Reference Manual for libRASCH-0.8.29

Raphael Schneider

Reference Manual for libRASCH-0.8.29
by Raphael Schneider

libRASCH Reference Manual Version 0.1.0 Edition
Published 2004

Table of Contents

List of Tables

Chapter 1. Function Reference for C

No special intro text.

ra_lib_init

Name

`ra_lib_init` — init libRASCH

Synopsis

```
LIBRAAPI ra_handle ra_lib_init (void);
```

Arguments

None

Description

Init libRASCH, read config-file and init plugins; returns ra_handle.

source file

ra.c

ra_lib_close

Name

`ra_lib_close` — close libRASCH

Synopsis

```
LIBRAAPI void ra_lib_close (ra_handle h);
```

Arguments

h

handle to ra instance

Description

Close libRASCH, close plugins and frees all allocated memory.

source file

ra.c

ra_lib_get_error

Name

`ra_lib_get_error` — get last error

Synopsis

```
LIBRAAPI long ra_lib_get_error (ra_handle h, char * text, size_t len);
```

Arguments

h

handle to ra instance

`text`

character buffer which receives error text

`len`

size of character buffer text

Description

This function returns the last error occured in libRASCH (or in one of the plugins). See 'ra_error.h' for meaning of returned number. If the number is negative, the last error was an OS specific error and the error number comes from the OS (just remove the minus sign). If text is not NULL, a short error description will be returned in text.

source file

ra.c

ra_lib_handle_from_any_handle

Name

`ra_lib_handle_from_any_handle` — retrive handle of libRASCH instance from any handle

Synopsis

```
LIBRAAPI ra_handle ra_lib_handle_from_any_handle (any_handle h);
```

Arguments

`h`

handle

Description

Returns handle of libRASCH instance from any handle.

source file

ra.c

ra_meas_handle_from_any_handle

Name

`ra_meas_handle_from_any_handle` — retrive measurement-handle from any handle

Synopsis

```
LIBRAAPI meas_handle ra_meas_handle_from_any_handle (any_handle h);
```

Arguments

h

handle

Description

Returns measurement-handle.

source file

ra.c

ra_lib_use_plugin

Name

`ra_lib_use_plugin` — set using a plugin (or not)

Synopsis

```
LIBRAAPI int ra_lib_use_plugin (ra_handle h, int plugin_index, int use_it);
```

Arguments

`h`

handle to libRASCH

`plugin_index`

index of plugin to change

`use_it`

flag using plugin (1: use plugin, !1: do not use plugin)

Description

Use or do not use plugin.

source file

ra.c

ra_plugin_get_by_num

Name

`ra_plugin_get_by_num` — get plugin-handle

Synopsis

`LIBRAAPI plugin_handle` **`ra_plugin_get_by_num`** `(ra_handle` *`h`*`, int` *`plugin_num`*`, int` *`search_all`*`);`

Arguments

`h`

handle to libRASCH

`plugin_num`

number of plugin

`search_all`

flag if all plugins should be searched (or only those with the use-it-flag enabled)

Description

Returns the plugin-handle for plugin #plugin_num.

source file

ra.c

ra_plugin_get_by_name

Name

`ra_plugin_get_by_name` — get plugin-handle

Synopsis

`LIBRAAPI plugin_handle` **`ra_plugin_get_by_name`** `(ra_handle` *`h`*`, const char *` *`name`*`, int` *`search_all`*`);`

Arguments

h

handle to libRASCH

name

name of plugin

search_all

flag if all plugins should be searched (or only those with the use-it-flag enabled)

Description

Returns the plugin-handle for plugin with name name.

source file

ra.c

ra_meas_find_first

Name

`ra_meas_find_first` — find libRASCH-handled measurements (return first one)

Synopsis

```
LIBRAAPI ra_find_handle ra_meas_find_first (ra_handle h, const char * dir,
struct ra_find_struct * rfs);
```

Arguments

h

handle to libRASCH

dir

directory which should be scanned for measurements

rfs

pointer to ra_find_struct which will held the infos for the found measurement.

Description

Search for libRASCH-handled measurements in dir. If at least on measurement was found a valid (!NULL) ra_find_handle will be returned and the infos about the measurement is in mfs.

source file

ra.c

ra_meas_find_next

Name

`ra_meas_find_next` — find libRASCH-handled measurements (returns next)

Synopsis

```
LIBRAAPI int ra_meas_find_next (ra_find_handle h, struct ra_find_struct *
rfs);
```

Arguments

h

find-handle

rfs

pointer to ra_find_struct which will held the infos for the found measurement.

Description

Returns 1 if another measurement was found (the information will be in mfs) or 0 if there are no more measurements searched with ra_meas_find_first().

source file

ra.c

ra_meas_close_find

Name

`ra_meas_close_find` — close the search for libRASCH-handled measurements

Synopsis

```
LIBRAAPI void ra_meas_close_find (ra_find_handle h);
```

Arguments

h

find-handle

Description

End a search for libRASCH-handled measurements (frees some memory). Must be called if the ra_find_handle h returned from ra_meas_find_first() will no be longer used.

source file

ra.c

ra_meas_open

Name

`ra_meas_open` — open a measurement

Synopsis

```
LIBRAAPI meas_handle ra_meas_open (ra_handle h, const char * file, const char
* eval_file, int fast);
```

Arguments

h

handle to libRASCH

file

file-name of the measurement

eval_file

file-name of the evaluation

fast

flag if some init-code should be done (=0) or not (=1)

Description

Opens the measurement file and returns a handle to the measurement. If the fast-flag is set, some initialisation-code (e.g. reading evaluation infos from the files) will not be done. This can be useful if only information about the measurment-object is needed but not about the evaluation. Getting the evaluation infos can be "very" time consuming.

source file

ra.c

ra_meas_new

Name

`ra_meas_new` — creates a new measurement

Synopsis

```
LIBRAAPI meas_handle ra_meas_new (ra_handle h, const char * dir, const char *
name);
```

Arguments

h

handle to libRASCH

dir

directory where the measurement-files should be stored

name

name of the measurement

Description

Creates a new measurement with the name name in the directory dir. The function returns the handle to the measurement. The measurement will be stored using the libRASCH file-format. Not full implemented. !!!

source file

ra.c

ra_meas_save

Name

`ra_meas_save` — saves a measurement

Synopsis

```
LIBRAAPI int ra_meas_save (meas_handle mh);
```

Arguments

mh

handle to measurement

Description

Saves the changes to a measurement (NOT to the evaluation). Plugin must support this. Not full implemented. !!!

source file

ra.c

ra_meas_close

Name

`ra_meas_close` — close a measurement

Synopsis

```
LIBRAAPI void ra_meas_close (meas_handle mh);
```

Arguments

mh

handle to measurement

Description

Close a measurement.

source file

ra.c

ra_info_get

Name

`ra_info_get` — get infos

Synopsis

```
LIBRAAPI int ra_info_get (any_handle h, int id, value_handle vh);
```

Arguments

h

handle

id

id of information wanted

vh

value_handle receiving the info

Description

Returns 0 if the wanted information was found and != 0 if not. The information will be in vh. For usage of the function please see user-manual.

