asc2avr	converts ASCII data (as produced by "avr2asc -t") to EEP avr file-format	x	x	x		x

avg2avr	converts NS avg to EEP avr file-format	x	x	x		x
avr2avg	converts EEP avr file-format to NS avg	x	x	x		x
ev2trg	merge events from NS ev2 file to EEP trg file	x	x	x		x

(1) only at the corresponding acquisition machines

1.2.2 Viewer

Name	Purpose	OSF1 alpha	Linux "i386"	IRIX mips	DOS "i386"
cntview	print cnt header informations	x	x	x	
avrview	print avr header informations	x	x	x	
xcnt	EEG/MEG Raw Data Viewer	x	x	x	
xeog	Electrooculogram Epoch Classification	x	x	x	
xavr	ERP Data Viewer	x	x	x	

1.2.3 Data Management

Name	Purpose	OSF1 alpha	Linux "i386"	IRIX mips	DOS "i386"
cntsetup	patch labels, scalings, compute bipolar channels etc.	x	x	x	
cntepoch	cut raw data records	x	x	x	x
cntcat	Raw Data Concatenation	x	x	x	x
cntdown	Raw Data Downsampling	x	x	x	
cntmono	obsolete, use cntsetup	x	x	x	
cntevents	extract control data from signal archives	x	x	x	
cnttrials	classify trials rejected or not(as cntaverage would do)	x	x	x	
trgcount	list available codes in trigger file (requires 'bash', 'awk')	x	x	x	x
trgisi	extract ISI values from trigger file (requires 'bash', 'awk')	x	x	x	x
trged	Conditional averaging 1 - trigger recoding	x	x	x	x
trgvalid	Conditional averaging 2 - trigger/response time validation (requires 'awk')	x	x	x	x
trgselect	Conditional averaging 3 - randomly select a specified #trials	x	x	x	x
avrdown	ERP Data Downsampling	x	x	x	
avrmerge	merge avr's with different channel sets to one file	x	x	x	
avrchannels	remove or rename channels in avr files	x	x	x	
xtrctavr	extract channels from avr file (requires 'perl')	x	x	x	
pasteavr	merges avr files (requires 'perl')	x	x	x	

1.2.4 Signal Processing

Name	Purpose	OSF1 alpha	Linux "i386"	IRIX mips	DOS "i386"
cntfilter	Finite Impulse Response Filter	x	x	x	

xfir	Finite Impulse Response Filter Design	x	x	x	
cntreref	Raw Data Rereferencing	x	x	x	
cntdetrend	remove offsets and linear trends	x	x	x	
cntreject	mark disturbed signal epochs	x	x	x	
cntreject_t	mark disturbed trials	x	x	x	
cntaverage	Single Subject ERP Calculation	x	x	x	
avrfilter	ERP Finite Impulse Response filter	x	x	x	
avrdiff	calculate ERP differences	x	x	x	
avrstats	calculate ERP significance	x	x	x	
avrreref	ERP Rereferencing	x	x	x	
avraverage	ERP Grand Averaging	x	x	x	
avrretrieve	ASCII Export of ERP Dataset Information	x	x	x	
avrprocess	Basic ERP Operations	x	x	x	
avrinterpol	Spatial ERP Interpolation	x	x	x	
avrdetrend	remove linear trends from ERP's	x	x	x	

2 USAGE

General usage description

- Calling Conventions
- Evaluation Steps
- General Hints

2.1 Calling Conventions

The EEP modules are started with shell commands and are controlled via command arguments. A call without arguments or with one of --help, -h or -? will produce a help screen which shows the allowed/required arguments for this module. This information is also included in the HTML documentation pages for the modules.
The arguments can be classified as:

short options: -a -b or -ab
: a '-' followed by a letter, multiple short options can be merged to one argument

long options: --help
: starting with '--', followed by a word

value options: -s 4 or -w liste1.txt
: a special short option, followed by an optional space character and a value (the modules know after which short option they have to look for a value argument)

filenames
: All other arguments; in most cases a data input file and/or a configuration file is required. Other filenames are often optional (their names can be derived automatically).

There are some global switches which are valid for each module. They are passed as long-option arguments in the command line (the first variant in the list is the default):

--bar/--nobar
: enable/disable the progress bar

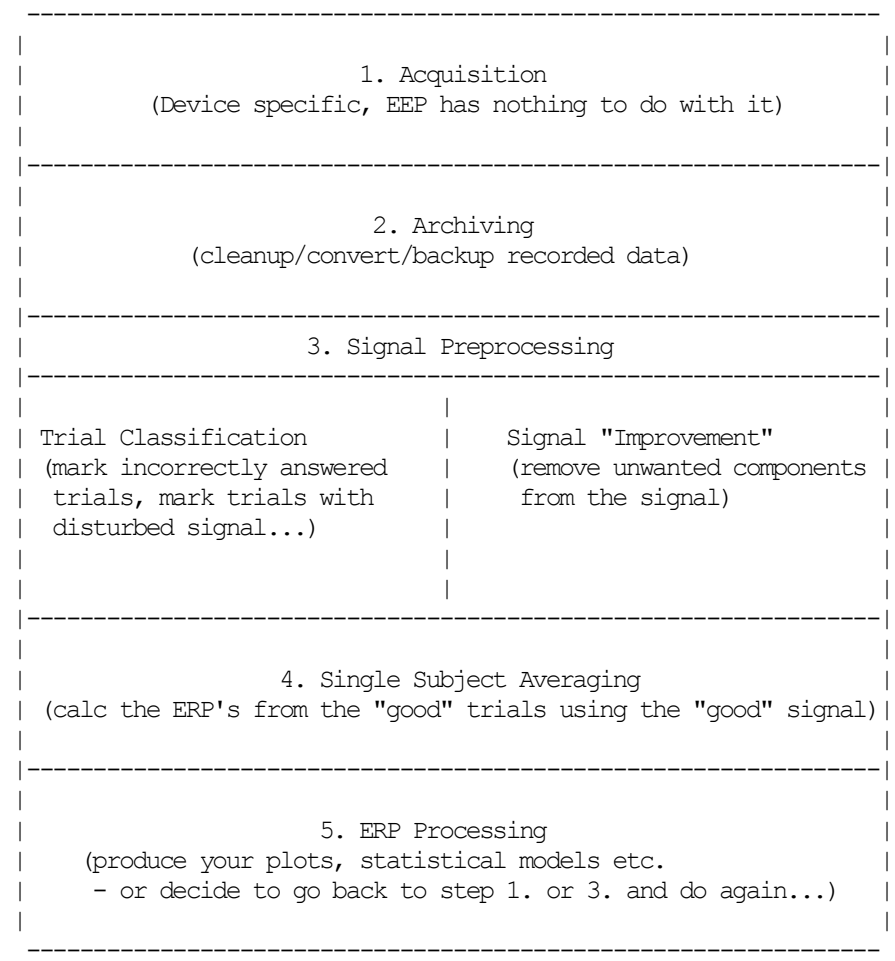
--nodebug/--debug
: additional output of internal tables, states ... (useful only in development)

--noquiet/--quiet
: suppress all terminal output (including error messages !!!)

--nolog/--log
: print all protocol output to stdout AND stderr if stdout is not a terminal

2.2 Evaluation Steps

The evaluation sequence for an ERP study could be outlined as follows:



Performing these steps leaves you with a couple of files in the so-called "EEP project tree". The user is responsible for keeping his project tree in a consistent state. EEP only supports this with its naming and directory conventions. The general hints below might also be helpful.

A typical EEP project tree could look like this:

```
think/cfg/average.cfg        - EEP configuration files
think/cfg/detrend.cfg
...
think/sh/eval.sh             - scripts, tools ...
think/sh/do_what_i_want.sh
think/sh/trgpatch.awk
...
think/vp01/vp01.cnt          - data record of one subject
think/vp01/vp01.trg            (signal data, classification data...)
think/vp01/vp01.rej
think/vp01/vp01.res
...
think/vp01/avr1/vp01cr.avr   - several ERP-datasets
think/vp01/avr1/vp01cf.avr     (each avr-file contains the ERP of
 ...                            one subject in one condition)
think/vp01/avr2/vp01cr.avr
think/vp01/avr2/vp01cf.avr
----- ----          --
  |    |            |
  |    |             ------- condition shortcut (2 characters recommended)
  |    |
  |     -------------------- subject code (4 characters recommended)
  |
   ------------------------ project codename
```

2.2.1 Acquisition

Data acquisition is done with device specific software. EEP supplies a couple of converters to transform the records in a supported format.

2.2.2 Archiving

After acquisition you have to create "clean signal archives" from recorded data. Typically, you will use the conversion and management modules here. The goal is to produce well-aligned datasets for all subjects in the study (equal channel number/labels, equal reference etc.).

2.2.3 Signal Preprocessing

Trial Classification

Before calculating an ERP you have to decide which set of trials should be averaged to form this ERP.
This requires a response dependent trial selection (**trged** or your own trigger file manipulation script) to mark incorrectly answered trials and a signal dependent trial selection to mark the disturbed trials(automatically via **cntreject**, **cntreject_t** or manually via **xcnt**, **xeog**).
None of the modules mentioned above changes signal data. All what they do is to produce one or more classification lists (.trg, .rej, .cls) which are used in the averaging step to select the good trials.
Note that the trial classification interacts with the signal processing steps. A good filter can repair disturbed trials or the classification results show that the signal must somehow be filtered...

Signal "Improvement"

The original signal record is often too noisy or otherwise disturbed. EEP 3.0 offers filter programs which allows to compensate/remove such artifacts (**cntfilter**, **cntdetrend**).

2.2.4 Single Subject Averaging

This step (**cntaverage**) uses the signal and all the classification lists to calculate the ERP's by averaging the signal of all good trials.
You can treat this step as an important fixpoint in each evaluation. All preceding steps can be seen as preparations to provide **cntaverage** with the information it needs to perform its job successfully.

2.2.5 ERP processing

Each ERP (stored as .avr file) is basically a simple channel * time data matrix for one subject in one condition. EEP offers several modules to combine/process/view/plot the individual ERP matrices and the study which they form together, but it doesn't claim to cover all possible evaluations one can think of. There is always the option and often the need to convert the .avr files and to proceed with your preferred computation tools.

2.3 General Hints

Time axis manipulations in the cnt-files (**cntcat**, **cntepoch**, **cntdown**), should be performed at first. The extern control data files(.trg, .rej, .cls) become invalid during such processing steps and you would have to recreate all this at great expense.
Each EEG/MEG record consists normally of at least 3 files: signal data (.cnt), triggers (.trg) and rejections (.rej). These three files should have the same basename (the part before the dot) after the first "standard preprocessing". The EEP modules can automatically find the correct triggers and rejections from the cnt filename this way and interactive work at command line level becomes much easier.
In shellscripts for automated evaluation, supply all filenames in the command line. Do not rely on the automatic filename rules in this case.
Consider also such tools as "make". It allows you to store and access frequently used shell commands in a convenient way. See the example Makefile for more details.
I suggest to remove all write permissions from data files which cannot be reproduced by automatic scripts. This prevents you from accidental overwriting valuable files. Note that most Unix programs, including the EEP modules, do not ask before overwriting existing files.
Although many of the EEP 3.x files are compatible with EEP 2.0, you should be careful when mixing the DOS and Unix modules. Make sure that each textfile has a CR-LF newline sequence before using it with DOS, make sure that files created in Unix do not violate the DOS filename limitations and make sure that the file permissions/ownership allow to access files created in Unix from DOS and vice versa.
EEP modules write a protocol of the current run to the standard output channel. You should save this output in logfiles.
At any time, the central project directory (normally at a server) should contain a clean, current project state. You should NOT store any intermediate result but you should bring all manual work (configuration files, scripts, manual trial classifications) from local drives back to the server.
Before you start evaluation processes - check whether you can save network load (reach a faster execution for yourself and for others) with a working copy of the input data in a local filesystem of the evaluation machine. This can speed up your work dramatically, especially with the interactive viewers and when you are performing many operations on the same dataset.

3 CONVERTERS

File conversion utilities

3.1 avr2asc - ASCII ERP Data Export

3.1.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avr2asc 3.6   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:18 2003
------------------------------------------------------------------------------
  avr2asc <avr in> [-c <label>] [-t] [<txt out>]

options:
  -c <label>     convert this channel only
  -t             write table header
  -v             write variances
  <txt out>      output file, if omitted, stdout
  --msec         write first column as time-axis (msec)

files:
  foo.avr  ->  stdout
```

3.2 bti2riff - BTi MEG Import

3.2.1 Synopsis

```
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-99
bti2riff 3.11   (SunOS 5.6 sun4m)            Thu Apr 22 16:12:37 1999
------------------------------------------------------------------------------
  bti2riff <dest cnt> [options]

options:
  -f               convert REFERENCE channels
  -e               convert EXTERNAL channels
                   (MEG, EEG and TRIGGER channels are handled by default)
  -t <x0,x1...>    the "no trigger" value for each trigger channel
                   (default is 0)
  -P <pat>
  -S <scan>
  -s <sess>
  -r <run>
  -p <pdf>         BTi database selection
                   (default is to convert the first posted pdf)
```

3.2.2 Description

bti2riff converts a SHORT or FLOAT type processed data file from the BTi database to the compressed EEP format. The selection can be made via the BTi psel utility or via the bti2riff arguments.
For FLOAT data, the overall offset is removed channelwise and the resulting values are scaled back to SHORT using the original scaling factors. An error occurs if a value will not fit into a 16 bit signed after this.
The unit "T" is converted to "fT", "V" to "uV" and the scalings are changed accordingly to achieve more "handy" numbers.

3.3 bv2cnt - BrainProducts EEG data Import

3.3.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
bv2cnt 1.16   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:25 2003
------------------------------------------------------------------------------
  bv2cnt <bv_file1> [<cnt out>]

  where:
    <bv_file#>      BrainVision header file
```

3.4 cnt2asc - ASCII Raw Data Export

3.4.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cnt2asc 3.6   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:21 2003
------------------------------------------------------------------------------
  cnt2asc <source cnt> [-t] [-r] [-s <start>] [-l <length>] [-c <channel
file>] [<txt out>]

options:
  -t                 enable table header output
  -r                 raw mode, no real-world scaling, -s, -l in samples
                     instead of seconds
  -s <start>         start point of output (seconds or samples)
  -l <length>        length of output (seconds or samples)
  -c <channel file>  select channels specified in channel file
  <txt out>          output file, if omitted, output written to stdout

files:
  foo.cnt --> stdout
```

3.5 cnt2edf - European Data Format Export

3.5.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cnt2edf 1.10   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:25 2003
-------------------------------------------------------------------------------
  cnt2edf [-e] <source cnt> [<dest edf>]

options:
  -e      use EVENT EDF+ main code (triggers encoded as 'stimulus onset')

files:
  foo.cnt --> foo.edf
  foo.trg
```

3.5.2 Description

`cnt2edf` converts a cnt file to European Data Format file using the EVENT EDF+ extensions (see the ANT web-site http://www.ant-software.nl/ for more information).

3.6 dig2cnt - Digital " DIG " format (UCSD)

3.6.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
dig2cnt 1.2   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:24 2003
------------------------------------------------------------------------------
  dig2cnt <cfg> <source raw> [<dest cnt>]
```

3.6.2 Description

`dig2cnt` converts a " DIG " file to the cnt format using the riff compression extensions. The DIG file format was designed at the University of California San Diego, in the group of Steve Hillyard and Martha Kutas. No trigger/event information is available; the .trg file contains only a discontinuity marker at the start of the data.

3.6.3 Configuration file

The `dig2cnt` converter needs an external configuration file that defines the number of channels, sampling rate, and other parameters.
An example configuration is given below for a 64 channel configuration, where channel 5 and 6 have been recorded using slightly different gain settings:

```
[default settings]
samplerate:500   ; (in Hz)
scalemin:-5460   ; (in uV) value corresponding to digital minimum -32768
scalemax:5460    ; (in uV) value corresponding to digital maximum  32767

[channels]
1:LOPf
2:ROPf
3:LMPf
4:RMPf
5:LEL scalemin:-5000 scalemax:5000
6:REL scalemin:-5000 scalemax:5000
7:LLPf
8:RLPf
9:LPrA
10:RPrA
11:LTFr
12:RTFr
13:LLFr
14:RLFr
15:LDPf
16:RDPf
17:LTOc
18:RTOc
19:LTCe
20:RTCe
21:LLCe
```

```
22:RLCe
23:CHT
24:BCK
25:LMFr
26:RMFr
27:MiFo
28:MiPf
29:MiFr
30:A2
31:LHEy
32:HEOG
33:LIOc
34:RIOc
35:LLOc
36:RLOc
37:LLPP
38:RLPP
39:LLPa
40:RLPa
41:LDCe
42:RDCe
43:LMCe
44:RMCe
45:LDOc
46:RDOc
47:LDPP
48:RDPP
49:LDPa
50:RDPa
51:LCer
52:RCer
53:LMOc
54:RMOc
55:NOSE
56:ANK
57:LMPa
58:RMPa
59:MiCe
60:MiPa
61:MiPP
62:MiOc
63:LLEy
64:RLEy
```

3.7 edf2cnt - European Data Format Import

3.7.1 Synopsis

```
--------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
--------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
edf2cnt 1.9   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 15:31:54 2003
--------------------------------------------------------------------------------
  edf2cnt [-c] [-e] <source edf> [<dest cnt>]


options:
  -c      scan EVENT channel (if present) for 'change' events (non-standard EDF)
  -e      remove EVENT EDF+ main codes

files:
  foo.edf  ->  foo.cnt
               foo.trg

notes:
  (edf2cnt decodes EVENT channel when present, but does not decode
   simultaneous 'multiple events')
```

3.7.2 Description

`edf2cnt` converts a European Data Format file to the cnt format using the riff compression extensions.
EDF can contain multiple channels at different sampling frequencies: the `edf2cnt` converter uses only channels of the same sampling frequency: channel selection is determined by the conditions

- units Volts (e.g. uV)
- the most frequently occurring sampling frequency:
    - in case of identical number of channels at different sampling frequencies, the 'first' sampling frequency is used

3.8 eep2riff - EEP 2.0 Format Import

3.8.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
eep2riff 3.11   (Linux 2.2.14-5.0 i686)      mvelde, Tue Feb 25 14:54:17 2003
------------------------------------------------------------------------------
  eep2riff <source cnt> [<dest cnt>]

files:
  foo.cnt  ->  fooc.cnt
               fooc.rej
```

3.8.2 Description

eep2riff converts (uncompressed) EEP 2.0 cnt-files (generated by "ns2eep") in the compressed EEP 3.x format. Altough EEP 3.x can deal with EEP 2.0 files, it is highly recommended to have the data in their compressed form.
The parts of the original file are transformed as follows:

EEP 2.0 binary header
: is copied from the source file to the destination file; this block is unused in EEP 3.x, it's archived only

signal data
: are compressed; control informations are extracted from the signal block (see below)

trigger table
: **ns2eep** converts all trigger bytes which are different from 0 (zero) and the DISCONTINUITY and DC_RESET flags.
Not converted are the EEP 2.0 FILTER-ARTIFACT flags.

rejection flags
: are concatenated to epochs and written as ASCII-files (*.rej).

3.9 egiraw2cnt - EGI Simple Binary Format

3.9.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
egiraw2cnt 1.3   (Linux 2.2.14-5.0 i686)     mvelde, Tue Feb 25 14:54:22 2003
-------------------------------------------------------------------------------
  egiraw2cnt <source rec> [<dest cnt>]
```

3.9.2 Description

`egiraw2cnt` converts a Net Station (Electrical Geodesics) 'Simple Binary Format' to the cnt format using the riff compression extensions.
Only supported is the 'Simple Binary Format' (Raw EEG File - With/Without Events) at Integer output precision[1], in A/D units (not converted to microvolts).
All trigger/event information is written to a .trg file.

[1] The floating point, double precision is not well tested in the current implementation.

3.10 ns2riff - NeuroScan SynAmps EEG Import

3.10.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
ns2riff 4.5   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:17 2003
------------------------------------------------------------------------------
  ns2riff [-t] <source cnt> <source cnt> ...  <dest cnt>

options:
  -t   write header information table(s) only, no conversion
```

3.10.2 Description

ns2riff converts NeuroScan SynAmps EEG records in the compressed EEP 3.x data format (loss-free compression to approx. 1/3 of the original size). Multiple files can be concatenated if their channel setup is the same.
The parts of the original file are transformed as follows:

NeuroScan binary header
: is copied from the first source file to the destination file; this block is unused in EEP 3.x, it's archived only

signal data
: are compressed; all sources are concatenated

trigger table
: **ns2riff** converts the triggers received via the 8-bit parallel SynAmps trigger port and the special marks for record start/stop and DC-reset which are generated by the Acquire program.
Not converted are the "Keyboard" and "Keypad" events (never used here).