source file

ra.c

ra_info_get_by_name

Name

`ra_info_get_by_name` — get infos

Synopsis

```
LIBRAAPI int ra_info_get_by_name (any_handle h, const char * name,
value_handle vh);
```

Arguments

h

handle

name

name (text string) of information wanted

vh

value_handle receiving the info

Description

Returns 0 if the wanted information was found and != 0 if not. The information will be in vh. For usage of the function please see user-manual.

source file

ra.c

ra_info_get_by_idx

Name

`ra_info_get_by_idx` — get infos

Synopsis

```
LIBRAAPI int ra_info_get_by_idx (any_handle h, int info_type, int idx,
value_handle vh);
```

Arguments

h

measurement-handle

info_type

type of information wanted

idx

index of information wanted

vh

value_handle receiving the info

Description

Returns 0 if the wanted information was found and != 0 if not. The information will be in vh. For usage of the function please see user-manual.

source file

ra.c

ra_info_set

Name

`ra_info_set` — set info

Synopsis

```
LIBRAAPI int ra_info_set (any_handle h, int id, value_handle vh);
```

Arguments

h

handle

id

info id

vh

value_handle with the info

Description

Set info 'id' with the data found in 'vh'. At the moment only some measurement and recording infos can be set. In the future, other infos (e.g. eval-name, eval-desc etc.) can be set with this function also.

source file

ra.c

ra_rec_get_first

Name

`ra_rec_get_first` — get root-recording

Synopsis

```
LIBRAAPI rec_handle ra_rec_get_first (meas_handle mh, long session);
```

Arguments

mh

measurement-handle

session

session number

Description

Returns the first recording-handle of session session.

source file

ra.c

ra_rec_get_next

Name

`ra_rec_get_next` — get next recording

Synopsis

```
LIBRAAPI rec_handle ra_rec_get_next (rec_handle rh);
```

Arguments

rh

recording-handle

Description

Returns the next recording which follows recording rh.

source file

ra.c

ra_rec_get_first_child

Name

`ra_rec_get_first_child` — get first sub-recording

Synopsis

```
LIBRAAPI rec_handle ra_rec_get_first_child (rec_handle rh);
```

Arguments

rh

recording-handle

Description

Returns the first child-recording of recording rh.

source file

ra.c

ra_rec_add

Name

`ra_rec_add` — add a recording

Synopsis

```
LIBRAAPI rec_handle ra_rec_add (meas_handle mh, rec_handle parent);
```

Arguments

mh

handle of the measurement

parent

handle of the parent recording (or NULL)

Description

Adds a recording to a measurement. The recording will be a child-recording of the parent-recording parent. The recording-handle will be returned. Not implemented yet. !!!

source file

ra.c

ra_dev_add

Name

`ra_dev_add` — add a recording-device

Synopsis

```
LIBRAAPI int ra_dev_add (rec_handle rh);
```

Arguments

rh

handle of the recording

Description

Adds a device to the recording rh. The number of the device will be returned. Not implemented yet. !!!

source file

ra.c

ra_ch_add

Name

`ra_ch_add` — add a recording-channel

Synopsis

```
LIBRAAPI int ra_ch_add (rec_handle rh);
```

Arguments

rh

handle of the recording

Description

Adds a channel to the recording rh. The number of the channel will be returned. Not implemented yet. !!!

source file

ra.c

ra_session_new

Name

`ra_session_new` — start a new recording session

Synopsis

```
LIBRAAPI int ra_session_new (meas_handle mh);
```

Arguments

mh

handle of the measurement

Description

Starts a new recording session and close the previous one. The function returns the new session number. Not implemented yet. !!!

source file

ra.c

ra_raw_add

Name

`ra_raw_add` — add raw data

Synopsis

```
LIBRAAPI size_t ra_raw_add (meas_handle mh, unsigned int ch, value_handle
vh);
```

Arguments

mh

handle of the measurement

ch

channel where data should be added

vh

data to be added

Description

Adds raw data to a measurement. The data is added to the current session. Not implemented yet. !!!

source file

ra.c

ra_raw_get

Name

`ra_raw_get` — get raw-signal data

Synopsis

```
LIBRAAPI size_t ra_raw_get (rec_handle rh, unsigned int ch, size_t start,
size_t num_data, DWORD * data, DWORD * data_high);
```

Arguments

rh

recording-handle

ch

channel

start

first sample

num_data

number of samples

data

buffer for samples

data_high

buffer for higher 4 bytes of samples (if size of one sample > 4 bytes)

Description

This function returns the sample's (raw-signal data) from the measurement mh and the recording rh. The first sample in data will be the sample number start (in sampleunits) and max. num_data are in data. The memory for buffer data must be allocated in calling function. The samples returned in this function are the samples as stored in the file. If the size of one sample is greater than 4 bytes, the upper part of the samples are stored in the data_high buffer. If the size is not greater than 4 bytes, the buffer is not needed.

source file

ra.c

ra_raw_get_unit

Name

`ra_raw_get_unit` — get raw-signal data scaled to 'unit'-values

Synopsis

```
LIBRAAPI size_t ra_raw_get_unit (rec_handle rh, unsigned int ch, size_t
start, size_t num_data, double * data);
```

Arguments

rh

recording-handle

ch

channel

start

first sample

num_data

number of samples

data

buffer for samples

Description

This function returns the sample's (raw-signal data) from the measurement mh and the recording rh. The first sample in data will be the sample number start (in sampleunits) and max. num_data are in data. The memory for buffer data must be allocated in calling function. The values are scaled to the unit of the channel.

source file

ra.c

ra_gui_call

Name

`ra_gui_call` — shows GUI-element of a plugin (if avail.)

Synopsis

```
LIBRAAPI int ra_gui_call (any_handle h, plugin_handle pl);
```

Arguments

`h`

processing-handle or view-handle

`pl`

plugin-handle

Description

If the plugin pl provides a GUI-element, this function shows this GUI-element and transfer control to it. The h variable contains some information, needed by the plugin. For more information if a plugin provides a GUI-element and what type/information is needed, please see the plugin specific documentation.

source file

ra.c

ra_proc_get

Name

`ra_proc_get` — get a processing handle

Synopsis

```
LIBRAAPI proc_handle ra_proc_get (meas_handle mh, plugin_handle pl, void
(*callback);
```

Arguments

`mh`

pl

  plugin-handle

*(*callback*

Description

This function returns a proc_handle for the plugin pl. The proc_handle will be returned initialized.

source file

ra.c

ra_proc_free

Name

`ra_proc_free` — free a processing handle

Synopsis

```
LIBRAAPI void ra_proc_free (proc_handle proc);
```

Arguments

proc

  processing-handle

Description

This function frees a processing-handle and all associated memory.

source file

ra.c

ra_proc_do

Name

`ra_proc_do` — process the signal

Synopsis

```
LIBRAAPI int ra_proc_do (proc_handle proc);
```

Arguments

proc

processing-handle

Description

This function calls the "processing" function of a process-plugin. For more information what information in proc is needed, please see the plugin specific documentation.