Notes

(MG, 03/2000)

ns2riff 3.1 modified to convert SCAN 4.1 NS SynAmps-Files to RIFF-raw3 format
DC-reset trigger shifted by 70ms with SCAN 3.x
by 80ms with SCAN 4.1

(MvdV, 02/2001)

a special executable is available to solve some problems with some older neuroscan files (3.x)

(MG 2001/08/10)

modified to convert SCAN 4.0 NS SynAmps cnts

3.11 pegasus2cnt - Pegasus 2 Import

3.11.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
pegasus2cnt 1.4   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:23 2003
------------------------------------------------------------------------------
  pegasus2cnt <source sig> [<dest cnt>]
```

3.11.2 Description

`pegasus2cnt` converts a Pegasus 2 file to the cnt format using the riff compression extensions.
All trigger/event information is written to a .trg file.
Patient information is written to a .info.txt file.

3.12 raw2cnt - Generic Raw Data Import Filter

3.12.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
raw2cnt 3.2   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:18 2003
------------------------------------------------------------------------------
  raw2cnt [options] <src file> <cfg> <dest cnt>

options:
  -t <type>    data type; one of ascii, s8, s16, s32, f32
               default: ascii
  -o <ofs>     start of data in file (in lines (-t ascii) or bytes)
               default: first line (1) or first byte (0)
  -b <block>   number of subsequent data points for one channel
               default: 1
  -l <length>  write max. <length> samples per channel
               default: 0 (convert until end of file)
  -f <freq>    sampling frequency (in Hz)
               default: 250
  -s           swap byteorder when 16/32 bit binary types are read
```

3.12.2 Description

raw2cnt is intended to ease the signal data exchange with other software. It can convert several flavours of simple data files to EEP's cnt format.
Since most software can export some kind of "plain", "ascii", "raw" or "unformatted" matrices, there is a good chance that you can import whatever data you have via **raw2cnt**.
Additional information (channel labels, scalings etc.) and the information how to decode the actual data is expected from a configuration file and from command line arguments.

3.12.3 Data File

The data file can be nearly any "plain" list of numbers. The - not so hard - requirements are:

- The data values must be stored as ASCII text, 8/16/32 bit signed integer or 32 bit floating point numbers.
- The data values must be arranged in a systematic way:
  n values for first channel, n values for second channel, ..., n values for last channel; next n values for first channel ...
- In ASCII data files, the data values must be separated using whitespaces and the numbers must be readable with scanf()'s "%f" conversion.

If the input filename is given as "-", the standard input channel is read. Hence, it is possible to have **raw2cnt** at the end of a conversion pipe:

```
cat something | something2raw.pl | raw2cnt - some.cfg some.cnt
```

3.12.4 Configuration File

Together with some command-line options, the configutration file describes how the data values are to be interpreted. It has one section "[channels]" with one line for each channel in the input data file:
label unit raw_scale cnt_scale
label
unique, max. 10 character channel label or keyword "SKIP" (for ignoring this channel)
unit
max. 10 character physical unit string
raw_scale
a scaling factor which is multiplied with all input data values to scale these values in the allowed cnt file data range. (max. 32 bit signed integer but 16 bit signed integer (-32768..32767) is usually enough - and consumes less diskspace).
cnt_scale
the scaling factor (in *unit* per bit) which is stored in the cnt file.
Some remarks to the scaling: the cnt file stores

value * *raw_scale*

which is required to be an integral value. But when acessing data, the EEP programs scale the data with *cnt_scale*. Thus you will get

value * *raw_scale* * *cnt_scale* (in *unit*s).

Typically, if the input values are in real-world units, *cnt_scale* equals 1.0 / *raw_scale* and, if the input values are "raw digitized values", *raw_scale* is set to 1.0 and *cnt_scale* is set to an arbitrary value which shoud be known by the producer of such data.

3.12.5 Examples

ASCII Files

Below is a hypothetical example for an ASCII EEG data file, exported by whatever program.

```
patient    : Max
date       : 3.3.1905
channels   : 6
rate       : 250
A1     A2     A3     A4    DIGITAL  TEST
23.0   21.2   44.5   57.2  0        42
24.0   22.2   45.5   59.2  0        42
25.0   23.2   46.5   57.2  2        42
23.0   21.2   44.5   57.2  0        42
24.0   22.2   45.5   59.2  4        42
25.0   23.2   46.5   57.2  0        42
```

The config file to interpret the data channels could look like this:

```
[channels]
A1      uV   10  0.1
A2      uV   10  0.1
A3      uV   10  0.1
A4      uV   10  0.1
DIGITAL bit   1  1
SKIP
```

The input file stores microvolts with one dezimal digit - multiplication with 10 makes it integral. The channel "DIGITAL" is used "as is" and "TEST" is ignored.

The command line arguments must be:
`-t ascii` - it's an ASCII file,
`-o 6` - data start in 6th line,
`-b 1` - one value per channel,
`-f 250` - the sampling rate.
Hence, the conversion command could be:

```
raw2cnt -t ascii -o 6 -b 1 -f 250 foo.asc foo.cfg foo.cnt
```

Binary Files

Just to demonstrate the flexibility: convert an EEP .avr file - which is a simple binary data file...
Assume the **`avrview`** program gives us the following information:

```
average file data:
  condition:    singleWo  (OCHRE)
  trials:       299/61/360 (averaged/rejected/total)
  channels:     4
  rate:         250.000 Hz
  samples:      351 (-200 .. +1200 ms)
channels:
    no  label
     1  FP1
     2  FPZ
     3  FP2
     4  AF7
```

From the .avr fileformat specification, we know that there is an 38 byte header, followed by an 16 byte per-channel header, followed by the ERP vector of the first channel, followed by a variance vector etc. The .cfg file could be

```
[channels]
FP1         uV  100 0.01
SKIP
FPZ         uV  100 0.01
SKIP
FP2         uV  100 0.01
SKIP
AF7         uV  100 0.01
SKIP
```

and the conversion command is:

```
raw2cnt -t f32 -o $[38+4*16] -b 351 -f 250 foo.avr foo.cfg foo.cnt
```

3.13 refa2cnt - TMS PORTI-32/MREFA EEG Import

3.13.1 Synopsis

```
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-99
refa2cnt ?.?   (Linux 2.2.14-5.0 i686)      Thu Oct 26 14:35:31 2000
------------------------------------------------------------------------------
  refa2cnt [-i] [-p] <cfg> <dest cnt>

options:
  -i   incoming data are impedances, not voltages
  -p   incoming data are postprocessed, not raw device data
```

3.14 riff2eep - EEP 2.0 Format Export

3.14.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
riff2eep 3.9   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:17 2003
------------------------------------------------------------------------------
  riff2eep <source cnt> [<dest cnt>]

files:
  foo.cnt  ->  fooc.cnt
  foo.rej
```

3.14.2 Description

riff2eep converts compressed EEP data to the uncompressed EEP 2.0 format. The separate rejection data of EEP 3.x are mangled into the destination file.
The compressed file data blocks are transformed as follows:

archived NeuroScan/EEP 2.0 header
: if present, copied from source to destination; the current EEP 3.0 header informations are written over this (maybe invalid) header

signal
: is decompressed; the control informations (trigger, rejections) are mangled into the signal data

trigger
: The trigger codes from the <evt> chunk of the source file which are coded as integer numbers from 1..255, "Rs" (DC-reset) or "__" (discontinuity) are converted. All other triggers cannot be stored in the EEP 2.0 files.

Some informations cannot be stored in EEP 2.0 files:

signal data bits 16..31
: are completely lost

non-integer sampling rates
: are rounded to the next integer, all external control files become invalid

channel units
: are completely lost, everything in a EEP 2.0 file is assumed to be a voltage

3.15 sig2cnt - BrainLab Raw Data Import

3.15.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
sig2cnt 1.1.1.1   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:24 2003
------------------------------------------------------------------------------
  sig2cnt <source sig> [<dest cnt>]
```

3.15.2 Description

`sig2cnt` converts a OSG Brainlab file to the cnt format using the riff compression extensions.

3.16 wasc2cnt - ASCII Raw Data ("Bonn-style") Import

3.16.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
wasc2cnt 3.4   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:23 2003
------------------------------------------------------------------------------
  wasc2cnt <source asc> [<source inf> [<dest cnt>]]
```

3.17 wg2cnt - Walter Graphtek Raw Data Import

3.17.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
wg2cnt 3.6   (Linux 2.2.14-5.0 i686)         mvelde, Tue Feb 25 14:54:23 2003
------------------------------------------------------------------------------
  wg2cnt <source wg1> [<dest cnt>]
```

3.18 Data Exchange with Matlab/Octave

3.18.1 Introduction

It is possible to access the EEP binary data files (*.cnt, *.avr) within Matlab and Octave. For Matlab, it is done via "Matlab Extension" (mex) functions. They are installed in the **/usr/local/share/mex** subdirectory, which has to be included in the **MATLABPATH** environment variable.
For Octave, the functions are installed in the **/usr/local/share/eep/octave** subdirectory which can be activated by placing the line
LOADPATH=strcat(LOADPATH, ":/usr/local/share/eep/octave//");
in your **$HOME/.octaverc**.
The functions are documented in Matlab style - type 'help <function>' in the Matlab prompt or have a look at the associated .m files.

3.18.2 Functions

Please contact ANT Software for further information on the Matlab/Octave data exchange.
A (*) indicates functions which are available for Matlab only, not for Octave.
The following functions provide basic cnt file read/write access.

CNT_OPEN
CNT_LOAD
CNT_DUP(*)
CNT_APPEND(*)
CNT_CLOSE

The cnt write access is somewhat restricted: There is no function to create a cnt file from scratch; you will have to duplicate(**CNT_DUP**) an input cnt. This safes you from supplying all the data which are needed to create a valid cnt, they are just copied. At the other hand, this means that you cannot change such things as the number of channels or the sampling rate. Furthermore, because a cnt file is a "compressed signal archive", there is no way to write data at random positions. You can only write sequentially (**CNT_APPEND**).
The following functions provide trial-based cnt file read access, employing the same lists and configuration as **cntaverage**:

TRIAL_OPEN(*)
TRIAL_LOAD(*)
TRIAL_CLOSE(*)

The following functions allow to load the contents of an .avr file into a Matlab structure or to store such a structure as .avr file to disk. These are completely written as m-files - feel free to improve/adapt if needed.

AVR_LOAD
AVR_STORE

Note that the EEP modules store the variance in .avr files as

$s^2 = 1 / n * (sum(x^2) - n * (mean(x))^2)$

You eventually need to scale it by n / (n - 1).

3.18.3 Example

The following Matlab code fragments illustrate how to use the functions. Note that you must write nested open/close pairs for your handles (each open handle costs one file descriptor and dozens KB of memory).

```
% trial-based read access ---------------------------------------

% open the EEP files for a subject "pa53" in the cwd

[src, chan, length, rate] = cnt_open('pa53.cnt');
[th,co,st,cl,le] = trial_open(src, '../cfg/average.cfg', 'pa53.trg',
'pa53.rej');

% add some code to find interesting trial indices using
% the condition vector co and the classification vector cl
% note that the first trial has the index 0 in trial_load !!!

need_this_one = ...

% load the trial number need_this_one
% (m has 2*le elements in this case)

m = trial_load(th, need_this_one, ['EOGV';'EOGH']);

% do not forget to close the handles

trial_close(th);
cnt_close(src);

% plain read/write access ----------------------------------------

% create input/output handles
[src, chan, length, rate] = cnt_open('pa53.cnt');
dst = cnt_dup('pa53_new.cnt', src, 0);

% now you can load data from src, do something intelligent
% with it and write it to dst

M = cnt_load(src, 0, 1000, chan);
do_what_I_want(M);
cnt_append(M);

cnt_close(dst);
cnt_close(src);
```

3.19 cntextract - Extraction of ascii epochs based on trigger definition

3.19.1 Synopsis

```
------------------------------------------------------------------------------
ANT Software b.v. (c) 2002 - EEProbe script library
cntextract 1.9 - mvelde - Fri Feb 14 13:20:58 CET 2003
------------------------------------------------------------------------------
cntextract 1.9 - program to extract ascii trials based on trg file

usage:
  cntextract <cnt file> <trg file> <trigger code> <trial start> <trial end> [<rej file>]

  <trigger code>    :   as found in the trg file
  <trial start>     :   start time (msec) of trial, e.g. -200
  <trial end>       :   end time (msec) of trial, e.g. 800
  [<rej file>]      :   optional file containing rejection intervals
  (output files use the same basename of the <cnt file>,  plus the trigger code,
   and numbered as a sequence of trials; first line contains the channel labels)
```

3.20 asa2avr - converts asa average file-format into EEP avr file-format

3.20.1 Synopsis

```
------------------------------------------------------------------------------
                        Max-Planck-Institute of Cognitive Neuroscience 1997

asa2avr.pl: Version 1.15 released by mvelde, 19.09.2002
------------------------------------------------------------------------------



   *** Syntax-info ******************************************************
   *                                                                      *
   *  converts asa average file-format into EEP avr file-format       *
   *  Usage: asa2avr asafile [<avrfile>] [--trials avr,rej] --conds
label,color*
   *  Options:                                                            *
   *  asafile   asa average file name                                     *
   *  avrfile    EEP-.avr file name                                       *
   *    options needed to define values unknown to standard asa files.*
   *                                                                      *
   ************************************************************************
```

3.21 avr2asa - converts EEP avr file-format into asa average file-format

3.21.1 Synopsis

```
------------------------------------------------------------------------------
                        Max-Planck-Institute of Cognitive Neuroscience 1997

avr2asa.pl: Version 3.5 released by mvelde, 19.09.2002
------------------------------------------------------------------------------



   *** Syntax-info ******************************************************
   *                                                                      *
   *  converts EEP avr in asa file-format                             *
   *  Usage: avr2asa avrfile [<msrfile>]  [--MEG|--ignore "reg_expr"] [--
ASCII] [--COV]*
   *  Options:                                                            *
   *  avrfile    EEP-.avr file name                                       *
   *  msrfile    ASA-filename (without extension,                         *
   *             output .msr and .msm files created)                      *
   *                                                                      *
   ************************************************************************
```

3.22 asc2avr - converts ASCII data (as produced by "avr2asc -t") to EEP avr file-format

3.22.1 Synopsis

```
------------------------------------------------------------------------------
                        Max-Planck-Institute of Cognitive Neuroscience 1997

asc2avr.pl: Version 1.8 released by rsmies, 24.01.2003
------------------------------------------------------------------------------


   *** Syntax-info ***************************************************
   *                                                                  *
   *  converts ASCII data (as produced by "avr2asc -t") to EEP avr file-
format*
   *  Usage: asc2avr <freq> [-b <offset>] <asc> <avr>                 *
   *  Options:                                                        *
   *    <asc>      ASCII input file                                   *
   *    <freq>     sampling frequency (in Hz)                         *
   *    <avr>      avr output file                                    *
   *    <offset>   baseline offset (trigger offset time in msec)      *
   *                                                                  *
   ********************************************************************
```

3.22.2 Notes:

This avr importer is not compatible with `avr2asc` option `--msec`

3.23 avg2avr - converts NS avg to EEP avr file-format

3.23.1 Synopsis

```
------------------------------------------------------------------------------
                        Max-Planck-Institute of Cognitive Neuroscience 1997

avg2avr.pl: Version 3.9 released by mvelde, 07.10.2002
------------------------------------------------------------------------------


   *** Syntax-info ******************************************************
   *                                                                      *
   *  converts NS avg to EEP avr file-format                              *
   *  Usage: avg2avr <avg-file> [<avr-file>]                              *
   *  Options:                                                            *
   *                                                                      *
   ************************************************************************
```

3.24 avr2avg - converts EEP avr file-format to NS avg

3.24.1 Synopsis

```
------------------------------------------------------------------------------
                        Max-Planck-Institute of Cognitive Neuroscience 1997

avr2avg.pl: Version 3.6 released by mvelde, 02.09.2002
------------------------------------------------------------------------------


   *** Syntax-info ******************************************************
   *                                                                      *
   *  converts EEP avr to NS avg file-format                              *
   *  Usage: avr2avg <avr-file> [options]                                 *
   *  Options:                                                            *
   *  <cfg-file>           contains display channel layout                *
   *  --ignore "reg_expr"  describes electrodes to be ignored             *
   *  --ref "label"        of the reference electrode                     *
   *                                                                      *
   ************************************************************************
```

3.25 ev2trg - merge NS ev2 file with EEP trg file

3.25.1 Synopsis

```
------------------------------------------------------------------------------
ANT Software b.v. (c) 2001 - EEProbe script library
ev2trg 1.4 - mvelde - 2003/02/10 15:50:04
------------------------------------------------------------------------------
  ev2trg <trg> <ev2>
arguments:
  <trg>, <ev2>           trigger file, ev2 file
output:
  <trg>                  updated trigger file
                         (original file is saved as <trg>~)
```

3.25.2 Notes

This script can be used to write/modify EEP .trg files
based on input files:

- 1st arg: trg file (created by ns2riff, or extracted by cntevents)
- 2nd arg: ev2 file from Neuroscan system

output:

- updated trg file (merged events from ev2 file with header and discontinuity markers from trg file

In neuroscan system, the behavioral responses of a subject and the corresponding event codes (triggers) are not stored in one file. These event codes should be merged from the STIM system. One of its output files is in the 'ev2' format.