source file

ra.c

ra_proc_get_result

Name

`ra_proc_get_result` — get the processing results

Synopsis

```
LIBRAAPI int ra_proc_get_result (proc_handle proc, long res_num, long
res_set, value_handle vh);
```

Arguments

proc

processing-handle

res_num

number (zero-based index) of the wanted result

res_set

number (zero-based index) of the wanted result-set

vh

value-handle

Description

This function returns in vh one result from a processing perfomed with ra_proc_do.

source file

ra.c

ra_lib_get_option

Name

`ra_lib_get_option` — returns an option value

Synopsis

```
LIBRAAPI int ra_lib_get_option (any_handle h, const char * opt_name,
value_handle vh);
```

Arguments

h

return option from this object

opt_name

name of the option

vh

after function-call, contains option value

Description

The function tries to return the option opt_name from the object h. At the moment only options for process-plugins (h needs to be a process-handle) are supported.

source file

ra.c

ra_lib_set_option

Name

`ra_lib_set_option` — set an option

Synopsis

```
LIBRAAPI int ra_lib_set_option (any_handle h, const char * opt_name,
value_handle vh);
```

Arguments

h

option will be set in this object

opt_name

name of the option

vh

option value

Description

The function set the option opt_name in object h. At the moment only options for process-plugins (h needs to be a process-handle) are supported.

source file

ra.c

ra_meas_copy

Name

`ra_meas_copy` — copy measurement

Synopsis

```
LIBRAAPI int ra_meas_copy (meas_handle mh, const char * dest_dir);
```

Arguments

mh

handle of measurement which should be copied

dest_dir

destination directory

Description

The function copy the files of measurement mh to directory dest_dir. IMPORTANT!!! Function _not_ completely tested. Use at your own risk.

source file

ra.c

ra_meas_move

Name

`ra_meas_move` — move measurement

Synopsis

```
LIBRAAPI meas_handle **ra_meas_move** (meas_handle *mh*, const char * *dest_dir*);
```

Arguments

mh

handle of measurement which should be moved

dest_dir

destination directory

Description

The function move the files of measurement mh to directory dest_dir. After the move the measurement will be reopend (to be able to handle the new positions of the files) and the functions returns the new measurement-handle. IMPORTANT!!! Function _not_ completely tested. Use at your own risk.

source file

ra.c

ra_meas_delete

Name

`ra_meas_delete` — delete measurement

Synopsis

```
LIBRAAPI int **ra_meas_delete** (meas_handle *mh*);
```

Arguments

mh

handle of measurement which should be deleted

Description

The function deletes the files of measurement mh. The measurement mh will be closed after the deletion. Therefore mh is no longer a valid measurement-handle. IMPORTANT!!! Function _not_ completely tested. Use at your own risk.

source file

ra.c

ra_raw_process

Name

`ra_raw_process` — common processing tasks for raw-data (e.g. powerline-noise filter)

Synopsis

```
LIBRAAPI int ra_raw_process (long task, value_handle task_data, size_t
num_data, DWORD * data, DWORD * data_high, rec_handle rh, unsigned int ch);
```

Arguments

task

id of the processing task (see RA_RAW_PROC_* defines)

task_data

task specific data

num_data

number of raw-data samples

data

raw-data

data_high

upper 32bits of raw-data (if we have 64bit samples; not supported yet)

rh

recording handle the raw-data comes from

ch

channel of the raw-data

Description

This function performs common processing tasks for the raw-data, the tasks are independent of signal-type. For example, one task is to filer powerline noise. For a complete set of available tasks, see the RA_RAW_PROC_* defines in the ra_defines.h header file.

source file

raw_process.c

ra_est_ch_type

Name

`ra_est_ch_type` — estimate channel type

Synopsis

```
LIBRAAPI int ra_est_ch_type (ra_handle ra, const char * folder, const char *
filename, int is_dir_based, const char * name, struct ra_est_ch_infos * inf);
```

Arguments

`ra`

ra-handle, needed to check for system-wide ch-map

`folder`

folder to look for the folder specific ch-map

`filename`

filename of the measurement, needed to look for the meas-specific ch-map

`is_dir_based`

flag if measurement files are stored in a folder

`name`

channel name

`inf`

this variable contains the estimated type

Description

This function tries to estimate the type of a recording channel from the channel name.

source file

estimate_ch_infos.c

ra_value_malloc

Name

`ra_value_malloc` — allocate and initialize a value_handle

Synopsis

```
LIBRAAPI value_handle ra_value_malloc (void);
```

Arguments

None

Description

Returns a initialzed value-handle.

source file

value.c

ra_value_free

Name

`ra_value_free` — free value-handle

Synopsis

```
LIBRAAPI void ra_value_free (value_handle vh);
```

Arguments

vh

   value-handle

Description

Frees the memory asocciated with vh.

source file

value.c

ra_value_reset

Name

`ra_value_reset` — reset value-handle

Synopsis

```
LIBRAAPI void ra_value_reset (value_handle vh);
```

Arguments

vh

value-handle

Description

Reset value-handle vh.

source file

value.c

ra_value_get_type

Name

`ra_value_get_type` — return value-type

Synopsis

```
LIBRAAPI long ra_value_get_type (value_handle vh);
```

Arguments

vh

value-handle

Description

Return the type of the value stored in vh. Check in ra_defines.h for the meaning of the returned value (RA_VALUE_TYPE_*).

source file

value.c

ra_value_is_ok

Name

`ra_value_is_ok` — checks if value is valid

Synopsis

```
LIBRAAPI int ra_value_is_ok (value_handle vh);
```

Arguments

`vh`

value-handle

Description

Returns ’1’ if value stored in vh is valid, ’0’ if not.

source file

value.c

ra_value_get_num_elem

Name

`ra_value_get_num_elem` — return number of elements of array values

Synopsis

```
LIBRAAPI unsigned long ra_value_get_num_elem (value_handle vh);
```

Arguments

`vh`

value-handle

Description

When an array is stored in vh, the number of elements of the array is returned.

source file

value.c

ra_value_get_info

Name

`ra_value_get_info` — returns id of info

Synopsis

```
LIBRAAPI long ra_value_get_info (value_handle vh);
```

Arguments

vh

value-handle

Description

Returns the id of the info set in vh. Check in ra_defines.h for for the meaning of info-id (RA_INFO_*).

source file

value.c

ra_value_set_info

Name

`ra_value_set_info` — sets the RA_INFO_* id of info

Synopsis

```
LIBRAAPI int ra_value_set_info (value_handle vh, long id);
```

Arguments

vh

value-handle

id

info-id

Description

Sets the id of the info in vh. Check in ra_defines.h for for the meaning of info-id (RA_INFO_*).

source file

value.c

ra_value_get_name

Name

`ra_value_get_name` — returns name of info

Synopsis

```
LIBRAAPI const char * ra_value_get_name (value_handle vh);
```

Arguments

vh

value-handle

Description

Returns a character pointer to the name of the info set in vh. The pointer is valid until vh is free'd or a new info was retrived.

source file

value.c

ra_value_get_desc

Name

`ra_value_get_desc` — returns description of info

Synopsis

```
LIBRAAPI const char * ra_value_get_desc (value_handle vh);
```

Arguments

vh

value-handle

Description

Returns a character pointer to the description of the info set in vh. The pointer is valid until vh is free'd or a new info was retrived.

source file

value.c

ra_value_info_modifiable

Name

`ra_value_info_modifiable` — returns flag if info in measurement file(s) can be modified

Synopsis

```
LIBRAAPI int ra_value_info_modifiable (value_handle vh);
```

Arguments

vh

value-handle

Description

Returns a flag if the info currently handled by the value-handle, can be modified in the measurement file(s).

source file

value.c

ra_value_info_set_modifiable

Name

`ra_value_info_set_modifiable` — set flag if info in measurement file(s) can be modified

Synopsis

```
LIBRAAPI void ra_value_info_set_modifiable (value_handle vh, int
can_be_modified);
```

Arguments

vh

value-handle

can_be_modified

Description

Sets the flag if the info currently handled by the value-handle, can be modified in the measurement file(s).

source file

value.c

ra_value_set_number

Name

`ra_value_set_number` — set a number in a value-handle

Synopsis

```
LIBRAAPI int ra_value_set_number (value_handle vh, long number);
```

Arguments

vh

value-handle

number

number

Description

Set the number number in the value-handle vh. This number is used when infos about channels, recording devices and plugins. number is also used when processing results are retrived.

source file

value.c

ra_value_get_number

Name

`ra_value_get_number` — get a number set in a value-handle

Synopsis

```
LIBRAAPI long ra_value_get_number (value_handle vh);
```

Arguments

vh

value-handle

Description

Returns the number number set in the value-handle vh.

source file

value.c

ra_value_set_short

Name

`ra_value_set_short` — set short value in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_short (value_handle vh, const short value);
```

Arguments

vh

value-handle

`value`

value to set in ra_info

Description

This function set the value in vh.

source file

value.c

ra_value_set_long

Name

`ra_value_set_long` — set long value in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_long (value_handle vh, const long value);
```

Arguments

`vh`

value-handle

`value`

value to set in ra_info

Description

This function set the value in vh.

source file

value.c

ra_value_set_double

Name

`ra_value_set_double` — set double value in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_double (value_handle vh, const double value);
```

Arguments

vh

value-handle

value

value to set in ra_info

Description

This function set the value in vh.

source file

value.c

ra_value_set_string

Name

`ra_value_set_string` — set a string in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_string (value_handle vh, const char * string);
```

Arguments

`vh`

value-handle

`string`

string to set in ra_info

Description

This function set the string ('\\0'-ended char *) in vh.

source file

value.c

ra_value_set_string_utf8

Name

`ra_value_set_string_utf8` — set a UTF-8 encoded string in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_string_utf8 (value_handle vh, const char *
string);
```

Arguments

vh

value-handle

string

string to set in ra_info

Description

This function set the string (‘\\0‘-ended char *) in vh.

source file

value.c

ra_value_set_voidp

Name

`ra_value_set_voidp` — set void-pointer value in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_voidp (value_handle vh, const void * value);
```

Arguments

vh

value-handle

value

value to set in ra_info

Description

This function set the value in vh.

source file

value.c

ra_value_set_short_array

Name

`ra_value_set_short_array` — set short-array values in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_short_array (value_handle vh, const short * array,
unsigned long num);
```

Arguments

vh

value-handle

array

array to set in ra_info

num

number of array elements

Description

This function set the short array array with num elements in vh.

source file

value.c

ra_value_set_long_array

Name

`ra_value_set_long_array` — set long-array values in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_long_array (value_handle vh, const long * array,
unsigned long num);
```

Arguments

vh

value-handle

array

array to set in ra_info

num

number of array elements

Description

This function set the long array array with num elements in vh.

source file

value.c

ra_value_set_double_array

Name

`ra_value_set_double_array` — set double-array values in value-handle

Synopsis

LIBRAAPI void **ra_value_set_double_array** (value_handle *vh*, const double * *array*, unsigned long *num*);

Arguments

vh

value-handle

array

array to set in ra_info

num

number of array elements

Description

This function set the double array array with num elements in vh.

source file

value.c

ra_value_set_string_array

Name

`ra_value_set_string_array` — set a string-array in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_string_array (value_handle vh, const char **
*array, unsigned long num);
```

Arguments

vh

value-handle

**array*

num

number of array elements

Description

This function set the string-array array with num elements in vh.

source file

value.c

ra_value_set_string_array_utf8

Name

`ra_value_set_string_array_utf8` — set a string-array (UTF-8 encoded) in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_string_array_utf8 (value_handle vh, const char **
*array, unsigned long num);
```

Arguments

vh

value-handle

**array*

num

number of array elements

Description

This function set the string-array ’array’ with ’num’ elements in vh. The strings in ’array’ are expected to be UTF-8 encoded.

source file

value.c

ra_value_set_voidp_array

Name

`ra_value_set_voidp_array` — set void-pointer-array values in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_voidp_array (value_handle vh, const void **
*array, unsigned long num);
```

Arguments

vh

value-handle

**array*

num

number of array elements

Description

This function set the long array array with num elements in vh.

source file

value.c

ra_value_set_vh_array

Name

`ra_value_set_vh_array` — set value-handle-array values in value-handle

Synopsis

```
LIBRAAPI void ra_value_set_vh_array (value_handle vh, const value_handle *
array, unsigned long num);
```

Arguments

vh

value-handle

array

array to set in ra_value

num

number of array elements

Description

This function set the value-handle array ’array’ with num elements in vh.

source file

value.c

ra_value_get_short

Name

`ra_value_get_short` — return short value from value-handle

Synopsis

```
LIBRAAPI short ra_value_get_short (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns the short value set in vh.

source file

value.c

ra_value_get_long

Name

`ra_value_get_long` — return long value from value-handle

Synopsis

```
LIBRAAPI long ra_value_get_long (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns the long value set in vh.

source file

value.c

ra_value_get_double

Name

`ra_value_get_double` — return double value from value-handle

Synopsis

```
LIBRAAPI double ra_value_get_double (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns the double value set in vh.

source file

value.c

ra_value_get_string

Name

`ra_value_get_string` — return character pointer from value-handle

Synopsis

```
LIBRAAPI const char * ra_value_get_string (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a character pointer to the string set in vh. The pointer is valid until vh is used in another function or vh is freed.

source file

value.c

ra_value_get_string_utf8

Name

`ra_value_get_string_utf8` — return character pointer from value-handle (UTF-8 encoded)

Synopsis

```
LIBRAAPI const char * ra_value_get_string_utf8 (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a character pointer to the string set in vh. The string is UTF-8 encoded. The pointer is valid until vh is used in another function or vh is freed.

source file

value.c

ra_value_get_voidp

Name

`ra_value_get_voidp` — return void-pointer from value-handle

Synopsis

```
LIBRAAPI const void * ra_value_get_voidp (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns the void-pointer set in vh.

source file

value.c

ra_value_get_short_array

Name

`ra_value_get_short_array` — return pointer to short array from value-handle

Synopsis