Example ev2 file:

```
   1      11  0    0   0.0000 1820
   2       0  2   -1   0.0000 2269
   3      11  0    0   0.0000 2948
   4       0  8   -1   0.0000 3431
   5      11  0    0   0.0000 4008
   6       0  2   -1   0.0000 4306
   7      11  0    0   0.0000 5119
   8      11  0    0   0.0000 6130
   9       0  2   -1   0.0000 6541
  10      11  0    0   0.0000 7190
  11      11  0    0   0.0000 8201
  12       0  2   -1   0.0000 8448
  13      11  0    0   0.0000 9211
  14       0  2   -1   0.0000 9389
  15      11  0    0   0.0000 10222
  16       0  8   -1   0.0000 10379
```

Requirements:
The ev2trg command script expects the behavioral data on lines that start with a number.
The trigger code is taken from the 2nd column,
The response code is taken from the 3rd column,
The 6th column is supposed to represent the sample number.
For response codes the following formula is used to encode the triggers in the trg file:

```
trigger= respcode*100 + previous_trigger
```

Example output trg file:

```
0.00200000 132
    0.000      5700  __
    3.640    245940  11
    4.538    305208 211
    5.000    340000  __
    5.896    394836  11
    6.862    458592 811
    8.016    534756  11
    8.612    574092 211
   10.238    681408  11
   12.260    814860  11
   13.082    869112 211
   14.380    954780  11
   16.402   1088232  11
   16.896   1120836 211
   18.422   1221552  11
   18.778   1245048 211
   20.444   1355004  11
   20.758   1375728 811
```

4 VIEWERS

Visualization utilities

4.1 avrview - Print avr header information

4.1.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrview 3.9   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:20 2003
------------------------------------------------------------------------------
  avrview [-l [-v]] [-t] <avr>

options:
  -t   produce a table of channel labels
  -l   produce a list of channel labels
  -v   check for valid variance information
```

4.2 cntview - Print cnt header information

4.2.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntview 3.14   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:22 2003
------------------------------------------------------------------------------
  cntview [-l] [-t] <cnt>

options:
  -t   produce a table of channel labels
  -l   produce a list of channel labels/scales
```

4.3 xcnt - EEG/MEG Raw Data Viewer

4.3.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
xcnt 3.28   (Linux 2.2.14-5.0 i686)          mvelde, Tue Feb 25 14:55:29 2003
-------------------------------------------------------------------------------
  xcnt <cnt> [<trg in>] [<rej in>] [<cfg>] [options]

options:
  -r <reref cfg>  enable rereferencing
  -f <fir>        enable FIR filtering
  -G <changrp>    load this channel group file
  -p <cmd pipe>   create a remote control pipe
```

4.3.2 Description

xcnt is a X/Motif based cnt file viewer. It allows EEG/MEG raw data inspection and interactive manipulation of trigger and rejection marks.

4.3.3 Configuration

Overview

The effective initial **xcnt** setup results from this configuration lookup/load sequence:

1. Default display-related settings are loaded from the resource database using the standard mechanism for X/Xt programs.
2. User specific display-related settings are loaded from your personal **xcnt** resource file (**Options - Save Options**), if any.
3. Project/file specific settings are read from the configuration files, which are passed as command line arguments or found automagically.

Configuration File

The xcnt-configuration file (default name "xcnt.cfg") is meant to store project-specific settings as channel scalings, the initial set of channels in the display etc. This configuration file is optional. You can create one via **File - Save Configuration As...** and adapt it to match your needs.

Resources

app-defaults/Xcnt

Some more permanent display options can be saved from within **xcnt** (**Options - Save Options**). They are stored as standard X resource lines in `~/.eep/xcnt.resources`. For xrdb gurus: these resources are written with full qualifiers and read after all other resources with lowest priority.

4.3.4 Usage

Loading Data

You can load data only via command line arguments. You must supply at least a cnt file. The other stuff is optional but recommended to load. There is no load functionality in the **xcnt** menus.

Locating Time Points

You can locate a specific time in your record with the horizontal scrollbar, with scrolling keys, by entering a start time or with a lookup function. In detail:
Scrollbar page increment/decrement, Cursor Down/Up, Page Down/Up:
+/- 90% of the displayed window
Scrollbar increment/decrement, Cursor Right/Left:
+/- 40% of the displayed window
Home/End:
record start/stop
Trigger Right/Left arrow button or key "t"/"T":
locate next/previous trigger
(comma-separated lists of max. 32 triggers are allowed in the trigger fields)
Rejection Right/Left arrow button:
set cursor #1 to the start of next/previous rejected epoch, cursor #2 to it's end

Manipulating Control Data

You can set/clear trigger points and rejection epochs with the corresponing buttons in the control area (rejections with the "r" and "c" keys too). Triggers are deleted only if the correct code is entered in the textfield and if cursor #1 is at the correct sample position. **xcnt** displays a warning if you try to leave it with unsaved changes.

4.3.5 Options

The **Options** menu offers, among others, these display control facilities:
Signal preprocessing and rendering:

- **Reference...**
  To select/calculate a reference signal which is substracted from other selected signals. The syntax for the dialog fields is analogous to the corresponding **cntreref** configuration file entries.

- **FIR Filter...**
  To select a FIR filter coefficients file which is applied to the input signals. Note that this kind of filtering has not the same sophisticated discontinuity handling as the **cntfilter** program.

- **DC offset compensation**
  If enabled, substract the mean of each displayed signal epoch before drawing (avoids drifts out of screen).

- **Draw Min/Max Bars**
  If the time axis is scaled to display long epochs, each pixel of screen would display many sample points of the signal. To reduce time/memory consumption, **xcnt** defaults to draw maximal two sample points in each screen pixel and ignores intermediate points completely.
  With this option enabled, all sample points are read/processed, and an additional vertical line is drawed to indicate the range between the minimum and the maximum value of all sample points which would fall into this screen pixel.

User interface details:

- **Cursors**
  Select between different cursor types.

- **Show Signal Values**
  Display values of one selected channel at the cursor positions in the bottom control/status area. The channel is selected by entering its label in the "Channel" field or by pressing Ctrl+Button1 at the desired signal graph.

- **Drag Signal**
  The displayed signal is moved together with the scrollbar slider. Useful only if you have not too many channels and a hellish fast computer...

- **Save Options**
  Saves the toggle states in this menu and the size and position of the main window and the channel legend. The settings saved this way are restored the next time you start **`xcnt`**, provided you do not override them in a configuration file.

The **Channels** menu offers some options for Channel Group display as described below:

- **Legend...**
  Display the **Channel Control** dialog. This dialog allows you to interactively select different colors by clicking on channel-label field. Also you can specify a custom sensitivity per channel (e.g. to change scaling on EOG channels). The changes that you make in the **Channel Control** dialog can be saved via the **File** menu: **Save Configuration As...**

- **Select...**
  To select a subset of the available channels. The entry dialog excepts regular expressions, or a comma-separated list of channel labels.

  ### Channel Groups

  Often it is not very useful to display all channels of a EEG/MEG record simultaneously. Channel groups allow the compilation of smaller channel subsets.
  Channel groups are defined in a text file "channels.cfg". At startup, **`xcnt`** either loads the file passed via the **-G** argument or tries to find this file as (in this order):

  ```
  ./channels.cfg
  ./cfg/channels.cfg
  ../cfg/channels.cfg
  ../../cfg/channels.cfg
  ```

  Two groups are created by **`xcnt`** itself:

  - **Native** - all channels in the cnt file in their original order
  - **Startup** - the channel selection from the configuration file(if any)

  For each loaded channel group you will get a button in the **Channels** menu.

channels.cfg

Normally, the classification holds only for one electrode configuration; for the electrode setups most used in a laboratory, some standard configuration files will be available. Please always use descriptive comments in your configuration, especially for non-standard configuration files.

Format

The channel groups configuration file follows the EEP standard configuration format, where the *Sections* are the identifiers of the channel groups and the *Items* indicate either a channel label or a reference to a channel group identifier.
A reference to a channel group is made by preceding its identifier with "`$`"; therefore "`$EOG`" indicates a channel group, that should be specified as
`[EOG]`
label1, label2, ...
Forward references are allowed, circular references are not allowed. Group names should be different from all available channel labels, and should consist of one word only.
A channel can only occur once within a group; repetitions will be ignored. You can specify channels that are not available in a cnt file. For example, it is possible to specify a setup for a 80-channel recording and use that for a 78-channel recording that fails two of the channels. **Therefore it is not possible to recognize errors in the channel label specification.**

Example

Consider an EEG recording using electrode positions F1, F2, F3, P1, P2, P3, EOGV, EOGH, REF, EKG, UKW. A possible grouping could be:

```
[Front]
F1, F2, F3

[Back]
P1, P2, P3

[EOG]
EOGV, EOGH

[AUX]
REF, EKG, UKW

[EEG]
$front, $back

[All]
$eeg, $eog, $aux
```

(A group 'All' is not really necessary, because the EEP modules process all channels as default behaviour - but this allows for a custom order of the channel display in the xcnt viewer)

4.4 xeog - Electrooculogram Epoch Classification

4.4.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
xeog 3.26   (Linux 2.2.14-5.0 i686)          mvelde, Tue Feb 25 14:54:07 2003
------------------------------------------------------------------------------
classification mode:
  xeog <cnt> [<cfg>] [<trg in>] [<rej in>] [<cls in>] [-a <average cfg>]
rejection mode:
  xeog -r <cnt> [<cfg>] [<trg in>] [<rej in>] -a <average cfg>
```

4.4.2 Generals

xeog offers two kinds of manual trial classification/selection according to EOG activity:

- manual prototype and trial classification for EOG artifact compensation via a regression algorithm (see the PostScript document `/opt/eep/doc/xeog.ps` for details)
- manual trial rejection (**-r** switch)

There are two types of epochs one have to classify:

EOG Events
: signal epochs highlighted by the `cntreject` utility; a representative subset of *blink* and *move* epochs ("*prototypes*") has to be selected from the events offered here; the propagation factor sets are calculated from this prototype selection; this step is not needed in rejection mode

Trials
: the parts of the EEG record one is going to use for averaging; one have to specify a correction method for each trial; this can be one of
uncorrected ("U")
use this trial "as is"
blink("B")
use the *blink* propagation factor set for the EOG compensation
move("M")
use the *move* propagation factor set for the EOG compensation
reject("R")
don't use this trial
In rejection mode you can choose only "U" or "R".

The **xeog** results in classification mode are

- EOG events classification file (*.cls)
- propagation factors file (*.pfc)
- trial classification file - a "classified" trigger file (*.trg)

In rejection mode there is only one result:

- trial rejection file(*.rej) - a rejection file with a 0..100 ms rejection mark (relative to trial) for each trial which is classified as "rejected"

Refer also to the prepared scripts in the `demo/classify` directory.

4.4.3 Requirements

xeog uses the general cnt-file Viewer `xcnt`. So you should simply try to view the raw data file at very first.
Second, you are required to have a valid `xeog` configuration file (see below).
Finally you need some data input beside the EEG record and the configuration:
a trigger file which is used for

- localization of EEG record discontinuities (to be ignored in the offered EOG event list)
- localization of trial epochs, if requested by the "-a" switch
- loading a stored trial classification, if present and requested by the "-a" switch

a rejection file which is used for

- localization of the epochs to be offered for prototype selection if no stored EOG event classification (.cls) is loaded
- initial trial rejection (as `cntaverage` would do), if requested by the "-a" switch and no stored trial classification is loaded

an EOG event classification file (.cls, optional) which is used for

- loading a stored prototype classification
- calculation of initial propagation factors from these prototypes

the `cntaverage` configuration file which is used for

- loading trial definitions

4.4.4 Configuration

Configuration File

The **xeog** configuration contains colors, sizes, scalings and channel selections for the data visualization and the EOG artifact compensation algorithm. The rejection mode usually needs an other setup than the classification mode - less channels, more epochs in the display. Accordingly, there are two example files in the sample configuration directory: `xeog.cfg` and `xreject.cfg`.

Resources

`app-defaults/Xeog`

4.4.5 User Interface

Menu Bar

File Menu

"Save..." functions for the results you can yield from **xeog**; note that you can load data only via command line. The exit function will display a warning message if unsaved classifications are found.

Classification Menu

Calculate Propagation Factors
: calculates the propagation factors from the current prototype classification; a error message is displayed if no prototypes are available
be careful to keep your current classification, the saved classification, the current factors and the saved factors in a consistent state - so the best is to select the prototypes, calc factors, save both, forget it, and deal with the trials only after this - don't hop around

Sync Trials with Rejections
: check the trial rejection windows for rejections and classify the trial "r" if rejections are found and "u" if not; this is done at startup by default if no trial classification is loadable

Sync Trials with Prototypes
: check the trial average windows for common time ranges with prototypes; set the trial classification to this prototypes; if a trial contains more then one prototype, the latter one is used

Set Rejected Trials to Blink
: nearly all rejections in trials are caused by eye blinks, so it may save some clicking if you start with blinks and remove the (few) non-blinks from this class

Set Unclassified Trials to Move
: `cntreject` usually marks only "large" EOG events. You will find a lot of your trials infected by EOG but not marked. You can either

- leave them incorrected - as you would have done before **`xeog`**
- view and classify trial by trial manually
- use the "move" factor set for all such trials since the number of low amplitude "moves" is greater than the number of low amplitude "blinks" and you cannot harm that much applying a wrong factor to low EOG amplitudes

Display Menu/Options Menu

Selects the data to be displayed and how they are displayed; no data changes; remember that you have the `xcnt` raw data viewer beside **`xeog`** if you need a more facilitated display.
The **Average Preview** contains the averaged ERP's for the channels from the **`[preview channels]`** or **`[eog channels]`** cfg section. The first calculation (started after selecting a condition toggle) may take a while. Once the averages are calculated they are updated after each classification change.

Diagram Area

You can (and have to) adjust and classify your epochs manually here. If you prefer the keyboard to the mouse you can move the drawing cursor using cursor, PgUp, PgDown keys and classify epochs pressing the matching letters. The "return" key synchronizes xcnt to the current epoch. Adjusting prototype epoch limits is possible with the pointer only.
The pointer classification methods should need no explanations - click around a little. In rejection mode there are no classification buttons available. The rejection is done by pointer clicks into the drawing and is indicated by dotted lines.

Control and Status Area

The **Scaling** group allows some basic diagram adaptions, the **Window** group gives you an impression where your displayed epochs are in the list (how much work is left). The

First: field acceps an epoch index, a condition label string or a condition shortcut string. The **Status** group shows more detailed informations about the data you process in **xeog**.

4.4.6 Classification Hints

- The *blink* estimation seems to be very precise. 20..30 good blinks, distributed over the whole record, may are enough. Don't use rare very large blinks. They are not typical but would dominate the factor estimation due to their amplitude. At the other hand don't use the small blinks even if they are frequent. They do nearly not contribute to the factor estimation so you can save your time if you ignore them.
- The same is valid for *move*, but since good moves are more rare than blinks you probably have to use all you can find...
- If you are not sure about your trial windows, set them to a larger range rather than to a shorter one. So your classification work is valid in more cases.
- Classify fast rather than careful in a first pass. Then perform a second pass in each class. You will see the misclassifications from the first pass very fast in the class context.
- Note that all evaluations you can do after manual selections will only make expressions about manipulated data, not about the measure or the system you examine! It is absolutely yours to decide whether your manipulations are valid or not.
- Start **xeog** always as a background process (with "&" appended to the command).

4.5 xavr - ERP Data Viewer

4.5.1 Contents

4.5.2 Synopsis

```
--------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
--------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
xavr 3.64   (Linux 2.2.14-5.0 i686)          mvelde, Tue Feb 25 14:54:09 2003
--------------------------------------------------------------------------
  xavr <cfg> [<cfg> ...] [<avr in> ...] [options]

options:
  -p [<dir>]     create an average file list from <dir> and its subdirs
                 if <dir> is omitted, the current directory is used
  -G <gav cfg>   create an average file matrix from the avraverage cfg
  -C <peak-file> load component specification
  -f <fir file>  enable FIR filtering of the displayed averages
  -r <reref cfg> enable referencing of the displayed averages
  -e             display ERP (default if nothing of [-edvs] is set)
  -d             display difference signal
  -v             display standard deviation band
  -s             display significance
  -t             display standard error of mean
  -D <dat file>  sensor positions for potential maps
  -N <number>    number of mesh control points for map interpolation
  -P <ps file>   write PostScript file(A4-Landscape) and exit
  -w             don't popup dialog windows at startup
  -S             slave mode, honor remote control requests
```

4.5.3 Description

xavr is the interactive exploration tool for ERP study data generated by EEP or, in other words, a powerful viewer for .avr files.
Specifically, **xavr** can

- collect the .avr files from the project tree in a more handy list or matrix of files

- rereference, average, filter, subtract ERP data "on-line"; save these processed data as .avr files
- display the data in highly configurable time series diagrams and contour plots
- create fully scaleable PostScript plots
- display data in a more interactive windows which allows to inspect signal values, to find peaks or to animate maps
- add annotation texts/pointers and include EPS illustrations in its plots

4.5.4 Configuration

Configuration File

A **xavr** configuration file is a layout description of one page of graphic objects, suitable for generating screen(Xlib) and PostScript output, and is described seperately.
Look at the files xavr*.cfg in the EEP configuration directory. Use one to start **xavr**, do your modifications and save your own version (**Edit - Configuration...**)

Resources

app-defaults/Xavr
Some more permanent dialog settings (current size/position, entered values) can be saved from within xavr (**Options - Save Settings**). They are stored as standard X resource lines in `~/.eep/xavr.resources`. For xrdb gurus: these resources are written with full qualifiers and read after all other resources with lowest priority.

4.5.5 Usage

Collect Average Files

xavr needs a list or a matrix of average files which it can offer for processing or drawing. There are several possibilities to build up the list/matrix They are located in the command line and in the **File** menu:
To prepare a **list** of files:

- Supply a list of average files in the command line. The files are collected in the list and displayed immediately. Use the shell capabilties here to build a list of files. The bash, for example, expands
  `xavr xavr.cfg su23/avr1/su23{c1,c2,c3}.avr`
  to a command with 3 average files (guess the names...).
- Supply a path to be scanned recursive for avr-files with the **`-p`** command line switch. All files found are collected in the list; nothing is displayed. Be careful with this option. It can take a while to scan a huge project tree in a remote directory and in most cases you certainly don't want to see a few hundred ERP's in one session.
- Call **File - Load Average List** from the menu. This performs the same action like the **`-p`** command line switch. If the list would become very large, the filenames can be filtered using a regular expression - typically one would use the subject code or the name of the subdirectory where the interesting averages reside.

To prepare a **matrix** of files:

- Supply a **avraverage** configuration file via the **-G** command line switch. In opposition to **avraverage** it is not an error if a listed file is not found. Maybe it's a good idea to have a "maximal" configuration file containing all subjects and conditions you expect to have and let **xavr** remove the currently incomplete rows/columns from this matrix.

- Load a **avraverage** configuration file via **File - Load Average Groups - Grand Average Configuration**. See the previous point.

- Scan the directory tree via **File - Load Average Groups - Directory Tree**.

The preparation of an average list/matrix has no effect to the display. Its only purpose is to give you access to the average files in a convenient way (without the need to surf up and down in the project directory tree). You have to use the selection dialogs to bring the data in the drawing.

Select Displayed Data

The **Select** menu gives you access to previously collected files.

- **Select - Average Files**
  The current list of average files is displayed in a dialog. You can make your selections in the list and add the selected files to the drawing by activating the **Add** button. **Set** clears the drawing before adding the selected files.

- **Select - Average Groups**
  This dialog allows the on-line grand averaging. See the example/explanation.

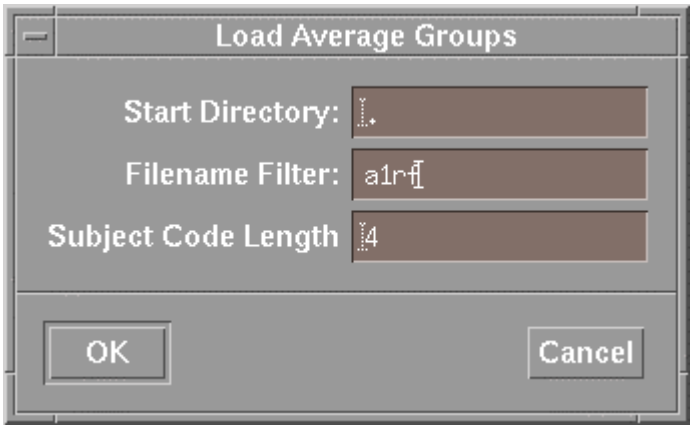
The other buttons in the **Select** menu control which information is loaded/derived from the selected avr files. Note that not each avr file contains variance information.

File - Load Average Groups - Directory Tree

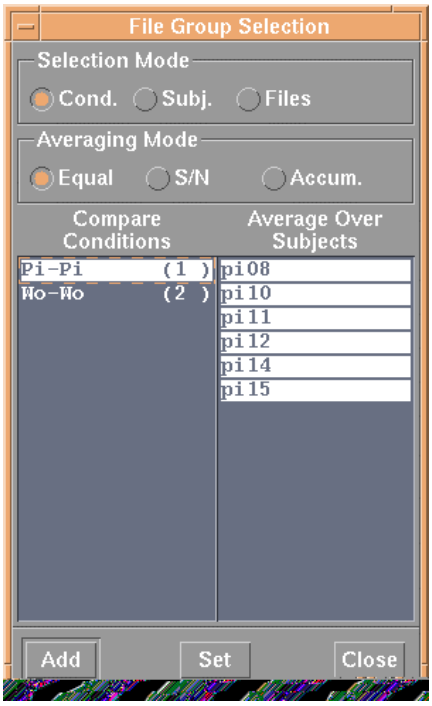
Assume the following files are present (starting in the current directory)

```
pi08/a1rf/pi081.avr
pi08/a1rf/pi082.avr
pi10/a1rf/pi101.avr
pi10/a1rf/pi102.avr
pi11/a1rf/pi111.avr
pi11/a1rf/pi112.avr
pi12/a1rf/pi121.avr
pi12/a1rf/pi122.avr
pi14/a1rf/pi141.avr
pi14/a1rf/pi142.avr
pi15/a1rf/pi151.avr
pi15/a1rf/pi152.avr
pi15/a1rf/pi153.avr
pi15/xeog/pi151.avr
pi15/xeog/pi152.avr
```

You fetch in this tree with the settings:



The last two files are skipped because their names do not match the filter expression. The available subject codes are built from the first 4 characters of the filename (**`pi08, pi10, pi11, pi12, pi14, pi15`**); the conditions from the rest of the filename (**`1, 2, 3`**)



Since all matrix elements must be filled, the condition **`3`** is removed. Finally, the subjects are sorted alphabetically.