```
LIBRAAPI const short * ra_value_get_short_array (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a pointer to the short array stored in vh. The pointer is valid until vh is used in another function or vh is freed. The number of elements of the array can be get by calling ra_value_get_num_elem().

source file

value.c

ra_value_get_long_array

Name

`ra_value_get_long_array` — return pointer to long array from value-handle

Synopsis

```
LIBRAAPI const long * ra_value_get_long_array (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a pointer to the long array stored in vh. The pointer is valid until vh is used in another function or vh is freed. The number of elements of the array can be get by calling ra_value_get_num_elem().

source file

value.c

ra_value_get_double_array

Name

`ra_value_get_double_array` — return pointer to double array from value-handle

Synopsis

```
LIBRAAPI const double * ra_value_get_double_array (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a pointer to the double array stored in vh. The pointer is valid until vh is used in another function or vh is freed. The number of elements of the array can be get by calling ra_value_get_num_elem().

source file

value.c

ra_value_get_string_array

Name

`ra_value_get_string_array` — return pointer to character pointer array from value-handle

Synopsis

```
LIBRAAPI const char ** ra_value_get_string_array (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a pointer to the character pointer array stored in vh. The pointer is valid until vh is used in another function or vh is freed. The number of elements of the array can be get by calling ra_value_get_num_elem().

source file

value.c

ra_value_get_string_array_utf8

Name

`ra_value_get_string_array_utf8` — return pointer to character pointer array from value-handle (UTF-8 encoded)

Synopsis

```
LIBRAAPI const char ** ra_value_get_string_array_utf8 (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a pointer to the character pointer array stored in vh. The strings are UTF-8 encoded. The pointer is valid until vh is used in another function or vh is freed. The number of elements of the array can be get by calling ra_value_get_num_elem().

source file

value.c

ra_value_get_voidp_array

Name

`ra_value_get_voidp_array` — return pointer to void-pointer array from value-handle

Synopsis

```
LIBRAAPI const void ** ra_value_get_voidp_array (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a pointer to the void-pointer array stored in vh. The pointer is valid until vh is used in another function or vh is freed. The number of elements of the array can be get by calling ra_value_get_num_elem().

source file

value.c

ra_value_get_vh_array

Name

`ra_value_get_vh_array` — return pointer to long array from value-handle

Synopsis

```
LIBRAAPI const value_handle * ra_value_get_vh_array (value_handle vh);
```

Arguments

vh

value-handle

Description

This function returns a pointer to the value-handle array stored in vh. The pointer is valid until vh is used in another function or vh is freed. The number of elements of the array can be get by calling ra_value_get_num_elem().

source file

value.c

ra_value_copy

Name

`ra_value_copy` — copy one value_handle to another value_handle

Synopsis

```
LIBRAAPI int ra_value_copy (value_handle dest, value_handle src);
```

Arguments

dest

destination value-handle

src

source value-handle

Description

This function copies the values from one value-handle to another value-handle and the name and description. The remaining fields are not copied.

source file

value.c

ra_value_get_single_elem

Name

`ra_value_get_single_elem` — copy one element from a value-handle in another value-handle

Synopsis

`LIBRAAPI int` **`ra_value_get_single_elem`** `(value_handle` *`dest`*`, value_handle` *`src`*`, size_t` *`elem_num`*`);`

Arguments

dest

destination value-handle

src

source value-handle

elem_num

zero-based index of the value which has to be copied

Description

This function copies one element from value-handle to another value-handle and the name and description. The remaining fields are not copied. The function is helpfull when a single value from an array, stored in a value-handle, is needed for another function expecting a single value in a value-handle.

source file

value.c

ra_list_add

Name

`ra_list_add` — add an entry to a linked list

Synopsis

`int` **`ra_list_add`** `(void **` *`*head`*`, void *` *`item`*`);`

Arguments

**head*

item

item to be added

Description

This function adds ’item’ to the linked list identified by ’head’ (head of list).

source file

linked_list.c

ra_list_insert

Name

`ra_list_insert` — insert an entry to a linked list

Synopsis

```
int ra_list_insert (void ** *head, void * prev, void * item);
```

Arguments

**head*

prev

previous item

item

item to be added

Description

This function inserts ’item’ in a linked list (identified by ’head’). The item will come directly after ’prev’.

source file

linked_list.c

ra_list_del

Name

`ra_list_del` — delete an entry from a linked list

Synopsis

```
int ra_list_del (void ** *head, void * item);
```

Arguments

**head*

item

item to be deleted

Description

This function deletes 'item' from a linked list (identified by 'head').

source file

linked_list.c

ra_comm_add

Name

`ra_comm_add` — add a signal receiver

Synopsis

```
LIBRAAPI int ra_comm_add (meas_handle mh, plugin_handle p, dest_handle dh,
const char * sig_name);
```

Arguments

mh

measurement handle

p

plugin handle

dh

destination handle

sig_name

name of the signal

Description

This function adds a new signal receiver p and dh for the signal sig_name to the inter-plugin-communication. Whenever the signal sig_name is emitted, the signal will be send to the plugin p. The plugin-instance will be identified with the help of dh.

source file

pl_comm.c

ra_comm_del

Name

`ra_comm_del` — delete a signal receiver

Synopsis

```
LIBRAAPI void ra_comm_del (meas_handle mh, dest_handle dh, const char *
sig_name);
```

Arguments

mh

measurement handle

dh

destination handle

sig_name

name of the signal

Description

This function deletes the signal receiver dh for the signal sig_name.

source file

pl_comm.c

ra_comm_emit

Name

`ra_comm_emit` — emits a signal

Synopsis

```
LIBRAAPI void ra_comm_emit (meas_handle mh, const char * sender, const char *
sig_name, int num_para, struct comm_para * para);
```

Arguments

`mh`

measurement handle

`sender`

name of the sender (needed for debug, can be NULL)

`sig_name`

name of the signal

`num_para`

number of signal parameters

`para`

signal parameters

Description

This function emits the signal sig_name wit num_para parameters.

source file

pl_comm.c

ra_get_filename

Name

`ra_get_filename` — return "case-corrected" filename

Synopsis

```
int ra_get_filename (const char * fn, int dir_based, char * buf, int
buf_len);
```

Arguments

fn

input filename

dir_based

buf

buffer for "case-corrected" filename

buf_len

length of buffer ’buf’

Description

The function tries to find how the given filename is used on the current system (case wise) and returns the filename case corrected.

source file

ra_file.c

ra_fopen

Name

`ra_fopen` — opens a file

Synopsis

```
FILE * ra_fopen (const char * fn, const char * mode, int dir_based);
```

Arguments

fn

filename

mode

the fopen file-open mode characters

dir_based

flag if measurement-files are stored in a directory (=1) or not (!=1)

Description

The function tries different case combinations to open the file if the given filename can not be found.

source file

ra_file.c

ra_eval_attribute_list

Name

`ra_eval_attribute_list` — get from ’h’ a list of available attributes

Synopsis

```
LIBRAAPI int ra_eval_attribute_list (any_handle h, value_handle vh);
```

Arguments

h

an eval-/event-class-/event-prop-handle

vh

value-handle receiving the attribute list

Description

This function returns a list of all attributes associated to handle ’h’.

source file

eval.c

ra_eval_attribute_get

Name

`ra_eval_attribute_get` — get from ’h’ the value of the attribute ’name’

Synopsis

```
LIBRAAPI int ra_eval_attribute_get (any_handle h, const char * id,
value_handle vh);
```

Arguments

h

an eval-/event-class-/event-prop-handle

id

ASCII-id of the attribute

vh

value-handle receiving the value

Description

This function returns the value stored in the attribute ’name’.

source file

eval.c

ra_eval_attribute_set

Name

`ra_eval_attribute_set` — set the value of the attribute ’name’ in ’h’

Synopsis

```
LIBRAAPI int ra_eval_attribute_set (any_handle h, const char * id,
value_handle vh);
```

Arguments

h

an eval-/event-class-/event-prop-handle

id

ASCII-id of the attribute

vh

value-handle with the value

Description

This function sets the value of the attribute ’name’.

source file

eval.c

ra_eval_attribute_unset

Name

`ra_eval_attribute_unset` — removes the attribute ’name’ in ’h’

Synopsis

```
LIBRAAPI int ra_eval_attribute_unset (any_handle h, const char * id);
```

Arguments

h

an eval-/event-class-/event-prop-handle

id

ASCII-id of the attribute

Description

This function removes the attribute ’name’.

source file

eval.c

ra_eval_add

Name

`ra_eval_add` — adds an evaluation

Synopsis

```
LIBRAAPI eval_handle ra_eval_add (meas_handle mh, const char * name, const
char * desc, int original);
```

Arguments

mh

measurement-handle

name

a short name of the evaluation

desc

a short description of the evaluation

original

flag if evaluation is the original evaluation

Description

This function adds an evaluation to a measurement.

source file

eval.c

ra_eval_copy

Name

`ra_eval_copy` — copy evaluation

Synopsis

```
LIBRAAPI eval_handle ra_eval_copy (eval_handle eh, const char * name, const
char * desc);
```

Arguments

eh

source evaluation handle

name

name of the copied evaluation

desc

description of the copied evaluation

Description

The function copies the evaluation given by ’eh’, the handle of the copy will be returned. If the parameter "name" is NULL or an empty string, the name of the source evaluation will be used and the prefix ’copy-’ will be added.

source file

eval.c

ra_eval_delete

Name

`ra_eval_delete` — delete evaluation

Synopsis

```
LIBRAAPI int ra_eval_delete (eval_handle eh);
```

Arguments

eh

evaluation handle

Description

The function deletes the evaluation given by ’eh’.

source file

eval.c

ra_eval_get_all

Name

`ra_eval_get_all` — returns all evaluation-handles

Synopsis

```
LIBRAAPI int ra_eval_get_all (meas_handle mh, value_handle vh);
```

Arguments

mh

measurement-handle

vh

value-handle receiving the evaluation-handles

Description

This function returns all evaluation-handles in the evaluation file associated with mh.

source file

eval.c

ra_eval_get_original

Name

`ra_eval_get_original` — returns the original evaluation-handle

Synopsis

```
LIBRAAPI eval_handle ra_eval_get_original (meas_handle mh);
```

Arguments

mh

measurement-handle

Description

This function returns the original evaluation-handle.

source file

eval.c

ra_eval_get_default

Name

`ra_eval_get_default` — returns the default evaluation-handle

Synopsis

```
LIBRAAPI eval_handle ra_eval_get_default (meas_handle mh);
```

Arguments

mh

measurement-handle

Description

This function returns the default evaluation-handle.

source file

eval.c

ra_eval_set_default

Name

`ra_eval_set_default` — set an evaluation to the default one

Synopsis

```
LIBRAAPI int ra_eval_set_default (eval_handle eh);
```

Arguments

eh

evaluation handle

Description

The function sets the evaluation given by 'eh' to the default one.

source file

eval.c

ra_eval_get_handle

Name

`ra_eval_get_handle` — returns the evaluation handle an event-class belongs to

Synopsis

```
LIBRAAPI eval_handle ra_eval_get_handle (class_handle clh);
```

Arguments

clh

event-class handle

Description

The function returns the evaluation handle which the evant-class given by 'clh' belongs to.

source file

eval.c

ra_class_add

Name

`ra_class_add` — adds an user-defined event-class to an evaluation

Synopsis

```
LIBRAAPI class_handle ra_class_add (eval_handle eh, const char * id, const
char * name, const char * desc);
```

Arguments

`eh`

evaluation handle

`id`

ASCII-id of the event-class

`name`

a short name for the event-class

`desc`

a short description of the event-class

Description

The function adds an event-class to the evaluation 'eh'. The parameter 'id' is used to identifiy the event-class and must contain only ASCII characters. The parameters 'name' and 'desc' are used to describe the event-class. If you want to use a pre-defined event-class, use ra_class_add_predef().

source file

eval.c

ra_class_add_predef

Name

`ra_class_add_predef` — adds an pre-defined event-class to an evaluation

Synopsis

```
LIBRAAPI class_handle ra_class_add_predef (eval_handle eh, const char * id);
```

Arguments

`eh`

evaluation handle

`id`

ASCII-id of the event-class

Description

The function adds the event-class ’id’ to the evaluation ’eh’. ’id’ is one of the pre-defined event-classes in libRASCH. If you want to add not a pre-defined event-class, use ra_class_add().

source file

eval.c

ra_class_delete

Name

`ra_class_delete` — deletes an event-class

Synopsis

```
LIBRAAPI int ra_class_delete (class_handle clh);
```

Arguments

`clh`

event-class handle

Description

The function deletes the event-class ’clh’.

source file

eval.c

ra_class_get

Name

`ra_class_get` — return event-class

Synopsis

```
LIBRAAPI int ra_class_get (eval_handle eh, const char * id, value_handle vh);
```

Arguments

eh

evaluation handle

id

event-class ASCII-id

vh

value handle

Description

The function returns all event-classes with the id ’id’. If ’id’ is NULL (or is an empty string), all event-classes in the evaluation ’eh’ are given.

source file

eval.c

ra_class_add_event

Name

`ra_class_add_event` — add an event

Synopsis

```
LIBRAAPI long ra_class_add_event (class_handle clh, long start, long end);
```

Arguments

`clh`

event-class handle

`start`

start position of the event in sample-units

`end`

end position of the event in sample-units

Description

The function adds an event to the event-class ’clh’. The start and end of the event are given by ’start’ and ’end’ respectively. If the event is a point in time, use for ’end’ the same value as ’start’. A unique event-id is returned.

source file

eval.c

ra_class_add_event_mass

Name

`ra_class_add_event_mass` — add a list of events

Synopsis

```
LIBRAAPI int ra_class_add_event_mass (class_handle clh, unsigned long
num_events, const long * start, const long * end, long * ev_ids);
```

Arguments

clh

event-class handle

num_events

number of events to add

start

array with the start positions of the event in sample-units

end

array with the end positions of the event in sample-units

ev_ids

array receiving the event-ids

Description

The function adds a list of events to the event-class ’clh’. The start and end of the events are given by ’start’ and ’end’ respectively. The event-ids are returned in the array ’ev_ids’.

source file

eval.c

ra_class_del_event

Name

`ra_class_del_event` — delete an event

Synopsis