After all, you can choose between two conditions, each with maximal six subjects. Note that the file access information is stored now for latter use. The application may (and certainly will) crash if something with the registered average files is changed!

It is possible to save this matrix as **`avraverage`** configuration file (**File - Save Average Groups**). Depending on the project size, network speed etc. it can be much faster to load such a file instead of scanning the tree again and again.

Select - Average Groups

The buttons in the upper part of the dialog controls the behavior as follows:

Selection Mode

Cond.
: Calculate one grand average per selected condition from all selected subjects (default).

Subj.
: Calculate one grand average per selected subject from all selected conditions

Files
: No grand averaging. The ERP's for all selected subjects in all selected conditions are loaded.

(If a grand averaging over exact one subject, respectively condition, is requested, the single files are loaded. No single file grand averaging is performed)

Averaging Mode

Equal

The input average files are weighted equal during grand averaging.
$x_{grand} = 1 / N_{files} * sum_{files}(x_{file})$

S/N

The input average files are weighted with their *Signal/Noise* ratio.
$x_{grand} = 1 / sum_{files}(sqrt(N\_trials_{file})) * sum_{files}(x_{file} * sqrt(N\_trials_{file}))$

Accumulate

Calculate the single trial average of all single trials in the input files.

Process Data

Using the buttons in the **Process** menu you can set some processing options which will either come into effect during the next drawing generation command (if **Deferred Processing** is set) or which are applied to the loaded files immediately.
For example, if your find an ERP too noisy and want to compare it with a low-pass filtered version, you could toggle **Deferred Processing** on, and load a .fir file. If you now **Add** the same file again in the selection dialog, the filtered ERP is displayed too. With **Deferred Processing** off, the displayed ERP is filtered immediately. This way is faster but gives no chance to compare different processing states of one file.
One should use the processing functions with care. For example, it might lead to some confusion if a rereferenced average file is rereferenced by the viewer again. For this reason, the processing options cannot be stored elsewhere, but few of them can be requested in the command line.

- **Reference**
  You can enter reference settings in the dialog fields exactly as you would write them in a **avrreref** configuration file.

- **Baseline**
  For each ERP channel, the mean of the given baseline window is calculated and substracted from the whole signal. It is merely a vertical shift, **not** the same like changing the baseline window in **cntaverage**.

- **FIR Filter**
  This point applies a Finite Impulse Response filter and requires a precalculated impulse response (*.fir). Note that this filter is useful only if all displayed ERP's have the same sampling rate.

The legend dialog indicates the processing steps performed by **xavr** with some shortcuts:

```
ref:    rereference
bsl:    baseline
gav:    grand average
fir:    FIR filter
```

The data processing steps are always performed in the following order

```
load -> [ref] -> [bsl] -> [gav] -> [fir] -> draw
```

Note that no variance/significance information is available after rereferencing, baseline correction or filtering.

Potential Maps

Generals

xavr is capable to create contour plots of the spatial field strength distribution at a spheric surface(*map*). The sphere is unwrapped into the plane to allow 2D plots with acceptable distortions. To create a map you have to include map objects and the required template data in your configuration file. The **avrinterpol** docu page contains examples.

Normally, the maps are an optional aside. The most information is still in the time series view. And this view usually fills the whole page - no space is left to include map objects. Mainly to solve this problem, **xavr** supports multiple pages. You add pages via **File - Open**, where you must load a configuration file. You toggle between the different pages via the **Window** menu or with [Shift]+Ctrl+Tab.

Spatial Information

The required spatial information (at which location, the hell, is which damned electrode) must come from a .dat file (3Dspace head digitizer). You will normally need only one .dat file per sensor configuration (one per study) - it is not necessary to use the individual positions for each subject.
The .dat file may contain valid sensor positions for channels whose data are not available or not sufficient for generating potential maps. **You have to remove these channels from the .dat file or your maps will become ugly!** (i.e. bipolar EOG channels, shorted reserve channels...)
It is possible to request additional control points where the amplitude data come from a spheric spline interpolation. This results in smoother contours but costs some computation time. The number of points controls the map interpolation as follows (*n* is the number you enter and n_{dat} is the number of valid sensor positions in the .dat file):

$n = 0$
: no interpolation at all, a value of -1e30 is assigned if no ERP channel is available for an sensor in the .dat file(this is the default)

$0 < n <= n_{dat}$
: no additional control points, interpolation for sensor positions with missed ERP channels

$n_{dat} < n$
: insert additional control points, need interpolation always

xavr can load the spatial informations via the command line switches **-D** and **-N** and in the **File - Load Sensor Positions** dialog. **xavr** shows this dialog automatically if a map must be drawn but no mesh is prepared.

Map Sequence

There is a **Edit - Create Map Sequence** dialog which allows to create a list of map objects which show the same data file at different times. The dialog controls the time step between subsequent maps and the format of the maps. The resulting maps are arranged linewise as long there is enough space for it.

Algorithm of Potential Map Generation

1. The sensor coordinates - measured at a real head surface - are projected to an unit sphere surface.

2. The sensor coordinates at the unit sphere surface are unwrapped into the plane.

3. If requested, additional control points are inserted in the plane.

4. The area spanned by the control points in the plane (the smallest convex polygon covering all points) is decomposed into a set of non-overlapping triangles where all control points are triangle corners and vice versa *(triangle mesh)*.

5. A potential value is assigned to each control point in the triangle mesh. For all points without an assigned data channel, the potential values are interpolated. This results into the *potential relief*.

6. The potential relief is intersected with horizontal planes at certain voltage levels. Each intersection of one triangle in space with one horizontal plane results in

exact one straight intersection line. The intersection lines of all triangles at one voltage level form the "isopotential polygons" for this voltage level.

7. The map you finally see is composed by the filled isopotential polygons (one color for each level) and - optionally - the outlines of the polygons(isopotential lines).

Interact with the Drawing

Zoom/Scroll

The drawing can be zoomed using the buttons/keyboard accelerators in the **Options** menu. You can also zoom a object to fit in the current viewport with the **Zoom** function from the context menu. In zoomed state, the context menu offers a **Unzoom** button which restores the previous view.
Button 1 pans the drawing - imagine you grab the sheet clicking onto it and move it below a frame. Keyboard scrolling is possible with cursor keys, Home, End, PageUp, PageDown.

Format

Most settings can be done only by editing the configuration file. But few things are possible in direct interaction. You can

- move an object within the page by pressing Ctrl-Button1 onto it and moving the frame.
- move object groups. Select multiple objects with Shift-Button1, move by pressing Button1 onto one selected object and moving the object group frame.
- delete objects via context menu, Delete key or **Edit - Delete Object(s)** menu button.
- draw arrows and text directly into the page via the **Edit - Create...** menu buttons.
- edit text object attributes via the **Edit..** button from the context menu
- in preparation to a black/white plot, set all signal trace colors to black and the potential map palette to "gray" using the **Options - Black/White** menu button. (**`xavr`** automatically assigns different dash patterns to signal traces with the same color.)
- assign new colors to signal traces. Click at the small sample line in the legend dialog and select a color from the menu that pops up. The **Other...** button allows to set the linewidth and the dash pattern too. The new parameters are assigned to the selected legend slot, not to a specific avr file. You get rid of these settings with the **Options - Discard Trace Styles** menu button.

Inspect

Diagram and map objects offer an **Inspect** button in the context menu. The inspect function for time series diagrams allows to look for peaks or to show signal values. For potential maps it allows to change the displayed time slice which gives nice animations of the potential distribution...

Components (peak scoring)

Amplitude, latency data extraction

In the **Signal Inspector** window you can

- move the mouse cursors independently in two curves using the left button
- add the amplitude/latency data to the **Signal Values** listing, using
    - the <Space> bar
      this adds the amplitude/latency data, and other waveform parameters to the listing, related to the selected cursor (last mouse action),
    - the right-mouse button; this opens a popup window showing the pre-defined peak labels. Select one, and parameters will be added to the listing, including the label (in the last column).
- click in the legend area to move the cursor to another waveform (if available); alternatively, you can use the **Ctrl-Up/Down** keys to walk the list of waveforms.

Use the **Sort** button in the **Signal Values** window to sort the entries. This is useful when you have to score peaks on a lot of curves, and want to review possible double entries. The data are stored in so-called "peak" files ***.pk**. The signal values are not stored automatically, you must use the **Save** button. The file thus created can be reloaded in a later session of **`xavr`** via the **File** menu, **Load Components File...**

Remarks

The scoring of peaks as described above, can be visualized in the main graphic area of the **`xavr`** viewer. The display is only updated after pressing on the **Apply** button of the **Signal Values** window. You can make this window visible at all times via **Options - Show component window...**.

Display/scoring of a peak requires the section **`[components]`** in the configuration files, and an entry **`component lab:name`**, where "name" is the label as it appears in the right-click popup menu in the **Inpect** window; this is also the label that is displayed in the curves.

- You can learn more about the display options in the description of the component definition in the configuration file.

PostScript Plot

The drawing can be saved as a PostScript page description via the **File - Print PostScript** dialog. It offers two targets for the PostScript output: files and commands. Files are simply stored to disk. Print commands are executed in a shell subbrocess. The standard input of this process is fed with the **`xavr`** PostScript output, error and status output is catched by xavr.

There are some prepared print and preview commands available(configurable via resource file). Note that not each of this commands works at each machine. Note also that **`xavr`** is blocked until the shell subprocess terminates.

A special print command is **`avrps`**. This is a shell script which is intended to generate an A4 landscape PostScript screen preview at each machine. This script is also used as the **File - Print Preview** menu command.

The **Current View** option in the dialog allows to print the current content of the **`xavr`** window, scaled to fit in the output media. This option is useful if you need a selected piece of the drawing as enlarged plot or as EPS illustration for later use - just zoom/move/resize your window to show the desired area and send it out.

4.6 xavr - Configuration File

4.6.1 Contents

4.6.2 Introduction

xavr introduces an extension of the common EEP configuration format. Most items are split into "subitems" as follows:

```
<label>  <key&gt:<value>  <key&gt:<value> .....
```

This style allows a very compact representation of the large amount of information which is needed for **xavr**.

Each **xavr** configuration file describes one page which contains several graphic elements (*objects*). The page coordinate system is in millimeters; point (0;0) refers to the lower left corner of the page, the positive x-axis points to the right, the positive y-axis to the upper side.

Available objects are:

diagram
: cartesian diagram for time series data with extensions for significance plot

map
: contour plot for spatial field distributions

maplegend
: color palette for potential maps

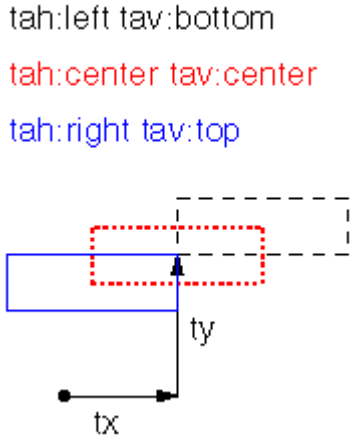
text
: a piece of arbitrary text

line
: a straight line connecting two points of the page, optionally with an arrow head

epsf
: Encapsulated PostScript File

"diagram", "map" and "maplegend" are complex objects, described by a lot of parameters. Most of the parameters are common for multiple, if not all, object instances. Therefore, the objects refer to *templates* which contain the common parameters and have only few own attributes. For example, each diagram object refers to a diagram template which contains axis scalings, sizes, linewidths, fonts..., but each diagram object has its own page coordinates and its own label.

Furthermore, major parts of complex graphic objects can be described by the graphic *primitives* "line" and "text".



For example, a x-axis of a cartesian diagram is a horizontal line with an optional text label(the unit); a axis division is also a line (the ticmark) with an optional text label (the axis value at this particular ticmark). **xavr** uses this sort of graphic abstraction in many places and provides a powerful and uniform configuration mechanism for the primitives.

4.6.3 Primitives

Linestyles

All lines to be drawn are described by the following set of parameters:

```
lcol:  color (a valid X color name)
lw:    linewidth (in mm)
ldash: dash pattern (one of the predefined strings "solid",
       "dashed", "dotted", "ldashed", "dashdot", "dashdotdot")
lcap:  how to draw line endpoints ("butt", "round", "projecting")
```

The linewidth parameter has effect for the screen drawing only if **Options - Draw Wide Lines** is enabled. Normally, all screen lines are 1 pixel wide (much faster). The PostScript output shows always the requested linewidth.

Textstyles

All text elements to be drawn are described by the following set of parameters:

```
tcol:  color (a valid X color name)
tfont: font (a valid PostScript font name - case sensitive!)
tsize: fontsize (in mm)
tx:    horizontal offset of text anchor point from text origin (in mm)
ty:    vertical offset of text anchor point from text origin (in mm)
tah:   horizontal text alignment relative to anchor point
       ("left", "right" or "center")
tav:   vertical text alignment relative to anchor point
       ("bottom", "top", "center")
thide: to enable/disable the text output ("off" or "on")
```

The figure shows how the text box anchor point is displaced from the text origin coordinates and three of the nine possible alignments of the text box relative to its anchor point.

Fonts and fontsizes are evaluated exactly in PostScript output only, screen text is in (slightly different) screen fonts. Be exact in the fontname spelling, errors will occur at PostScript interpretation time only! Valid standard fontnames are:

```
Times-Roman
Times-Italic
Times-Bold
Times-BoldItalic
Helvetica
Helvetica-Oblique
Helvetica-Bold
Helvetica-BoldOblique
Courier
Courier-Oblique
Courier-Bold
Courier-BoldOblique
```

The `thide` attribute is somewhat special: it might be useful to have a text present (for object size calculations) but not to draw/plot it. Don't worry too much about it.

ERP Attribute Placeholders

In some contexts it is possible to use placeholders which are dynamically expanded with values from the actually loaded files. Valid placeholders are:

```
&C  - condition label
&N  - number of trials/averages
&M  - mean number of trials of the input averages
```

```
      valid for xavr's on-line calculated grand averages only
  &R  - number of rejected trials
  &F  - filename
  &Sx - subject code (first x characters of the filename)
  &Tx - condition code (after first x characters of the filename)
  &I  - index of the file in the current selection
  &L  - channel label
```

These placeholders are also used to specify data selections. This is done using expressions of the form:

```
sel:<placeholder(s)>=<literal>
sel:<placeholder(s)>=/<regex>/
```

In conjunction with a diagram object, for example

```
sel:"&I&L=/^[14]F/"
```

means that this diagram displays the signals from the first and the fourth loaded file which have a channel label starting with "F" ...

4.6.4 Templates

4.6.5 Global Defaults

Example

```
[global defaults]
page        width:297 height:210 grid:5
frame       type:eep hmargin:20 vmargin:9
trace_line  lw:0.25 lcol:black ldash:solid
scale_line  lw:0.15 lcol:black ldash:solid
text        tfont:Helvetica tsize:2
arrow       lw:0.40 lcol:black ldash:solid ang:30 len:3.5
```

This block describes a A4 landscape page with a 5 mm snap grid. A thin black frame line is generated. All signal data traces have a width of 0.25 mm. Other lines (diagram axes, axis divisions) are 0.15 mm wide, black and solid. Text is in Helvetica 2 mm. The `arrow` line defines the parameters which are used for interactively created arrows.

Reference

```
page
  width:  page width (in mm)
  height: page height (in mm)
  grid:   the snap grid spacing for interactive object placement (in mm)
frame
  type:   "eep" (a thin black frame) or "mpi" (a thick dark-blue frame)
  vmargin:
  hmargin: space between sheet edge and frame line
```

The frame settings are evaluated in PostSript output only. The screen just shows a dashed line to indicate it.

4.6.6 Diagram Templates

A diagram template is described by a large pile of parameters. Fortunately you'll never need to write them all, you can rely on a lot of defaults and need to supply only the differing settings. The following sections explain the full parameter set for reference purposes. The best approach is to start up with one of the sample configurations (xavr*.cfg) and to play around with this stuff. There are also some commented examples.

Example

```
[diagram template]
name       default
field      width:21 height:14
label      tsize:2 fmt:"%s"
time_axis  unit:s   scale:1 pos:0.0 min:0.3  max:0.6
ampl_axis  unit:uV  scale:1 pos:0.0 min:-5.0 max:5.0
time_div   step:0.2  len:1 fmt:"%.1f"
ampl_div   step:1 len:1 fmt:"%.0f"
```

Reference

```
field
  width:  width of the area covering all diagram parts (in mm)
  height: height of the area covering all diagram parts (in mm)
  sel:    channel/file selection
label
  fmt:    printf format string for diagram label output("%s")
  t...:   textstyle parameters for the label string plot
ampl_axis/time_axis
  unit:   unit string (e.g "uV" or "fT")
  scale:  size of the unit above in terms of xavr base units
          (s for time axis, avr data units for amplitude axis)
  min:    value of left or lower edge of diag (in axis units)
  max:    value of right or upper edge of diag (in axis units)
  pos:    axis position (in units of the other axis)
  l...:   linestyle parameters for the axis line
  fmt:    printf format string for unit string output("%s")
  t...:   textstyle parameters for the unit string
          (text origin is the upper/right axis end point)
ampl_div/time_div/ttest_div
  step:   spacing between division lines (in axis units)
  pos:    position of a single division/marker line (in axis units)
  len:    length of the division lines (in mm, 0 means whole diagram)
  align:  alignment of division lines relative to axis line
          ("center", "left", "right", "top", "bottom")
  l...:   linestyle parameters for the division line
  fmt:    printf format string for (float) value output ("%.3f")
  t...:   textstyle parameters for the value string
          (text origin is the lower/left endpoint of the division line)
ttest_line
  state:  enable/disable the running ttest prob. diagram("on" or "off")
  height: height of the diagram (in mm)
  min:    start point for probability drawing(in time axis units)
  max:    end point for probability drawing(in time axis units)
ttest_grey
  state:  enable/disable the prob. greyscale bar ("on" or "off")
  height: height of the greyscale bar (in mm)
  min:    start point for probability drawing(in time axis units)
  max:    end point for probability drawing(in time axis units)
  t...:   textstyle parameters for the greyscale legend of the
          "scale" template (evaluated in PostScript only)
```

The default label position is (0;0) (lower-left corner of the diagram). The x-offset is fieldwidth, the y-offset is fieldheight, and the alignment is top-right. This means that the label appears in the upper right corner of the diagram field.
Axis division values are not permitted to cover axis lines and the spacing between division lines must be larger than 3 * division linewidth. Otherwise, nothing is drawn. The probability axis of the semilogarithmic t-ttest diagram is hard-coded (min:1 max:0.001). Linestyle parameters are inherited from time/ampl axes. Divisions lines (`ttest_div`) are possible with `pos` only, not with `step`.

4.6.7 Map Templates

Example

```
[map template]
name         default
field        width:60 palette:bluered
time_axis    unit:s  scale:1 fmt:"&C: %.3f .. %.3f %s" tsize:2 ty:5
ampl_axis    unit:uV scale:1 min:-5.5 max:5.5
ampl_div     step:1 lw:0.2
outline      lw:0.2
sensors      size:1
legend       height:5 width:40 fmt:"%+.1f"
```

Reference

```
field
  width:    width/height of the square covering all map parts (in mm)
  palette:  color palette for amplitude (z-axis) encoding
            (the name of an own map palette or one of
             the predifined palettes "bluered","grey","mapview",
             "spectrum")
  sel:      file selection
  mesh:     for debugging; toggle drawing 0f the triangle mesh (0 or 1)
time_axis
  unit:   time unit string ("s" or "ms")
  scale:  size of the unit above in terms of xavr base units (seconds)
  fmt:    format string for map label output
          (printf placeholders for tmin, tmax and the time axis unit and
          ERP attribute placeholders)
  t...:   textstyle parameters for the label string
          (text origin is the map object position)
ampl_axis
  unit:   amplitude unit string ("uV" or "fT", "z" enables
          z-standardization)
  scale:  size of the unit above in terms of average data units
  min:    amplitude value assigned to the lower end of the color scale
  max:    amplitude value assigned to the upper end of the color scale
ampl_div
  step:   the amplitude range represented by one color
  l...:   linestyle parameters for the iso... lines
          (0 means no isolines)
outline
  l...:   linestyle parameters for the outline polygon
sensors
  size:   diameter of the dot which represents the sensor pos. (in mm)
  fmt:    printf format string for sensor label plot ("%s")
  t...:   textstyle parameters for sensor label plot
          (text origin is sensor position)
  l...:   linestyle parameters for sensor dot plot
legend
  height: height of the color scale legend
  width:  width of the color scale legend
  fmt:    printf format string for min/max amplitude values ("%.1f")
  t...:   textstyle parameters for min/max amplitude values
```

4.6.8 Map Palette

Description

A **[map palette]** section describes a sequence of colors which can be used in **[map template]** sections.
The first line defines the name of the palette. This line is optional. A palette without a name is used for all maps.
It is followed by a list of <color> [<coordinate>] pairs, where <color> can be any valid X color specification and the optional <coordinate> is the relative position of the given color (0.0 - minimum, 1.0 - maximum) in the generated palettes.
The actual palette colors are computed by linear interpolation of the red/green/blue components between these given supporting points. (Eventually, your display hardware gives you only approximations of the requested colors. You need to look at the final PostScript output then.)