```
LIBRAAPI int ra_class_del_event (class_handle clh, long event_id);
```

Arguments

`clh`

event-class handle

`event_id`

event-id of the event

Description

The function deletes the event ’event_id’ from the event-class ’clh’.

source file

eval.c

ra_class_get_event_pos

Name

`ra_class_get_event_pos` — get the start and end position of an event

Synopsis

```
LIBRAAPI int ra_class_get_event_pos (class_handle clh, long event_id, long *
start, long * end);
```

Arguments

clh

event-class handle

event_id

event-id of the event

start

here the start position will be returned

end

here the end position will be returned

Description

The function returns in ’start’ and ’end’ the position of the evvent ’event_id’ from the event-class ’clh’. The positions are in sample-units.

source file

eval.c

ra_class_set_event_pos

Name

`ra_class_set_event_pos` — set the start and end position of an event

Synopsis

```
LIBRAAPI int ra_class_set_event_pos (class_handle clh, long event_id, long
start, long end);
```

Arguments

clh

event-class handle

event_id

event-id of the event

start

the start position of the event

end

the end position of the event

Description

The function sets the start and end position of the event 'event_id' in the event-class 'clh' to the values 'start' and 'end'.

source file

eval.c

ra_class_get_events

Name

`ra_class_get_events` — returns the events in a specific region

Synopsis

```
LIBRAAPI int ra_class_get_events (class_handle clh, long start, long end, int
complete, int sort, value_handle vh);
```

Arguments

clh

event-class handle

start

start of the region of interest

end

end of the region of interest

complete

flag if the events have to be complete in the region of interest

sort

flag if the found events should be sorted according their start position

vh

value handle receiving the event-id's asked for

Description

The function returns in 'vh' the events which are inside a specific region. The region of interest (ROI)is given by 'start' and 'end'. If the 'complete' flag is set, the events have to be complete inside the ROI. If it is not set, all events are returned which starts or ends inside the ROI.

source file

eval.c

ra_class_get_prev_event

Name

`ra_class_get_prev_event` — returns the event-id preceeding the given one

Synopsis

```
LIBRAAPI long ra_class_get_prev_event (class_handle clh, long event_id);
```

Arguments

clh

    event-class handle

event_id

    event-id

Description

The function returns the id of the event preceeding the one given in ’event_id’. If the given event is the first one ’-1’ is returned.

source file

eval.c

ra_class_get_next_event

Name

`ra_class_get_next_event` — returns the event-id following the given one

Synopsis

```
LIBRAAPI long ra_class_get_next_event (class_handle clh, long event_id);
```

Arguments

`clh`

  event-class handle

`event_id`

  event-id

Description

The function returns the id of the event following the one given in 'event_id'. If the given event is the first one '-1' is returned.

source file

eval.c

ra_class_get_handle

Name

`ra_class_get_handle` — returns the event-class handle the event-property belongs to

Synopsis

```
LIBRAAPI class_handle ra_class_get_handle (any_handle h);
```

Arguments

h

Description

The function returns the event-class handle the event-property 'ph' belongs to.

source file

eval.c

ra_prop_add

Name

`ra_prop_add` — adds an user-defined event-property to an event-class

Synopsis

```
LIBRAAPI prop_handle ra_prop_add (class_handle clh, const char * id, long
value_type, const char * name, const char * desc, const char * unit, int
use_minmax, double min, double max, int has_ignore_value, double
ignore_value);
```

Arguments

clh

event-class handle

id

ASCII-id of the event-property

value_type

RA_VALUE_TYPE_* of the values stored in the event-property

name

a short name for the event-property

desc

a short description of the event-property

unit

unit of the values stored in the event-property

use_minmax

flag if min-/max-values are valid

min

minimum value

max

maximum value

has_ignore_value

flag if 'ignore_value' is valid

ignore_value

a value which is interpreted as non-valid and can be ignored

Description

The function adds an event-property to the event-class 'clh'. The parameter 'id' is used to identifiy the event-property and must contain only ASCII characters. The parameters 'name' and 'desc' are used to describe the event-class. If you want to use a pre-defined event-property, use ra_prop_add_predef().

source file

eval.c

ra_prop_add_predef

Name

`ra_prop_add_predef` — adds an pre-defined event-property to an event-class

Synopsis

```
LIBRAAPI prop_handle ra_prop_add_predef (class_handle clh, const char * id);
```

Arguments

clh

event-class handle

id

ASCII-id of the event-property

Description

The function adds the event-property 'id' to the event-class 'clh'. 'id' is one of the pre-defined event-properties in libRASCH. If you want to add not a pre-defined event-property, use ra_prop_add().

source file

eval.c

ra_prop_delete

Name

`ra_prop_delete` — deletes an event-property

Synopsis

```
LIBRAAPI int ra_prop_delete (prop_handle ph);
```

Arguments

ph

event-property handle

Description

The function deletes the event-property ’ph’.

source file

eval.c

ra_prop_get_all

Name

`ra_prop_get_all` — returns all event-properties of an event-class

Synopsis

```
LIBRAAPI int ra_prop_get_all (class_handle clh, value_handle vh);
```

Arguments

clh

vh

ra-value handle receiving the event-property handles

Description

The function returns in ’vh’ all event-properties available in event-class ’clh’.

source file

eval.c

ra_prop_get

Name

`ra_prop_get` — returns event-property ’id’

Synopsis

```
LIBRAAPI prop_handle ra_prop_get (class_handle clh, const char * id);
```

Arguments

clh

event-class handle

id

ASCII id of the wanted event-property

Description

The function returns a handle to the event-property ’id’ from event-class ’clh’. If the event-property is not available ’NULL’ is returned.

source file

eval.c

ra_prop_set_value

Name

`ra_prop_set_value` — set value in event-property

Synopsis

```
LIBRAAPI int ra_prop_set_value (prop_handle ph, long event_id, long ch,
value_handle vh);
```

Arguments

ph

event-property handle

event_id

event-id

ch

channel number

vh

contains value which should be set

Description

The function set the value stored in ’vh’ in event-property ’ph’ for event-id ’event_id’ and channel ’ch’. If the value is channel independent use ’-1’ for the channel number.

source file

eval.c

ra_prop_set_value_mass

Name

`ra_prop_set_value_mass` — set multiple values in event-property

Synopsis

```
LIBRAAPI int ra_prop_set_value_mass (prop_handle ph, const long * event_id,
const long * ch, value_handle vh);
```

Arguments

ph

event-property handle

event_id

array containing event-id’s

ch

array containing channel number’s

vh

contains values to be set

Description

The function set the values stored in ’vh’ in event-property ’ph’ for the event-id’s ’event_id’ and channel’s ’ch’. If the value is channel independent use ’-1’ for the channel numbers. The length of ’event_id’ and ’ch’ has to be the same. And the number of values in ’vh’ has to be the same as in ’event_id’/’ch’.

source file

eval.c

ra_prop_get_ch

Name

`ra_prop_get_ch` — returns the channels for which data is available

Synopsis