Examples

```
[map palette]
name strange
rgb:00/f0/00  0.0
rgb:00/00/f0  0.2
rgb:00/00/ff  1.0

[map palette]
name bluered
blue
white
red
```

4.6.9 Legend Template

The **[legend template]** section instructs **xavr** to create line and text objects which together form a legend. This way you have a dynamic legend which explains the current view without any interaction. This can be disabled in the **Options** menu.

Example

```
[legend template]
headline    fmt:"Legend:" x:230 y:100 tsize:4.5
avr_entry   fmt:"&C (n=&N)"           tsize:3.5 len:6
diff_entry  fmt:"&s4 &c - &S4 &C"     tsize:3.5 len:6
```

Reference

```
headline
  fmt:    legend headline string
  x:      position of the legend headline text
  y:      position of the legend headline text
  dh:     horizontal distance between entries (in mm), default:0
  dy:     vertical distance between entries (in mm), default:-1.4*fontsize
  t...:   textstyle parameters for the legend strings
avr_entry/diff_entry
  fmt:    legend format string, subject to
          ERP attribute expansion
  len:    length of the sample line segment (in mm)
  t...:   textstyle parameters for the legend strings
```

4.6.10 Style Template

The **[style template]** assigns individual linestyle parameters to selected trace lines in time series diagrams. It overrides the `trace_line` parameters in the **[global defaults]** section and the color which is stored in the avr files. It is overridden by the interactive trace style settings.

Example

```
[style template]
trace  sel:"&C=rare"     lcol:red  ldash:dotted
trace  sel:"&C=/^freq/"  lcol:blue ldash:solid lw:5
```

This would plot all ERP's with the condition label "rare" using red-dotted lines and all ERP's with a condition label starting with "freq" using 5mm-thick blue-solid lines.

4.6.11 Components

The **[components]** defines the types of components which will be shown. Components are extra labels which can be added to an diagram by using the signal inspector or loading a component file.

Example

```
[components]
component    lab:P2 align:top len:3 tfont:.5
component    lab:P1 align:neg multiple:on len:2 lcol:red
component    lab:N1 align:bottom multiple:off len:5
```

Reference

```
lab:      the label of the component.
align:    the alignment of the component. Can be top, bottom, neg, pos.
          top: the label will be at the top.
          bottom: label will be at the bottom.
          neg: label will be on the side with the lowest coordinate.
          pos: label will be on the side with the highest coordinate.
          standard is neg.
len:      the length of the line.
multiple: when on, a diagram can contain multiple components of the same type.
          standard is off.
l...:     the linestyle parameters for the line.
          the lcol has as standardvalue "dynamic". This means it uses the settings
          of the line to which the component belongs. When lcol is dynamic the ldash
          will automaticly use the explicitly set options for the line in the legend
          in xavr and not the value put in the configurationfile.
t...:     the textstyle parameters except for tx, ty, tav, tah.
          standard is taken from [global defaults] text for tfont, tsize and thide.
          for tcol the standard is "dynamic" which means it uses the color of the
          line to which the component belongs.
```

4.6.12 Objects

Description

The **[objects]** section defines the drawing elements. If you want to see a diagram/map/line... in your drawing, you need to add a line to this list.
Since **xavr** allows interactive modifications of objects, it has to reflect such changes into the list. For this reason, whenever you have made such changes and enter the **Edit - Configuration...** dialog, **xavr** looks for the **[objects]** keyword in the file, throws

away all stuff that follows and creates a completely new object list from the current internal state.
Therefore, the **[objects]** section has to be the last section in the file and should contain no special user formattings or comments!
Some object lines are generated as comments. Currently, this are the objects which form the legend. Since they are generated dynamically at each data loading time, it is not necessary to load them at layout creation time. If you are not satisfied with the automatic legend, you can comment out the **[legend template]**, uncomment the legend objets and adapt them as needed. After this you have a static legend in the drawing.

Example

```
[objects]
text       x:148.00 y:185.00 txt:"Page Title" tsize: 8.00 tah:center
diagram    x: 38.00 y:126.00 tpl:default lab:FZ
diagram    x: 38.00 y: 58.00 tpl:default lab:PZ
maplegend  x:176.34 y: 26.46 tpl:default
map        x:172.00 y:108.00 tpl:default tmin: 0.200 tmax: 0.400 file:1
map        x:172.00 y: 38.00 tpl:default tmin: 0.200 tmax: 0.400 file:2
```

Reference

```
diagram
  x:
  y:      coordinates of the lower left corner of the diagram (in mm)
  tpl:    name of the assigned diagram template
  lab:    diagram label, the diagram will show the data for this channel
          if no sel:... is specified
  sel:    file/channel selection, overrides the template settings
map
  x:
  y:      coordinates of the lower left corner of the map field (in mm)
  tpl:    name of the assigned map template
  tmin:   start of the time slice for this map (in time_axis units)
  tmax:   stop of the time slice for this map (in time_axis units)
  file:   file selection
          (1 means: this map will show the first loaded/calculated file,
           2 means: this map will show the second loaded/calculated file,
           ...
           0 means: this map will show the last loaded/calculated file)
  sel:    file selection, overrides file: and the template settings
maplegend
  x:
  y:      coordinates of the lower left corner of the
          map legend color bar(in mm)
  tpl:    name of the assigned map template
line
  x:
  y:
  x2:
  y2:     coordinates of the line end points (in mm)
  l...:   linestyle parameters
  ang:    opening angle of the arrow head triangle (in degrees)
  len:    length of the arrow head triangle (in mm)
text
  x:
  y:      coordinates of the text origin point (in mm)
  txt:    text string to be drawed
  t...:   textstyle parameters
epsf
  x:
  y:      lower left corner of the illustration field (in mm)
```

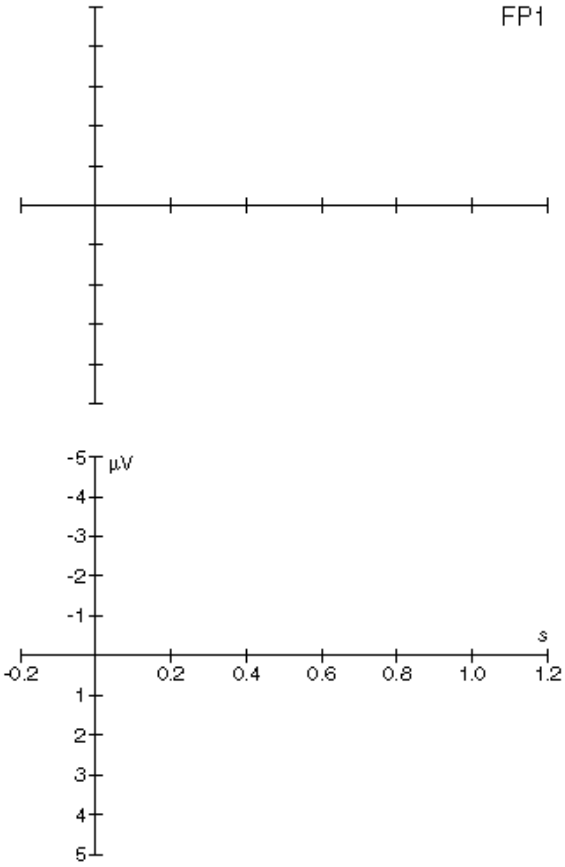
```
w:      width of the illustration field (in mm)
h:      height of the illustration field (in mm); unspecified means
        fixed scale, height is computed from width and the EPS file
        bounding box
rot:    rotation angle for the included illustration (in deg);
        unspecified means 0 or -90 (%%Orientation: Landscape)
name:   name of a strictly conforming EPS file
```

Diagram template example

The following sample pictures have passed a long chain:

xavr -> PostScript -> xpsview image (2:1 scale) -> xv grabbed image -> gif file -> your browser

You may find that some information was lost on the way...




```
[diagram template]
name       default
field      width:40 height:30
label      tsize:2
time_axis  unit:s   scale:1 pos:0.0 min:-0.2
max:1.2
ampl_axis  unit:uV  scale:1 pos:0.0 min:5.0
max:-5.0
time_div   step:0.2  len:1
ampl_div   step:1    len:1

[diagram template]
name       scale
label      thide:on
time_axis  fmt:"%s"
ampl_axis  fmt:"%s"
clear_div
time_div   step:0.2  len:1  fmt:"%.1f"
ampl_div   step:1    len:1  fmt:"%.0f"

[objects]
diagram  lab:FP1    x: 38.00 y:144.00
tpl:default
diagram  lab:SCALE  x: 38.00 y:110.00
tpl:scale
```

This is the most common "plain" case - the "default" data diagrams are very simple (you have to arrange 60 and more of them at one page). There is no axis text and only one division per axis - in the amplitude (here: voltage) axis it is one line per µV, which results in 11 ticks for the whole 10 µV axis. Note that you reach a "negatives up" behavior by assigning the voltage minimum to the axis maximum and reverse. (BTW, you cannot have a reversed time axis in the current version).

Usually we need at least one diagram with axis descriptions. Therefore a additional template named "scale" is defined. This template, as any template, inherits all settings from the "default" template and needs only the differing settings. The differences are:

- the new name "scale", needed to identify the template
- we want not to plot a diagram label, therefore we set `label thide:on`
- the axes should plot their units, so we add the `fmt:"%s"`, what means that the axis `unit:` text is plotted
- the divisions are new (we want to plot divisions with values now) so we clear the old divisions and create identical new ones with a format string for the values added.

4.7 xavr - Keyboard/Pointer bindings

Beside of the standard Motif keyboard/pointer methods and the menu accelerators, there are special bindings for some **`xavr`** elements.
Note that this topic is highly (mis)configuration dependent... Usually, "Button 1" refers to the left mouse button, "Button 3" to the right one. The "Delete"/"Backspace" keys have their own long story on Unix systems and will often not work as expected.

Main Graphic Area

Key	Action
Cursor Up/Down/Left/Right	scroll drawing by 40% of the displayed width/height
PageUp, PageDown, Home, End	scroll drawing to its boundaries
Ctrl+Tab, Shift+Ctrl+Tab	switch to next/previous page
Button 1 Motion	pan page or move selected objects
Shift+Button 1	select/deselect object
Shift+Button 1 Motion	select objects within a box
Ctrl+Button 1 Motion	move object
Button 1 in "text mode"	popup the text object creation dialog
Button 1 in "arrow mode"	"rubberband" drawing of an arrow line
Esc	deselect all objects, leave text or arrow mode, refresh drawing
Delete	delete selected objects
Button 3	popup the context menu

Average Selection Lists

Button 2	select a single item
Insert	select all items
Button 3/Delete/Backspace	deselect all items
I (the "i" key)	invert selection

Configuration Editor

Ctrl+S	popup the "Save Configuration" dialog
Ctrl+Z	undo last text change
Shift+Ctrl+Z	redo last undo

Legend dialog

Button 3 at the sample line	popup the line attributes menu
Button 3 at the legend text entry	popup the ERP context menu

"Signal Inspector" Graphic Area

[Shift]+Ctrl+Left/Right	locate previous/next minimum/maximum
Ctrl+Up/Down	assign the current cursor to previous/next trace in list
Return, Enter or Space	append the current data point to a value listing (Component window)
Button 3 in the waveform area	Open the component list (if available in the current configuration - display page)

"Map Inspector" Graphic Area

Button 1 Down/Motion	show polar coordinates and an interpolated value at current pointer position

5 DATA MANAGEMENT

Change/control the recording files

5.1 avrchannels - Remove or rename channels

5.1.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrchannels 3.5   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:18 2003
-------------------------------------------------------------------------------
  avrchannels [-r] <cfg> <avr in> [<avr in> ...]
  avrchannels -s <cfg> <avr in> <avr out>

options
  -s      single file mode, (output name possible)
  -r      replace mode, output overwrites(!) input
```

5.1.2 Configuration

```
; example configuration for avrchannels

; list of channels which should be not copied to output
[remove]
x3, x4, x5, eogv, eogh

; list of channels which should be renamed in output
[rename]
CZ CZZ
FZ FZZ
F4 F8
F8 F4
```

5.2 avrdown - ERP Data Downsampling

5.2.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrdown 3.5   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:19 2003
-------------------------------------------------------------------------------
  avrdown -f <freq> <avr in> <avr in> ...
  avrdown -s -f <freq> <avr in> [<avr out>]

options:
  -f <freq>  new sampling frequency in Hz
  -s         single file mode (output name possible)

files:
  foo.avr  ->  foos.avr
```

5.3 avrmerge - Merge ERP data

5.3.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrmerge 3.7   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:20 2003
-------------------------------------------------------------------------------
  avrmerge -o <avr out> [-l <label>] [-c <color>] <avr in> <avr in>...

options:
  -o <avr out> output filename
  -l <label>   conditon label for avr out
  -c <color>   condition color for avr out
```

5.4 cntcat - Raw Data Concatenation

5.4.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntcat 3.16   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:10 2003
------------------------------------------------------------------------------
  cntcat [-d] [-t] <src cnt> <src cnt> ...  <dst cnt>
options:
  -d     accept different channel layouts, use the common subset
  -t     use trigger files, rather than internal trigger list

files:
  foo1.cnt  ->  foodst.cnt
  foo2.cnt      foodst.trg
  foo3.cnt
  ...
  foo1.trg      (trigger files only used with option -t)
  foo2.trg
  foo3.trg
  ...
```

5.4.2 Description

cntcat concatenates cnt files. Normally, it requires exactly the same channel sequence in all input files. The `-d` switch can override this if you are really sure that you want to throw away all channels which do not occur in all input files.
Scaling factors are taken from the first input file. The data from other files are rescaled, if necessary.

In dst cnt, the former sample #0 of each src cnt is marked by a discontinuity trigger.

By default, the events as stored in the internal event lists are copied to the output file, using the correct offsets. However, as a special feature it is also possible to copy new events to the output file using the `-t` switch. However, this requires the presence of all related TRG (trigger) files, with identical basename as the CNT files.

5.5 cntdown - Raw Data Downsampling

5.5.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntdown 3.12   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:10 2003
------------------------------------------------------------------------------
  cntdown <source cnt> [<dest cnt>] -s <skip>|-m <skip>|-f <freq>

options:
  -s <skip>  write each <skip>th source sample to dest
  -m <skip>  write each mean of <skip> source samples to dest
  -f <freq>  resample to <freq> Hz via linear interpolation

files:
  foo.cnt  ->  foos.cnt
               foos.trg
```

5.5.2 Description

cntdown changes (decreases) the sampling rate of .cnt-files. This allows to unify data which were recorded at different sampling rates. Another useful application is just to reduce the amount of stored data (and thus all evaluation runtimes). For example, a 1000Hz EEG/MEG could be downsampled to 250Hz or slower if you look for (10..20)Hz components only.

There are three algorithms implemented:

pull mode
: each *skip*-th sample point is loaded from input and copied to output

mean mode
: the mean of each *skip* sample points is calculated from input and stored in output

interpolation mode
: output points comes from linear interpolation between the previous and the next input point (this mode could "upsample" too, but such a process makes no sense to me)

According to the recommended EEP usage rules (on page 16), a **cntdown** run should be one of the first steps in an evaluation. One can avoid a lot of trouble with invalid or inaccurate control data files (.trg, .rej) this way...

Consider also *Shannon*'s theorem, filtering (section 6.12, on page 121) and other signal processing fundamentals.

5.6 cntepoch - Cut raw data records

5.6.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntepoch 3.13   (Linux 2.2.14-5.0 i686)      mvelde, Tue Feb 25 14:54:09 2003
------------------------------------------------------------------------------
  cntepoch <source cnt> [<dest cnt>] [<cfg>] [<trg in> [<trg out>]]

files:
  foo.cnt    ->  fooe.cnt
  foo.trg        fooe.trg
```

5.6.2 Configuration File

```
;------------------------------------------------------------------------------
;Example configuration file for cntepoch
;------------------------------------------------------------------------------

; IMPORTANT NOTE:
; cntepoch should normally NOT be used
;
; The only useful application is to remove longer session pauses from your
; record (if you didn't stop recording itself for some reason).
; This should be done immediately after recording and before archiving.
;
; Otherwise, cntepoch will cause trouble: invalid .trg, .rej, .cls files,
; edge artifacts in subsequent signal processing, the need to
; store the epoched copy of the cnt AND the original cnt...


StartTrigger:         ; if this trigger is found in cnt or trg file,
255                   ;   cntepoch will start to copy CNT-data to the output
file
StopTrigger:          ; if this trigger is found in cnt or trg file,
254                   ;   cntepoch will stop to copy CNT-data to the output
file

;254, 12, 3           ; you can use a comma-separated list of triggers
                      ; for both start and stop
                      ;   (in one line!)

                      ; The default trigger codes for epoching are '255' and
                      ; '254'. These triggers may be set (manually) using xcnt
                      ; However, they may also be set
                      ; automatically during the recording session

FilterLength          ; cntepoch extends the copied sections to avoid edge
41                    ; artifacts in latter evaluations
                      ; 41 is the default which means that 20 additional
                      ; samples are copied before start and after end
                      ; (a 41 point FIR filter needs x[-20]..x[+20] to
                      ; compute x[0])
```

5.6.3 Description

cntepoch looks for *StartTrigger* in the input trigger list. After one is found, it's position is reduced by *int(FilterLength/2)* samples. The data are copied from src to dst from here up to *Position(StopTrigger) + int(FilterLength/2)*. The first sample of each copied epoch is marked by a discontinuity trigger.

If the *FilterLength/2* offset exceeds the recorded data range, a constant signal with the first, respectively the last, sample value is assumed.

The codes which are declared as *StartTrigger* or *StopTrigger* are consumed by **cntepoch** and are not present in the output trigger lists.

If no configuration file is specified in the command line then hard coded default settings according to the sample configuration above will come into effect.

5.6.4 IMPORTANT NOTE

cntepoch should normally **NOT** be used.

The only useful application is to remove longer session pauses from your record (if you didn't stop the recording itself for some reason). This should be done immediately after recording and before archiving.

Otherwise, **cntepoch** will cause trouble: invalid .trg, .rej, .cls files, edge artifacts in subsequent signal processing, the need to store the epoched copy of the cnt AND the original cnt...

5.7 cntevents - Extract control data from signal archives

5.7.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntevents 3.8   (Linux 2.2.14-5.0 i686)      mvelde, Tue Feb 25 14:54:11 2003
-------------------------------------------------------------------------------
  cntevents [-p][-s] <src cnt> [<trg out>]
  cntevents [-p] -m  <src cnt> [<trg in> [<trg out>]]
  cntevents -r <src cnt> [<rej out>]
options:
  -s     skip discontinuities("__" and "Rs")
  -m     merge discontinuities into <trg in>
  -r     convert EEP 2.0 rejection marks to ASCII-list
  -p     pipe mode; write triggers to stdout, no log output
files:
  foo.cnt  ->  foo.trg
```

5.7.2 Description

Each cnt-file stores triggers as reference points in the time axis. Usually, they are recorded during the EEG session. The modules do not use this internal triggers (they use an external ASCII file *.trg) but maintain the internal table if the time axis is changed.

`cntevents` allows to convert the internal triggers to the external ASCII file.

It can be quite expensive to handle unexpected reset or discontinuity triggers in processing scripts. To work around this, **`cntevents`** offers the **`-s`** and **`-m`** options. Create a "clean" file, modify it, and finally include the resets/cutmarks. **Note that `cntdetrend`, `cntaverage`, `xeog`, `cntfilter` and `cntreject` need the resets/cutmarks for their correct behavior!**

5.8 cntmono - convert 2 monopolar channels to 1 bipolar

5.8.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntmono 3.12   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:21 2003
-------------------------------------------------------------------------------
  cntmono <source cnt> [<dest cnt>] <cfg> [-r]

options:
  -r  reverse mode

files:
  foo.cnt  ->  foob.cnt   (forward)
  foo.cnt  ->  foom.cnt   (reverse)
```

5.8.2 Description

`cntmono` converts 2 monopolar channels to 1 bipolar channel; overwrite first mono.

`cntmono` should normally **NOT** be used: this program is obsolete; better use `cntsetup`.

5.9 cntsetup - CNT File Hacker's Friend

5.9.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntsetup 3.7   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:11 2003
------------------------------------------------------------------------------
  cntsetup [-i <regex>|-x <regex>] <operation> <src cnt> <dest cnt>
  cntsetup -c <expr> <src cnt> <dest cnt>
  cntsetup -R <src cnt> <dest cnt>

options:
  -i <regex>     select channels for <operation>
  -x <regex>     exclude channels from <operation>
  -c <expr>      compute a channel
  -R             try to "repair" cnt file checksum errors
operations:
  -s <factor>    scale scaling factors
  -d <factor>    scale raw data values
  -u <unit>      set unit strings
  -l <label>     set channel label
  -r             remove channels
  -f <freq>      set sampling frequency
```

5.9.2 Description

`cntsetup` allows to change the channel list, the per channel or per-file attributes in cnt-files. This program has much of a "hacker tool", it is rarely used and not well-tested. - so check its output carefully before deleting the original files!

Compute/Insert Derived Channels (-c option)

The **`-c`** allows to specify an expression how to compute a signal from other signals and constants. If the target channel is already present, it is overwritten; otherwise, an additional channel is created in the output cnt. **`<expr>`** can have the following forms (to be extended with a real expression parser...):

```
  <label> = <label> - <label>
  <label> = <label> + <label>
  <label> = <const> * <label>
```

It is possible to specify multiple, comma-separated expressions. Note that whitespaces are required to separate labels and operators (since '+' and '-' are legal characers in channel labels). This means also that you must quote the whole expression in shell commands. Typical usage is to insert bipolar channels from monopolar channels or to insert an artificial reference channel.

```
  cntsetup -c 'VEOG = V+ - V-, HEOG = H+ - H-' in.cnt out.cnt
  cntsetup -c 'A1 = 0.0 * A2' in.cnt out.cnt
```

"Repair" Corrupted Files (-R option)

If an evaluation fails with something like `cnt: checksum error...` or `raw3: unknown compression...`, this indicates that the file is corrupted and the data cannot be decoded. As a last resort, if no correct copy/backup of the file is available, you can use this option. The full file is read, the corrupted epochs - typically one second - are replaced with zeroes and surrounded by discontinuity triggers ("__").

Note that the corrupted signal epochs are lost in any case! And it will not help at all in some cases, where the failure is in the internal management tables and not only in the signal data block. So it's really better to have a backup at hand...

Manipulate Channel Attributes

See the synopsis info for the list of supported operations. Multiple channels are selected with the **`-i`** or **`-x`** pattern, whichever is more convenient for you. Per default, all channels are processed. It is possible to combine some operations in one call.

5.10 cnttrials - Trial classification

5.10.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cnttrials 3.8   (Linux 2.2.14-5.0 i686)      mvelde, Tue Feb 25 14:54:21 2003
-------------------------------------------------------------------------------
standard:
  cnttrials <source cnt> <average cfg> [<trg in>] [<rej in>] <cls trg out>
```

5.11 trgcount - list all triggers and # occurrences

5.11.1 Synopsis

```
------------------------------------------------------------------------------
ANT Software b.v. (c) 2002 - EEProbe script library
trgcount 1.10 - mvelde - Fri Feb 14 09:10:26 CET 2003
------------------------------------------------------------------------------
trgcount  - program to count #trials per condition (trigger) in trg file

usage:
  trgcount <trg file>
```

5.12 trged - Conditional averaging 1 - trigger recoding

5.12.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
trged 3.10   (Linux 2.2.14-5.0 i686)         mvelde, Tue Feb 25 14:54:12 2003
------------------------------------------------------------------------------
  trged [-d] [-u] [-s [<file>]] -f <cmd_file>|-e <cmd> ... [<trg in>]

options:
  -f <cmd_file>   file(s) containing trigger search/replace command(s)
  -e <cmd>        trigger search/replace command
  -d              put no discontinuity triggers in output
  -u              put no unmatched triggers in output
  -s [<file>]     write replacement statistics to <file>
                  (to standard error if <file> is omitted)
```

Some notes to the command line (this one is not typical for EEP modules):

trged can act as a pipe. If no trigger input file is given, stdin is read. Output is always sent to stdout.

The final script to be executed results from the concatenation of all commands behind a **-e** switch and all commands from the files indicated by a **-f** switch in the given order. You can have multiple **-e** commands and **-f** command files. Don't forget to protect **trged** commands from the shell if you supply them at command-line!

5.12.2 Description

trged is a search-and-replace tool for trigger code sequences in EEP trg files. It's main purpose is the response dependent recoding of the triggers which are generated by the stimulus sequence. The recoded triggers in turn are needed by the **cntaverage** module to distinguish the EEG epochs which corresponds to correct or incorrect answered trials.

The discontinuity triggers "__" and "Rs", which can be generated without your control at any position during recording or evaluation, are ignored in the trigger sequence match.

5.12.3 Script Syntax

This is for reference. The example below is more explanative.

script
: list of *command*s (one per line in case of command files)

command
: *search_sequence whitespaces replace_sequence*

search_sequence
: a list of *regular_expression*s started, separated and closed by a slash "/" character

regular_expression
: a "case insensitive Extended Regular Expression (POSIX.2)" with the restriction that it must not contain slash, space or tab characters (they have a special meaning to **trged** and there is no quoting mechanism in the actual implementation)

whitespaces
: any combination of space and/or tab characters

replace_sequence
: a list of *replace_expression*s started, separated and closed by a slash "/" character, must have the same number of entries like the *search_sequence*

replace_expression

- any literal string with the restriction that it must not start with "+" or "-" and must not contain slash, space or tab characters; each occurence of the ampersand "&" character is replaced by the original trigger code
- a "+" or "-" character followed by a integer number; this value is added to the original trigger to produce the new trigger

There are some restrictions concerning the script size:
max. 32 commands
max. 80 characters per command
max. 16 entries in each search/replace sequence

5.12.4 Example

Assume you have trial codes "12", "13", "14" followed by a response code "43". This is completely wrong since THE ANSWER is "42" (praise "Deep Thought" for finding it). So you want to change these codes to "22", "23", "24". This could be done by the following **trged** script:

```
/12/43/  /22/43/
/13/43/  /23/43/
/14/43/  /24/43/
```

Since the search sequence consists of regular expressions and the replace sequence supports special tags for "add a constant" and "keep unchanged", you can do the same thing with the completely equivalent command:

```
/1[234]/43/  /+10/&/
```

The long first form is self-documenting and easily understood by everyone, whereas the second one looks like a random pile of characters and is cool... Make your own choice.

Using the second form, you could write a shellscript line

```
cntevents -p foo.cnt | trged -e "/1[234]/43/  /+10/&/" > foo.trg
```

which extracts the archived triggers from the cnt file, passes them through the editing script and finally writes the recoded triggers to a file.

5.13 trgisi - list inter-stimulus interval timing information

5.13.1 Synopsis

```
--------------------------------------------------------------------------------
ANT Software b.v. (c) 2002 - EEProbe script library
trgisi 1.9 - mvelde - Fri Feb 21 09:13:23 CET 2003
--------------------------------------------------------------------------------
trgisi 1.9 - program to extract ISI times from a trigger file

usage:
  trgisi [--msec] <trg file> <trigger1> <trigger2> [<mode>] [<maxisi>]

  <trg file>        :   trigger file
  <trigger1>        :   target trigger (for 'response stimulus')
  <trigger2>        :   response trigger code

  optional arguments:
  specify '--msec' as first argument to obtain output in milli-seconds
    (default is seconds)
  <mode> is one of '--values' '--mean' '--min' '--max' '--stats'
    if no mode specified, output will contain all single isi values
  <maxisi> is the maximum ISI time > 0 (sec) for the reaction trigger to occur
    ISI times longer or equal to maxisi will not be taken into account
    (the default value for maxisi is 2 sec)
    (mode --values, output will show this default value of 2 (sec):
                1. for too long ISI
                2. in case of an unmatched pair trigger1/trigger2)
```

5.14 trgvalid - Conditional averaging 2 - trigger/response time validation

5.14.1 Synopsis

```
------------------------------------------------------------------------------
ANT Software b.v. (c) 2001 - EEProbe script library
trgvalid 1.5 - mvelde - 2003/02/10 15:50:04
------------------------------------------------------------------------------
  trgvalid <trigger1> <trigger2> <tmin> <tmax> [-r <mod1> <mod2>] <trg> <trg>
...

arguments:
  <trigger1>, <trigger2>  first, second code of trigger pair
  <tmin>, <tmax>          valid interval for trigger pairs (msec >= 0)
  <trg>                   trigger file(s) to process, output overwrites
                          (original file is saved as <trg>~)
options:
  -r <mod1> <mod2>        replace: trigger pair is modified to mod1, mod2
                          default mod1, mod2:
                            trigger number#  ->  255 - <trigger#>
                            trigger string   ->  <trigger>_mod

tips:
  use '\&' for mod1, mod2 to substitute the original trigger
  use e.g. '+10' for mod1, mod2 to add 10 to the original trigger

examples:

  trgvalid 12 43 200 1000 foo.trg
  -> search for trigger 12 followed by 43 within [200 .. 1000] msec,
     replace with 243 and 212 respectively.

  trgvalid 1[234] 43 200 1000 -r +10 \& foo.trg
  -> search for trigger codes 12, 13, 14 followed by 43,
     within the interval [200 .. 1000] msec, and replace
     trigger codes with 22, 23, 24 respectively, keep 43 unchanged.
     (regex mode is enabled when using '[]' and/or '^', and/or '$';
      regex mode only supported in trigger1/2 arguments)
```

5.15 trgselect - Conditional averaging 3 - randomly select a specified #trials

5.15.1 Synopsis

```
-------------------------------------------------------------------------------
ANT Software b.v. (c) 2002 - EEProbe script library
trgselect 1.9 - mvelde - Fri Feb 14 13:20:58 CET 2003
-------------------------------------------------------------------------------
trgselect  - program to randomly select a specified #trials based on trg file

usage:
  trgselect <trg file> <trigger code> <n select> [<trial start> <trial end>
<rej file>]

  <trg file>        :   trigger file
  <trigger code>    :   as found in the trg file
  <n select>        :   randomly select n of the triggers for processing
                        (name of <trg file> is seed for random process)

  optional arguments (required when taking into account rejections)
  <trial start>     :   start time (msec) of trial, e.g. -200
  <trial end>       :   end time (msec) of trial, e.g. 800
  <rej file>        :   file containing rejection intervals
  (assumes the use of this rej file during further processing)

  (trgselect writes to standard output:
   this is basically a copy of the trg file, changing the de-selected triggers
to '_<code>_')
  NOTE: no recoding is performed on rejected trials:
  - for trials inside a rejected interval (from rej file)
  - for trials classified as rejected (in the eog correction step)
  Further processing should use the same rejection file, and/or eog
compensation mode.
```

6 SIGNAL PROCESSING

Signal processing programs

6.1 avraverage - ERP Grand Averaging

6.1.1 Synopsis

```
--------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
--------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avraverage 3.22   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:15 2003
--------------------------------------------------------------------------------
  avraverage <cfg> <output dir> [<output filename base>]
```

6.1.2 Configuration File

```
; Example configuration file for avraverage


; the input .avr filenames are built from the template
; by expandig "&t" with the condition code and "&s" with
; the subject code below
[filename template]
./&s/a1rf/&s&t.avr

; subject codes to build the .avr filenames ("&s")
[subjects]
pi01
pi12

; condition codes to build the .avr filenames ("&t")
; a grand average is computed for each condition here
; from all subject listed above
[conditions]
1
2

[weighted]
no    ; each average is weighted with 1.0
;sn   ; to weight each input average with sqrt(Ntrials)
;cum  ; accumulate the single trials from the input averages

; list of channels for output averages
; you can comment out this section to request the largest common
; subset of channels available

;[channels]
;F9, F7, F5, F3, FZ, F4, F6, F8, F10
;EOGV, EOGH

; output average time range
; you can comment out this section to request the largest common
; time range available
; (values have to be separated by exact two dots, no whitespaces)
```

```
;[time range]
;-200..1200        ; ms


; For latency jitter compensation: An avrretrieve-like ASCII table
; is read (here: foo.tab);  the [time range] window above is
; interpreted relative to the latency in the given column (here: tgmin)
; then, not relative to 0 in the .avr file.
;
; The correct line is found by matching subj, cond, chan of each input
; avr vector with the "subj", "cond", "chan" columns of the ASCII table,
; whichever are present.

;[time base]
;foo.tab  tgmin
```

6.1.3 Description

Performing a study using the EEP software results in one or more four-dimensional ERP datasets (subject * condition * channel * time). Each dataset consists of several two-dimensional matrix files (.avr files - containing a channel * time matrix). These files are stored in a directory tree which naming conventions encode the (subject * condition) dimensions of the study.

avraverage gives you access to the full dataset, allows selection in all 4 dimensions and can derive *grand averages* from the selected input averages.

The input filelist generation is designed for computing one grand average per listed condition, each one from all listed subjects. This complements the filename rules introduced by the **cntaverage** module. See the more flexible **avrprocess** module if these rules do not match your needs.

Former versions of **avraverage** had options for generating "SAS output". While this functionality is still present, it is obsolete now and replaced by **avrretrieve**.

6.1.4 File Naming

The output filenames are generated from the `[conditions]` codes and the command line arguments as follows:

```
avraverage grand_av.cfg test run
```

would produce the result files:

```
./test/run.n            numbers of trials in the input files
./test/runc1.avr        one grand average for
./test/runc2.avr        each listed condition
...
```

6.2 avrdetrend - Remove linear trends from ERP's

6.2.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrdetrend 3.6   (Linux 2.2.14-5.0 i686)     mvelde, Tue Feb 25 14:54:19 2003
-------------------------------------------------------------------------------
  avrdetrend [-l] [-t] <cfg> <avr in> <avr in> ...
  avrdetrend -s [-l] [-t] <cfg> <avr in> [<avr out>]

options:
  -s       single file mode (output name possible)
  -l       write estimated trend parameters (one .lin file per .avr)
  -t       test mode, output file will contain the trends

files:
  foo.avr  ->  food.avr
               food.lin
```

6.2.2 Configuration File

```
;------------------------------------------------------------------------------
;Example configuration file for avrdetrend
;------------------------------------------------------------------------------

[estimation method]
robust                  ; robust or leastsquares

[exclude ranges]        ; to exclude parts of the ERP from trend estimations
200..400                ; (msec)
;600..700

[correction threshold]  ; min. abs. of slope for correction (units per sec)
0.1                     ; i.e. 'a1' in  y = a0 + a1 * t

[max linearity error]
30.0                    ; average abs. deviation / RMS

[baseline]              ; the baseline is gone after detrending
-200..0                 ; this section is needed if you want to restore a
                        ; real baseline window (msec)

;[channels]              ; to restrict the detrending to a channel subset
;cpz, cz, fcz,
;fz, pz
```

6.3 avrdiff - Calculate ERP differences

6.3.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrdiff 3.6   (Linux 2.2.14-5.0 i686)        mvelde, Tue Feb 25 14:54:19 2003
-------------------------------------------------------------------------------
  avrdiff <avr in #1> <avr in #2> <avr out #1 - #2> [-l <label>][-c <color>]

options:
  -l <label>  conditon label for avr out
  -c <color>  condition color for avr out
```

6.4 avrfilter - ERP Finite Impulse Response filter

6.4.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrfilter 3.9   (Linux 2.2.14-5.0 i686)      mvelde, Tue Feb 25 14:54:20 2003
------------------------------------------------------------------------------
  avrfilter <fir in> <avr in> <avr in> ...
  avrfilter -s <fir in> <avr in> [<avr out>]

options:
  -s       single file mode (output name possible)

files:
  foo.avr  ->  foof.avr
```

6.5 avrinterpol - Spatial ERP Interpolation

6.5.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrinterpol 3.7   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:16 2003
------------------------------------------------------------------------------
  avrinterpol [-t] <avr> <dat> -x <excl regex> -i <interp regex> [<avr out>]
  avrinterpol [-t] <avr> <dat> -x <excl regex> <interp dat>      [<avr out>]

options:
  -x <excl regex>   to exclude channels from interpolation database
  -i <interp regex> to select the channels which are to be interpolated
  -t                test mode, list channel usage only

files:
  foo.avr  ->  fooi.avr
```

avrinterpol uses case insensitive extended regular expressions(POSIX.2) for channel selections. See the **grep** manpage for details. Use the **-t** option to check which channel is really used for what.

For example, the command

```
avrinterpol foo.avr foo.dat -x "eog|^x" -i "^F7$"
```

would produce a file "fooi.avr" where the channel "F7" is interpolated from all channels of foo.avr except of the channels which contain the substring "eog" or which start with the letter "x".

6.5.2 Description

avrinterpol computes "virtual" ERP signal data at requested positions at the unit sphere surface. Basically, a model of the amplitude topography at the unit sphere surface is generated from "good channel" signal data in an ".avr" file and the corresponding sensor positions in a ".dat" file. Using this model a interpolated value at any point of the sphere surface can be computed.

avrinterpol supports two modes corresponding to the main purposes:

Repair Single Channels

Defect ERP channels (drift, sensor failure ...) are replaced using interpolation from good channels. This way you can avoid to remove this channel or subject completetly from your study. The "repair mode" is selected by the presence of the **-i** switch in the command line.

To be used for interpolation input a channel must fullfill the condition:

- channel data are present in the input .avr file and
- channel position is present in the .dat file and
- label doesn't match the exclude pattern (**-x**) and

- label doesn't match the interpolate pattern (**-i**)

The generated output .avr file consists of copied and interpolated channels. A input channel is copied to the output .avr file if:

- channel is present in the input .avr file and
- label doesn't match the interpolate pattern

A channel is replaced by its interpolated version in the output .avr file if:

- channel is present in the input .avr file and
- channel has a valid position in the .dat file and
- label matches the interpolate pattern

Complete Redesign

Sometimes one needs to compare ERP data obtained under different acquisition setups. **`avrinterpol`** allows the transformation from the recorded ERP (described by an .avr and the corresponding .dat file) to a new virtual .avr file which contains channels according to the positions in another .dat file. The "redesign mode" is selected by the absence of the **-i** switch in the command line.

To be used for interpolation input a channel must fullfill the condition:

- channel data are present in the input .avr file and
- channel position is present in the input .dat file and
- label doesn't match the exclude pattern (**-x**)

The generated output .avr file consists of copied and interpolated channels. A input channel is copied to the output .avr file if:

- channel is present in the input .avr file and
- label does match the exclude pattern

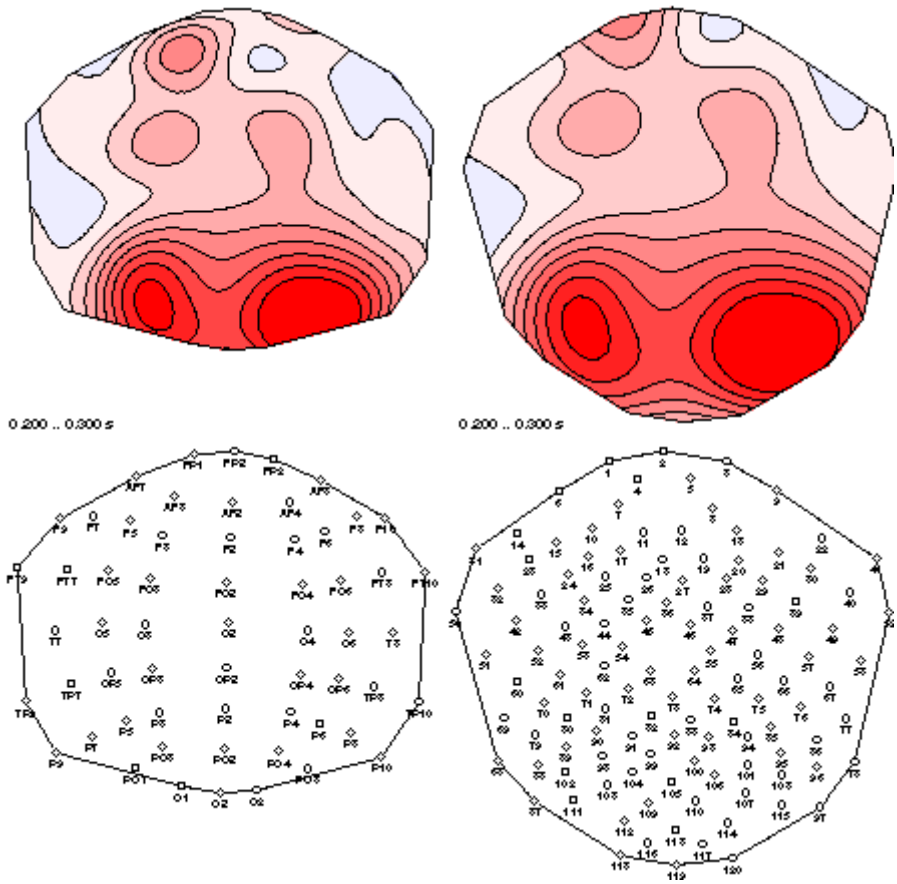
A interpolated channel is created in the .avr file if:

- channel has a valid position in the .dat file and
- label does not match the exclude pattern

6.5.3 Example("redesign" mode)

Below you see a ERP, as measured in a 64 channel setup and the same dataset "redesigned" to a 128 channel setup. Both versions are used to produce a **`xavr`** contour plot in a high resolution.

As expected, both versions produce the same topography - except that the 64 and 128 channel designs cover slightly different parts of the head surface. No noticeable data loss or falsification is caused by the **`avrinterpol`** run.



Surface field distributions, based on 64 real sensors(left) and 128 interpolated sensors

6.6 avrprocess - Basic ERP Operations

6.6.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrprocess 3.10   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:16 2003
-------------------------------------------------------------------------------
  avrprocess -o <avr out> [-l <label>] [-c <color>]
             <process options> [<avr in> ... ]

common options:
  -o <avr out> output filename/output directory
  -l <label>   conditon label for avr out
  -c <color>   condition color for avr out
  -t <ms>      add <ms> to trigger offset for avr out

process options:
  --set        set header information
  --diff [-v]  calc average difference
               -v   conserve trial numbers and variances
  --cum        accumulate single trials
  --avr  [-w]  calc grand average
               -w   S/N ratio weighting
```

6.6.2 Description

avrprocess is a general command line interface to several processing options. It's main purpose is to use the shell capabilities to build a list of input files. This way is faster and more flexible than having a special configuration with a proprietary list generation for each job.

Each run of **avrprocess** generates exact one average output file. You have to specify a name for it and you can specify a new label and a new color (per default, this information is copied from the first input file).

The output average always contains the largest common channel subset and the largest common time range of all input averages.

6.6.3 Processes

--diff [-v]

calc average difference(first file - second file)
exact two input files required
normally, the output avr will contain no variance info; the number of trials is set to 1, the number of rejected trials is set to 0
with the -v option, the variances are copied from the first operand and the trial numbers are unchanged - use with care, this is normally an **invalid** operation

--set

set header information
exact one input file required
no data changes; output contains new label and/or color only

--cum

accumulate single trials
output contains the average of the single trials of the input averages(mainly used to merge blockwise calculated averages into one full-session average)

--avr [-w]

calc grand average
output contains the mean and variance of the input averages; optionally, you can weight the input files with $sqrt(N_{Trials})$

6.6.4 Examples (bash required)

```
avrprocess -o vp01r.avr -l rare_all -c green --cum vp01r{1,2,3}.avr
```
accumulates the single trials of the files

```
vp01r1.avr
vp01r2.avr
vp01r3.avr
```
- assume they contain blockwise ERP's of a condition "rare" - into the output average

```
vp01r.avr
```
which then contains the ERP of the rare condition for the whole session

6.7 avrreref - ERP Rereferencing

6.7.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrreref 3.10   (Linux 2.2.14-5.0 i686)      mvelde, Tue Feb 25 14:54:15 2003
-------------------------------------------------------------------------------
  avrreref <cfg> <avr in> <avr in> ...
  avrreref -s <cfg> <avr in> [<avr out>]

options:
  -s       single file mode (output name possible)

files:
  foo.avr  ->  foor.avr
```

6.7.2 Configuration File

```
; example configuration file for avrreref/cntreref

; rule for virtual reference calculation
[reference calculation]
CZ             ; the CZ channel is the new reference

;CZ, FZ, PZ    ; the mean of the listed channels is the new reference
(CZ+FZ+PZ)/3

;CZ * 0.5      ; CZ * 0.5 is the new reference

;all, -CZ      ; the mean of all channels, except of CZ, is the new reference


; list of channels for rereferencing
[reference subtraction]
all, -eogv, -eogh   ; the virtual reference is subtracted from all channels,
                    ; except of EOGV, EOGH

;F1, F2, F3         ; the virtual reference is subtracted from the listed
                    ; channels only

; create a new channel in the output file which stores the virtual reference
;[reference insertion]
;REF                ; label of the new channel
```

6.8 avrretrieve - ASCII Export of ERP Dataset Information

6.8.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrretrieve 3.5   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:17 2003
-------------------------------------------------------------------------------
  avrretrieve <cfg> <output file> [options]

options:
  -P [<dir>]   scan directory <dir> for present subjects/conditions
               (do not use the files listed in the <cfg>)
  -F <regex>   use only <regex> matching filenames in the directory scan
  -S <length>  subject code length for directory scan
  --cfg        write the directory scan results as configuration file

  --mean       retrieve window averages
  --min        retrieve (smallest local) window minima
  --max        retrieve (largest local) window maxima
  --gmin       like --min but use global min. if no local one is found
  --gmax       like --max but use global max. if no local one is found

  --bsl        retrieve mean baseline values
  --count      retrieve accepted/total trial count
```

6.8.2 Configuration File

```
; Example configuration file for avrretrieve

; the input .avr filenames are built from the template
; by expanding "&t" with the condition code and "&s" with
; the subject code below

[filename template]
./&s/a1rf/&s&t.avr

; condition codes to build the .avr filenames ("&t")
[conditions]
1
2

; subject codes to build the .avr filenames ("&s")
[subjects]
pi01
pi02
pi03

; channel subset selection
; comment out this section to select the largest common channel subset
[channels]
F5, FZ, F6, EOGV, EOGH
```

```
; window
[windows]
100..300..500..700
;100..300, 300..500, 500..700 ; both notations are equivalent
```

6.8.3 Description

Performing a study using the EEP software results in one or more four-dimensional ERP datasets (subject * condition * channel * time). Each dataset consists of several two-dimensional matrix files(*.avr files) containing a (channel * time) matrix. These files are stored in a directory tree which naming conventions encode the (subject * condition) dimensions of the study.

avrretrieve gives you access to such datasets, allows selection in all dimensions and, as the name suggests, allows to retrieve selected information from the dataset. The retrieved information is written in plain ASCII to allow the import in whatever application you prefer for your subsequent processing steps.

6.8.4 Retrievable Information

--mean
: window average for each subject/condition/channel/window

--min/--max
: window minimum/maximum and its latency for each subject/condition/channel/window; strictly spoken, the "largest/smallest local maximum/minimum within the window" is retrieved - it is possible that no such value exist (in monotonous signals) and an error message occurs

--gmin/--gmax
: global window minimum/maximum and its latency for each subject/condition/channel/window

--bsl
: mean baseline compensation value for each subject/condition/channel; requires the .bsl files from the **cntaverage** run

--count
: accepted/total number of trials for each subject/condition

--cfg
: produce the [subjects] and [conditions] sections of a configuration file according to the results of a directory scan; requires the -P option and friends

The options --mean, --max, --min, --gmax, --gmin might be combined. Each option enables additional column(s) in the output table.

The options --bsl, --count and --cfg must be used exclusive.

6.8.5 Examples

The output file foo.tab, produced by

```
avrretrieve foo.cfg foo.tab --mean --min
```

could look like

```
subj cond chan  win          mean         ymin         tmin
pi01 Pp   F5    -0148..-0100 -6.24072e-01 -8.18361e-01 -0.123
pi01 Pp   F5    -0100..+0100  1.77407e+00 -9.34474e-01  0.050
pi01 Pp   FZ    -0148..-0100 -9.32307e-02 -1.48055e+00 -0.140
pi01 Pp   FZ    -0100..+0100  9.77079e-01  2.24657e+00  0.068
...
```

The output file foo.tab, produced by

```
avrretrieve foo.cfg foo.tab --count
```

could look like

```
subj cond total accept
pi01 Pp     90   42
pi01 Pw    120   66
pi02 Pp     90   33
pi02 Pw    120    5
...
```

6.9 avrstats - create an ERP t-test avr

6.9.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
avrstats 1.6   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:24 2003
------------------------------------------------------------------------------
  avrstats <avr in #1> <avr in #2> <tst out> [-l <label>][-c <color>]

options:
     <tst out> running t-test statistic filename in avr format
               (when omitted, output to stdout)
  -l <label>  identifier label for tst out
  -c <color>  identifier color for tst out
```

6.9.2 Description

This command can be used to create an 't.avr' file that contains the result of the running t-test. This calculation is performed exactly identical as in the xavr viewer. The output 't.avr' allows you to view the result also in mapping view to display a t-map or significance-map.

6.10 cntaverage - Single Subject ERP Calculation

6.10.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntaverage 3.28   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:14 2003
-------------------------------------------------------------------------------
standard:
  cntaverage <source cnt> <cfg> [<trg in>] [<rej in>]
             [-p <output path>/[<new code>]] [-c]

eog compensation:
  cntaverage -e [-a] <source cnt> <cfg> [<cls trg in>] [<pfc in>]
             [-p <output path>/[<new code>]] [-c]

options:
  -p ...     output file name
             all text before the last "/" is interpreted as a path
             which will be created if needed
             the rest, if exists, is the new subject code
  -c         calc covariance matrices
  -e         enable eog compensation mode
  -a         apply the eog compensation to eog channels
```

6.10.2 Configuration File

```
;--------------------------------------------------------------------------
;Example configuration file for cntaverage
;--------------------------------------------------------------------------

Conditions:             ; list of conditions to average

10                      ; condition code (read from .trg file)
Cond1 (GREEN)           ; condition label / color

20,2A=A                 ; multiple trigger codes may be summed up into
Group_1 (SKY)           ; one average
                        ; '=..' assigns the new condition code

                ; "condition code" (first line) should be a 2 character
                ; code, it appears in the output .avr filename
                ; "condition label" (second line) is a max. 10 char.
                ; identifier which is stored in the file, it appears
                ; in ERP plot legends


Channels:               ; list of channel labels
FP1                     ; (may be a subset of all channels available)
FP2                     ; if no "Channels" section is spezified, all channels
F7                      ; are averaged
F3
```

```
FZ
EOGV
EOGH

Window:                 ; averaging window in msec - relative to trigger
-200..1500

Baseline:               ; definition of baseline
-300..0                 ; (may extend beyond or be completely outside of
                        ; averaging window, if required..)
                        ; you can turn off the baseline calculation by
                        ; writing "ABS" instead of the window borders

; the baseline area reference trigger can be distinct from the
; average window reference

;Reference displacement: -1         ; you can specify a trigger index offset

;Scanning for triggers:  9 9 15 15  ; or a trigger value for each condition
                                    ; (the reference displacement value
indicates
                                    ; the direction for baseline trigger
search)

Rejection:              ; range where rejection marks are being checked for
-200..1500              ; (rejection may temporarily be disabled by placing
                        ; the keyword 'OFF' behind 'Rejection:')
```

The allowed condition colors are:

BLUE, GREEN, CYAN, RED, MAGENTA, YELLOW, WHITE, BLACK, STEEL, SKY, MINT, SEA, LEAVES, OLIVE, SIENNA, LIGHTGREEN, OCHRE, APRICOT, ORANGE, CRIMSON, ROSE, PINK, PURPLE, LILAC, AUBERGINE, PLUM, UV

6.10.3 Description

Input Trial Selection/Processing

cntaverage computes means/variances of the corresponding sample points in multiple epochs (*trials*) of one raw data file.

It uses its configuration file and the wealth of signal/trial classifcation lists which you got from the preceding evaluation steps (**cntreject**, **xcnt**, **trged**, **xeog**).

In detail, the following happens for each condition which is defined in the .cfg file:

Standard:

1. All triggers which are to be averaged for this condition are located in .trg. The absolute trial epochs are computed from the (absolute) trigger latency and the (relative) window size definitions in .cfg.

2. If a .rej file is present and a rejection checking window is defined in the .cfg, these epochs are checked for overlapping with any of the rejected signal epochs. Trials with such overlaps are classified as "rejected".

3. All trial epochs (the smallest epoch which covers the averaging window and the baseline window, if any, and the rejection window, if any) are checked for

containing discontinuity ("__") or DC-reset ("Rs") triggers within their limits. Such trials are "rejected" too.

4. The nonrejected trials are loaded. If requested, a baseline compensation value is computed for each trial/channel as the mean of the signal values within the baseline window. This value is subtracted from the signal within the averaging window.

5. The means/variances of the corresponding sample points of the averaging window of all nonrejected trials are computed. That's it. The means are believed to represent the *Event Related Potential* for one subject in one condition.

6. The computed means, variances, trial numbers and some additional information are stored as .avr file for subsequent evaluations.

The EOG compensation mode differs in the trial classification mechanism and signal data processing:

1. All triggers which are to be averaged for this condition are located in .trg. The absolute trial epochs are computed from the (absolute) trigger latency and the (relative) window size definitions in .cfg.

2. A classification attribute for each "trial triggering trigger" is expected and read from .trg. (The trigger file with this attribute comes from **xeog**'s "Save Trial Classification" function). It is an error if no classification can be found for one of the trials.

3. Trials with a "m" or "b" classification are passed through the EOG compensation step. See the algorithm for details.

4. Further processing is done as in the standard mode starting with step 4.). The "u" trials and the corrected "m" and "b" trials are nonrejected now, the "r" trials are rejected.

Output File Naming

The output files are named according to the following examples (see also the recommended EEP project tree naming conventions). Assume the following files are present

```
./foo.cnt
./foo.trg
./foo.rej
../cfg/average.cfg
```

and you have the following conditions defined in ../cfg/average.cfg

```
Conditions:

da,di,dum=d1
Daddel (GREEN)

li,la,lei=l1
Laber (RED)
```

Then, the command
cntaverage foo.cnt ../cfg/average.cfg -p test/
would produce the files

```
./test/foo.bsl     - baseline value listing
./test/food1.avr   - ERP for subject foo in condition
"Daddel"(d1)
./test/fool1.avr   - ERP for subject foo in condition
"Laber"(l1)
```

Or, the command
cntaverage foo.cnt ../cfg/average.cfg -p test/pa53
would produce the files

```
./test/pa53.bsl    - baseline value listing
./test/pa53d1.avr  - ERP for subject pa53, formerly known as
"foo",
                     in condition "Daddel"(d1)
./test/pa53l1.avr  - ERP for subject pa53, formerly known as
"foo",
                     in condition "Laber"(l1)
```

6.11 cntdetrend - Remove offsets and linear trends

6.11.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntdetrend 3.15   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:14 2003
------------------------------------------------------------------------------
  cntdetrend <source cnt> [<dest cnt>] <cfg> [<trg in>] [<rej in>]
files:
  foo.cnt      ->  food.cnt
  foo.trg          food.trg
  foo.rej          food.lin
```

6.11.2 Configuration File

```
;------------------------------------------------------------------------------
;Example configuration file for cntdetrend
;------------------------------------------------------------------------------

Estimation window size:
 10000                  ; in msec  (requested)
                        ; -> actual sizes will be determined by DC-resets
                        ;    and position of synchronizing triggers
Synchronize on:
 10 (-500),             ; trigger numbers (.TRG-file required)
 11 (-200), 12 (-300)

Exclude range(s) (ms):
 1000..1700, 3500..4000 ; in msec  (max. 30 separate ranges)
relative to:            ; -> ranges refer to specified triggers,
 12, 20, 22             ;    not necessarily to beginning of epochs

Estimation method:
;Robust                 ; minimizes absolute deviations
                        ;   (slower, but normally more precise estimates)
 LeastSquares           ; minimizes squared deviations

Correction threshold:
 0.0                    ; min slope, i.e. 'b' in  y=a+b*t
Max. linearity error:
 30.0                   ; average abs. deviation / RMS
Exception handling:
 DC                     ; correct DC-offset only
;none                   ; neither correct DC nor slope
```

6.11.3 Description

Splitting into Windows

The estimation of a EEG/MEG drift by a linear function is valid for epochs of few seconds only. Therefore it is necessary to split the continuous signal. The resulting discontinuities should not be in the interesting trial epochs.

To manage this, **`cntdetrend`** looks at first for present resets/discontinuities and forces a window limit there. The resulting epochs are temporary split to windows according to the length in the configuration file.

If a *Synchronize* trigger position is found in the range *+/- 0.8 * Estimation Window Size*, this position is taken instead of the requested position. If this new position is nearer then *0.5 * Estimation Window Size* to a previous window limit, then it is ignored (to avoid short windows which are bad for the detrending algorithms), otherwise the new position becomes a real window limit.

This means, that a *Estimation Window Size* of at least 1.25 total stimulus intervals ensures that at least one stimulus trigger is found in the shift range. If all stimulus triggers are configured as *Synchronize* triggers, every window limit (discontinuity artifact) can be shifted out of the trial window.

Trend Estimation

A linear trend is estimated and subtracted independently in each window and each channel. Rejected epochs and *Excluded* ranges are excluded from the trend estimation.

The estimation is possible via simple regression (least error squares) or via a robust estimation (least error absolutes). To reduce the computation effort, the estimation is not done with the original sample points. The signal is virtually downsampled to about 15 Hz, where each new sample point comes from the mean of the original sample points.

It is possible to supply a test criterion for the validity of the linear estimation and to choose an exception handling (see Configuration).

6.12 cntfilter - Finite Impulse Response Filter

6.12.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntfilter 3.25   (Linux 2.2.14-5.0 i686)     mvelde, Tue Feb 25 14:54:12 2003
------------------------------------------------------------------------------
  cntfilter <source cnt> [<dest cnt>] [<trg in>] <fir in> [-i|-x <regex>] [-d]

options:
  -i <regex>    include selected channels for filtering
  -x <regex>    exclude selected channels
  -d            remove DC offsets from filtered channels

files:
  foo.cnt  ->  foof.cnt
  foo.trg
```

6.12.2 Description

`cntfilter` reads a precalculated finite impulse response from a textfile (*.fir) and applies this Finite Impulse Response Filter to the cnt file. Filters can be designed with the **`xfir`** module. Some filters are available in the `fir` directory /opt/eep/fir.

6.12.3 Discontinuity Handling

The filtering window (N_{coeffs} samples) does not "overrun" any signal discontinuities to reduce unwanted edge effects. The record is filtered in independent epochs instead, where record start, record stop, discontinuity("__") and DC reset("Rs") are the epoch limits.

At each epoch limit, the signal is virtually extended by a constant signal of $(N_{coeffs} - 1) / 2$ samples. This means that the edges of a filtered signal epoch are not influenced by the signal values of the previous/next epoch. Of course, the edges **are** influenced by the artificial constant signal.

In the case of record start, record stop and discontinuity ("__" trigger), the first, respectively the last, sample value is used to extend the signal. The signal values near DC reset positions are normally trash and a bad choice to estimate the past/future of the recorded signal. Therefore, the signal value of *reset_time -/+ 15 ms* is used to extend the signal. There are options `-l <ms>` and `-r <ms>` to override the 15 ms built in.

6.13 cntreject - Mark disturbed signal epochs

6.13.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntreject 4.9   (Linux 2.2.14-5.0 i686)      mvelde, Tue Feb 25 14:54:13 2003
-------------------------------------------------------------------------------
  cntreject <source cnt> <cfg> [<trg in>] [<rej out>] [-a] [-+ [<rej in>]]

options:
  -a          choose stddev parameters automatically
  -+ [<file>] add-mode, merge the new rejections with the input file

files:
  foo.cnt     ->  foo.rej
  foo.trg
  foo.rej
```

6.13.2 Configuration File

Standard Deviation Mode

```
;--------------------------------------------------------------------------
;Example configuration file for standard deviation REJECTion
;--------------------------------------------------------------------------

StdDev mode 200ms       ; selects StdDev mode and width of sliding window
                        ; -> see "rejectt.cfg" for other mode
Histogram scaling 0.1   ; specifies range for which histogram is computed
                        ; (may be omitted; default is 0.1 = 10% of full scale)
Cutoff point 15         ; the modal value of the distribution is multiplied by
                        ; this factor to yield the AUTO MODE threshold
                        ; (may be omitted; default is 15)

EOGV                    ; channels to test for rejection...
   33.75                ; threshold SD (uV); if this threshold is surpassed,
                        ; the corresponding sample will be marked 'rejected'
EOGH
   33.75
FZ
   20.00
```

Threshold Mode

```
;--------------------------------------------------------------------------
;Example configuration file for Threshold REJECTion
;--------------------------------------------------------------------------

Threshold mode          ; selects Threshold mode

EOGV                    ; channels to test for rejection...
   -30                  ; low threshold (uV)
   33.75                ; high threshold (uV) if surpassed,
                        ; the corresponding samples will be marked 'rejected'
EOGH
   -40
   33.75
```

6.13.3 Description(Standard Deviation Mode)

The standard deviation is calculated for a moving window at each position. This position is marked "rejected" whenever the limit is exceeded for one of the watched channels.

At discontinuities, the signal is virtually extended with constant values before the standard deviation is calculated. This means that a rapid signal change, resulting from epoching or concatenation, leads not to a rejection mark around it.

6.14 cntreject_t - Mark disturbed trials

6.14.1 Synopsis

```
------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntreject_t 3.8   (Linux 2.2.14-5.0 i686)    mvelde, Tue Feb 25 14:54:13 2003
------------------------------------------------------------------------------
  cntreject_t <cnt> <cfg> [<trg>] [<rej out>] -a <average cfg> [-+ [<rej in>]]

options:
  -+ [<rej in>]    add-mode, merge the new rejections with the input file
```

6.14.2 Configuration File

```
; example configuration for cntreject_t

[trial rejection thresholds]
EOGV -30 30        ; two values - min/max
EOGH  60           ; one value  - max delta
                   ; all values in cnt channel units (ft or uV)
```

6.14.3 Description

cntreject_t performs a "trialwise" outlier check. It loads EEG/MEG trial epochs exactly as the **cntaverage** module would do. For each trial rejection window, the channels listed in the configuration are checked and exceptions are marked as "rejected". In detail:

- min/max (channel label followed by 2 threshold values)
  all samples with values below min or above max are marked as rejected
- delta (channel label followed by 1 threshold value)
  min and max of the signal are located, if max-min exceeds the threshold, the epoch between min/max is marked as rejected

6.15 cntreref - Raw Data Rereferencing

6.15.1 Synopsis

```
-------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
-------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
cntreref 3.6   (Linux 2.2.14-5.0 i686)       mvelde, Tue Feb 25 14:54:11 2003
-------------------------------------------------------------------------------
  cntreref <source cnt> <cfg> [<dest cnt>]

files:
  foo.cnt  ->  foor.cnt
```

6.15.2 Configuration File

```
; example configuration file for avrreref/cntreref

; rule for virtual reference calculation
[reference calculation]
CZ              ; the CZ channel is the new reference

;CZ, FZ, PZ     ; the mean of the listed channels is the new reference
(CZ+FZ+PZ)/3

;CZ * 0.5       ; CZ * 0.5 is the new reference

;all, -CZ       ; the mean of all channels, except of CZ, is the new reference


; list of channels for rereferencing
[reference subtraction]
all, -eogv, -eogh   ; the virtual reference is subtracted from all channels,
                    ; except of EOGV, EOGH

;F1, F2, F3         ; the virtual reference is subtracted from the listed
                    ; channels only

; create a new channel in the output file which stores the virtual reference
;[reference insertion]
;REF                ; label of the new channel
```

6.15.3 Description

cntreref computes a virtual reference channel from other channels and subtracts this signal. Note that you could perform the rereferencing also "on the fly" in **xcnt** or later with the ERP's (**xavr**, **avrreref**). Thus, **cntreref** is mostly used to repair/unify raw data records with different reference settings, not for playing around with the reference effects.

6.16 xfir - Finite Impulse Response Filter Design

6.16.1 Contents

6.16.2 Synopsis

```
--------------------------------------------------------------------------------
EEProbe 3.1-9   ANT Software - Enschede  (www.ant-software.nl)  (c) 2000-2002
--------------------------------------------------------------------------------
EEP 3.1  Max-Planck-Institute of Cognitive Neuroscience 1996-2002
xfir 3.10   (Linux 2.2.14-5.0 i686)          mvelde, Tue Feb 25 14:54:09 2003
--------------------------------------------------------------------------------
  xfir [<fir>]
```

6.16.3 Description

xfir is a X/Motif based Finite Impulse Response(FIR) Filter design program. It allows the interactive design of the filter coefficient sets which are needed by **cntfilter**, **xcnt**, **avrfilter** and **xavr** to filter signals.

6.16.4 Usage

You will enter the desired design method and filter characteristics in the **File - Modify** dialog and calculate the actual response using the **Apply** button of this dialog. This must be repeated until an acceptable solution is found. The result can be written to disk as a .fir file using **File - Save**.

The filter length must be an odd number and all the frequencies must have plausible values. The error checking is quite sensitive but the messages are Spartan.

The Remez-Exchange design offers an **Auto** toggle. When set, **xfir** ignores the given length and tries to find the shortest filter which fulfills the given passband ripple and stopband gain requirements. The optimization which is done here is not very sophisticated. It is slow, it fails often to find any solution - especially for "extreme" filters (see below) and you can stop it only by canceling the whole program...

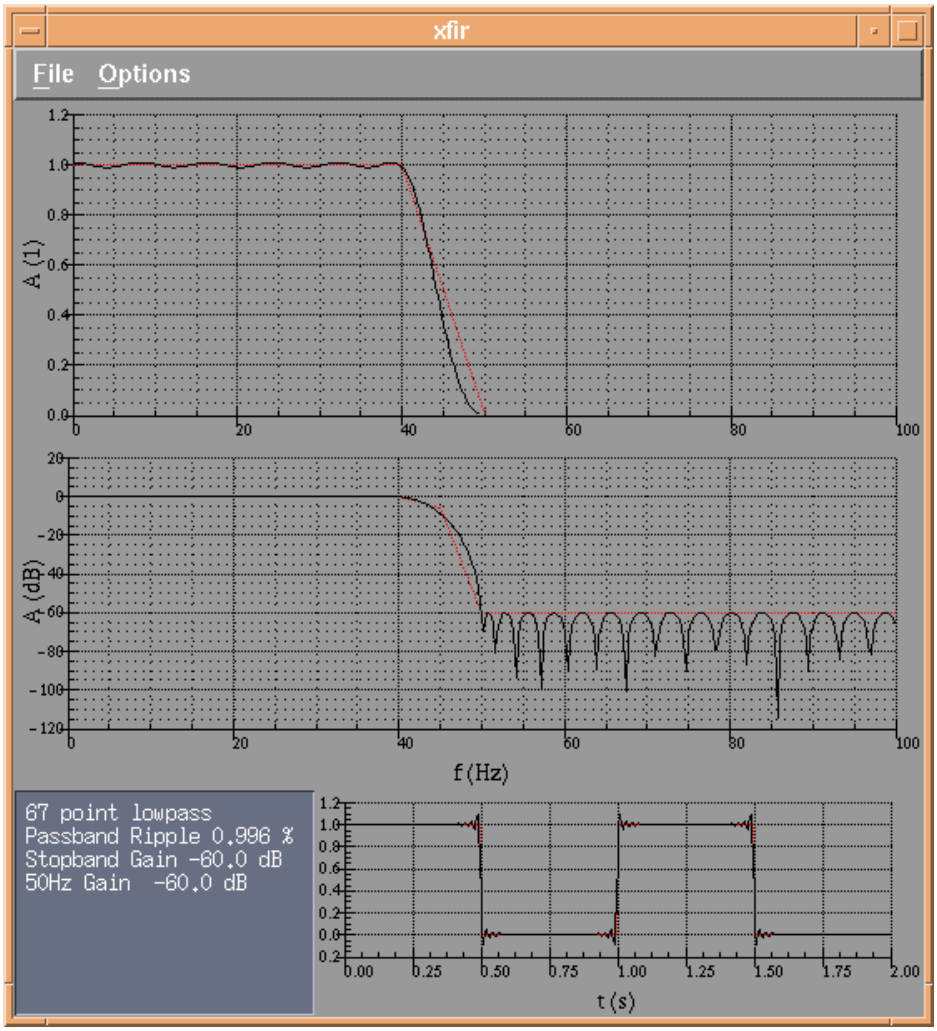
xfir is integrated with the EEP viewer programs. With **Options - Export Now**, the current filter is exported (surprise!) and all instances of **xcnt** and/or **xavr** at the same screen show how the data would look if they were filtered with this coefficient set. With the **Options - Export Always** toggle enabled, this is done automatically for each calculated filter. Note that this can be slow for expensive filters.

6.16.5 The xfir Screen

The upper two diagrams show the actual (black) and the desired(red-dotted) magnitude response, according to the actual design parameters. Fourier Series filters have a desired stopband gain of -infinite dB; this is replaced with -90 dB to allow the display at a finite monitor. The frequency axis limits are configurable via the **Options - Format** dialog.

The desired response is not displayed when the filter coefficients are loaded from a file. The design parameters are not reliably known in this case.

The lower diagram shows a sample 1 Hz rectangular pulse (red-dotted) and it's filtered version (black). The textfield lists some attributes and characteristic points of the actual filter.



***xfir** screenshot*

6.16.6 Practical Design Rules

- Filters in general should have a constant amplification of 1 (0 dB) in the interesting passband and 0 (-inf dB) in the unwanted stopband. For the authors taste, -45 dB in the stopband is acceptable, -60 dB is really good, -90 dB sounds like HiFi equipment quality and is surely overkill...

- FIR filters should be as short as possible and as long as necessary. An approximation for a good length to start with is

  $$n = 2 * f_{sample} / f_{crit}$$

  where f_{crit} is the lowest critical frequency in your design. Such a filter is long enough to cover 2 periods of the lowest interesting frequency and has a chance to handle it...
- The Fourier-Series method is better for "extreme" filters, where f_{crit} is far away from f_{sample} (say, a 0.5 Hz highpass for a 250 Hz sampled signal). The Remex-Exchange method offers more control for "reasonable" filters (say, a 10..30 Hz bandpass for a 250 Hz sampled signal).
- A wider transition band in a Remex-Exchange design can improve the other characteristics - and therefore reduce the required length - dramatically. But a noticeable real passband/stopband must be always present before/between transition bands. In other words, the "desired response" line must have horizontal pieces before and between slopes.

6.16.7 FIR Files

The files created by **`xfir`** are different from the classic EEP 2.0 *.fir files. They store both the filter design parameters and the calculated FIR coefficients. EEP 2.0 had a separate configuration file for the design.

EEP 3.1 modules (xfir, xcnt, xavr, cntfilter, avrfilter) can read the **filter coefficients** from both flavors of *.fir. **`xfir`** can read the **filter design** only from its own files. The EEP 2.0 programs cannot read the new *.fir files. Use your favorite text editor for conversions.

Some example fir files are available.

6.16.8 References

The SciPlot Widget
Copyright (c) 1996 Robert W. McMullen

FIR LINEAR PHASE FILTER DESIGN PROGRAM
James H. McClellan, Thomas W. Parks, Lawrence R. Rabiner

Rorabaugh, Britt.
Digital Filter Designer's Handbook, McGraw-Hill 1993

7 APPENDICES

Files

- File Types
- Configuration Files
- Signal Data Files
- Trigger Files

7.1 EEP - File Types

The following table lists the file types which are understood/produced by the EEP modules and points to the file format description (if any). The different files are distinguished by the filename extensions.

Extension	Description
cnt	continuous data file the huge EEG/MEG raw-data files
trg	trigger file ASCII list of time/code pairs which mark certain points in the cnt
res	ERTS result file lists information about the presented stimuli and the subject responses; produced during the EEG session, used in scripts to produce sufficient EEP trg files
rej	rejection file ASCII list of time epochs which usually mark disturbed epochs in the cnt; produced by the automatic classifiers **cntreject**, **cntreject t** or interactively by **xeog**, **xcnt**
cls	prototype classification for EOG artifact compensation; ASCII list of prototypic blink/move epochs to derive the propagation factors; produced/used by **xeog**
pfc	propagation factors the propagation factors for the EOG artifact compensation; produced by **xeog**, used by **cntaverage**
pk	peak file contains the peak (component) parameters as made during the peak scoring in the xavr viewer. Components are bound to a condition based on the (basename) of the avr file.
avr	average file the ERP data calculated from EEG/MEG; basically one channel/time matrix of means and one matrix of variances per file; compatible with EEP 2.0
cfg	configuration file parameter sets for the different EEP modules; partly compatible with EEP 2.0
lin	linear components produced by **cntdetrend**; lists straight line parameters for each channel/epoch, these lines are subtracted during the **cntdetrend** run
bsl	baseline produced by **cntaverage**; lists the mean/variance of the subtracted single trial baseline values for each channel/condition
cvd cvn	data/noise covariance matrix optionally produced (one per condition) by **cntaverage**; in ASA (ASCII) format
fir	FIR filter coefficient vector ASCII list of the finite impulse response of a desired digital filter; EEP 3.x has a own format but can still read the EEP 2.0 fir's; produced by **xfir**, used by **cntfilter**, **avrfilter**, **xavr**, **xcnt**
dat	3D-space head digitizer file ASCII list of the sensor coordinates in space; used for spatial interpolation in **xavr**'s contour plots and in **avrinterpol**
n tab sas raw asc	ASCII export of EEP data; produced by **avrretrieve** and some of the converters to allow subsequent processing in SAS, Excel, awk, perl...

7.2 Configuration Files

7.2.1 Contents

7.2.2 Introduction

EEP configuration files are text files which contain the parameters for the different evaluation modules. The syntax depends on the module. For historical reasons, it is not very consistent. Nevertheless there are some common rules:

- one-line comments are started with "`;`"
- configuration files are evaluated case-insensitive (except of filenames and other case-sensitive keywords for external communication)
- sample configurations are available
- The newer configurations without compatibility constraints have an uniform style. Examples for the old style are **`cntepoch`**, **`cntreject`**, **`cntdetrend`**, **`cntaverage`**. Examples for the new style are **`xeog`**, **`avrretrieve`**.

7.2.3 General EEP 3.x Configuration Style

The configuration is always split into *sections*, each containing a variable length list of *items*. This preprocessing is done by one standard lexer. Each module has to parse only the items from the sections it is interested in.

Specification

This is for reference. The example below is more explanative.

configuration_file
: sequence of *section*s, *comment*s and empty lines in any order

comment
: everything between a semicolon character and a newline, inclusive

section
: *section_title* newline *item_list*

section_title
: opening brace **[**, *section_key*, closing brace **]**, newline
whitespaces are allowed before, between, behind the elements above

section_key
: any string made of *whitespaces*,letters, digits and underscore

item_list
: list of *item*s, separated by comma, newline or both

item
: any string (can contain **,;[]** characters only if enclosed in double quotes)

whitespaces
any combination of space and/or tab characters

Example

```
; note the different item separations here: comma, newline or both
[Channels]
F1, F2, F3
F4,
F5
F6

; the EEP configuration format supports only one-line comments
; but there is a simple trick to comment out a whole section
;  - just make the section keyword invalid
[rem Channels]
P1, P2, P3,
P4, P5, P6

; you can format your text using whitespaces
[  My     Calculation   Factor          ]
     0.0345       ,

; double quotes allow to have the special characters ",;[]"
; in configuration items
[objects]
text  txt:"Hello, world! [what a text]"
```

7.2.4 "X-Resource" or "Application-Default" Files

As all other X-Toolkit based programs, the graphical EEP modules (**`xcnt`**, **`xavr`**, ...) have an additional powerful configuration mechanism - the resource database (http://www.plig.org/xwinman/resource.html).

This mechanism is not thought for daily use, but it allows personal fine-tuning of many user interface details such as colors, fonts, window sizes... Have a look in the respective `app-defaults` sample file and the "X" manpage to get a basic understanding what can be set and how it is done.

7.3 Signal Data Files

The filename extension "cnt" was introduced by the NeuroScan acquisition software and stands for "**co**n**t**inuous EEG data file".

In EEP, "cnt" is used as the general shortcut for several variants of signal data files. In this context "cnt" means only: a file which somehow allows to retrieve/store

- a signal matrix ($N_{channels}$ * $N_{samples}$ sample points in max. 32 bit signed accuracy)
- unit and amplitude axis scaling factors for each channel
- a unique label for each channel (channel labels in EEP are evaluated case insensitive, must not be longer than 10 characters and must consist of the allowed characters only (letters, digits, '-', '+' and '_'; NO whitespaces, parentheses, punctuations...)
- one common time axis scaling factor for all channels
- optionally, a list of time/code pairs (better known as "triggers" or "events")

Currently, these fileformats are valid input to the cnt... programs:

Name	Extension	Description
NeuroScan (ns)	cnt	the data files produced by our NeuroScan SynAmps amplifier software; read only, output is converted to riff
EEP 2.0 (eep)	cnt	the file format of the DOS EEP 2.0 software (a proprietary, slightly modified version of an older NeuroScan file format)
EEP 3.x (riff)	cnt	the highly recommended proprietary EEP 3.x format; files are compressed to about 1/3 of the size, compared with the other two cnt variants
EEP average	avr	the ERP matrix files used in the EEP software; read only, output is converted to riff

Each of the formats above can contain data which are not needed/maintained by EEP modules. They try to keep these data unchanged. Other things are not 100% portable between the formats.

In summary, it's possible to read each filetype listed above with the cnt... programs, but you should use the compressed format whenever possible and you cannot rely on any little detail in any format.

7.4 Trigger Files

7.4.1 Introduction

EEP uses trigger files - simple ASCII lists of time/code pairs - to mark certain time points in the continuous EEG records. These are used by different EEP modules to navigate on the time axis, especially to find and classify the interesting trial epochs.

This concept allows to control the evaluation by editing the table. Since there is a lot of great software for all kind of text manipulation available, we have infinite flexibility and power here at no cost.

7.4.2 Specification

The trigger ASCII list consists of one header line followed by one line for each trigger

```
<interval>  <blocksize>
<time_1>    <offset_1>   <code_1>
<time_2>    <offset_2>   <code_2>
...
<time_n>    <offset_n>   <code_n>
```

where the components are:

interval
: Sampling interval (in seconds). This is normally compared with the interval stored in the cnt file to detect mismatches.

blocksize
: Size (in bytes) of the data block for one sample in the EEP 2.0 fileformat ($N_{channels}$ * 2 + 4).
This value is unused in EEP 3.x programs but the parser expects to find a number here.

time
: Trigger position (in seconds).

offset
: File offset (in bytes) in the EEP 2.0 fileformat which corresponds to the trigger position
(900 + (*blocksize* - 4)/2 * 75 + *time*/*interval* * *blocksize*).
This value is unused in EEP 3.x programs but the parser expects to find a number here.

code
: String of max. 8 letters, digits, underscores. Note that case-insensitive string comparisons are performed to find triggers - "1", "01", "001" are all different values, "foo", "Foo" and "FoO" are equal values.

7.4.3 Example

```
0.00400000 52
    4.236     57768  11
    4.328     58964  12
    4.740     64320  13
    4.996     67648  14
    8.904    118452  91
   12.732    168216  31
   12.828    169464  32
   13.172    173936  33
```