```
LIBRAAPI int ra_prop_get_ch (prop_handle ph, long event_id, value_handle vh);
```

Arguments

ph

event-property handle

event_id

event-id

vh

ra-value receiving the channel-list

Description

The function return in ’vh’ the list of channels where values are available for event-id ’event_id’ and event-property ’ph’.

source file

eval.c

ra_prop_get_value

Name

`ra_prop_get_value` — get value from event-property

Synopsis

LIBRAAPI int **ra_prop_get_value** (prop_handle *ph*, long *event_id*, long *ch*, value_handle *vh*);

Arguments

ph

event-property handle

event_id

event-id

ch

channel number

vh

receives value

Description

The function returns a value in ’vh’ from event-property ’ph’ for event-id ’event_id’ and channel ’ch’. If you interested in the channel independent value use ’-1’ for the channel number.

source file

eval.c

ra_eval_save

Name

`ra_eval_save` — save evalution

Synopsis

```
LIBRAAPI int ra_eval_save (meas_handle mh, const char * file, int use_ascii);
```

Arguments

mh

measurement handle

file

name of the evaluation file (optional)

use_ascii

flag if event-values should be stored as ASCII-text (=1) or as

Description

binary MIME64 encoded data (=0) This function saves the evaluation(s) that belongs to the measurement mh. If no filename file is given, the default-filename will be used. If use_ascii is !=0 than the event values are stored as ASCII text.

source file

save_eval_v1.c

Chapter 2. Structure Reference for C

No special intro text.

Just a dummy text.

Chapter 3. Function Reference for Perl (OO-Interface)

Nothing special.

Just a dummy text.

Chapter 4. Function Reference for Python (OO-Interface)

No special intro.

Just a dummy text.

Chapter 5. Function Reference for Matlab/Octave

No special intro.

Just a dummy text.

Chapter 6. Infos Reference

The tables in this section list the informations, which are available from libRASCH. For measurements, recordings and measurement objects not all informations must be available.

The first column ('Info Name') list the info-name which must be used in the Perl, Python, Matlab, Octave functions to get informations. The second column ('Info Constant') list the constants used in C/C++ programs to get informations. The third column ('Info Description') gives a short description of the information returned from libRASCH.

Table 6-1. List of info-names for library infos -- ra_lib_get_info

Info Name	Info Constant	Info Description
num_plugins	RA_INFO_NUM_PLUGINS_L	number of plugins
lib_version	RA_INFO_VERSION_C	version of libRASCH

Table 6-2. List of info-names for measurement infos -- ra_meas_get_info

Info Name	Info Constant	Info Description
num_sessions	RA_INFO_NUM_SESSIONS_L	number of sessions in a measurement
num_obj_infos	RA_INFO_NUM_OBJ_INFOS_L	number of infos about a measurment object
num_rec_gen_infos	RA_INFO_NUM_REC_GEN_INFOS_L	number of general infos
num_rec_dev_infos	RA_INFO_NUM_REC_DEV_INFOS_L	number of infos about a recording device
num_rec_ch_infos	RA_INFO_NUM_REC_CH_INFOS_L	number of infos about a channel
num_eval_infos	RA_INFO_NUM_EVAL_INFOS_L	number of infos about an evaluation
max_samplerate	RA_INFO_MAX_SAMPLERATE_D	maximum samplerate used in the recording
ch_xscale	RA_INFO_CH_XSCALE_D	position scale-factor for a channel
meas_size	RA_INFO_SIZE_L	size needed on disc of measurement in Bytes
meas_file	RA_INFO_FILES_CA	files belonging to measurement
meas_path	RA_INFO_PATH_C	full path of measurement incl. measurement name
meas_in_dir	RA_INFO_DIR_L	flag if measurement was saved in directory

Table 6-3. List of info-names for measurement-object (person) infos -- ra_obj_get_info

Info Name	Info Constant	Info Description
person_name	RA_INFO_OBJ_PERSON_NAME_C	Last name of the measurement object
person_forename	RA_INFO_OBJ_PERSON_FORENAME_C	First name of the measurement object
person_birthday	RA_INFO_OBJ_PERSON_BIRTHDAY_C	Birthday of the measurement object
person_height	RA_INFO_OBJ_PERSON_HEIGHT_L	Height of the measurement object
person_weight	RA_INFO_OBJ_PERSON_WEIGHT_L	Weight of the measurement object
person_street	RA_INFO_OBJ_PERSON_STREET_C	street of the address of the measurement object
person_city	RA_INFO_OBJ_PERSON_CITY_C	city of the address the measurement object
person_postalcode	RA_INFO_OBJ_PERSON_POSTALCODE_C	postalcode of the address the measurement object
person_country	RA_INFO_OBJ_PERSON_COUNTRY_C	country of the address the measurement object
person_address	RA_INFO_OBJ_PERSON_ADDRESS_C	address the measurement object (sometimes this field contains the whole address)
person_phone1	RA_INFO_OBJ_PERSON_PHONE1_C	phone-number of the measurement object
person_phone2	RA_INFO_OBJ_PERSON_PHONE2_C	phone-number 2 of the measurement object
person_fax	RA_INFO_OBJ_PERSON_FAX_C	fax-number of the measurement object
person_email	RA_INFO_OBJ_PERSON_EMAIL_C	email-address of the measurement object
person_website	RA_INFO_OBJ_PERSON_WEBSITE_C	website of the measurement object
person_comment	RA_INFO_OBJ_PERSON_COMMENT_C	comment about the measurement object

Table 6-4. List of info-names for measurement-object (patient) infos -- ra_obj_get_info

Info Name	Info Constant	Info Description
patient_id	RA_INFO_OBJ_PATIENT_ID_C	ID of the patient

Info Name	Info Constant	Info Description
patient_reason	RA_INFO_OBJ_PATIENT_REASON_C	reason of the recording
patient_diagnosis	RA_INFO_OBJ_PATIENT_DIAGNOSIS_C	diagnosis
patient_therapy	RA_INFO_OBJ_PATIENT_THERAPY_C	therapy
patient_medication	RA_INFO_OBJ_PATIENT_MEDICATION_C	medication of the patient
patient_hospital	RA_INFO_OBJ_PATIENT_HOSPITAL_C	Hospital where the recording was performed
patient_department	RA_INFO_OBJ_PATIENT_DEPARTMENT_C	Department where the recording was performed
patient_doctor	RA_INFO_OBJ_PATIENT_DOCTOR_C	Doctor who analyze the evaluation of the recording
patient_examiner	RA_INFO_OBJ_PATIENT_EXAMINER_C	Person who evaluate the recording
preg_num_fetus	RA_INFO_OBJ_PREG_WOMAN_NUM_FETUS_L	number fetus
preg_gestation_date	RA_INFO_OBJ_PREG_WOMAN_GESTATION_DATE_C	gestation date

Table 6-5. List of info-names for recording infos -- ra_rec_get_info

Info Name	Info Constant	Info Description
rec_type	RA_INFO_REC_GEN_TYPE_L	type of recording (see RA_REC_TYPE_*)
rec_name	RA_INFO_REC_GEN_NAME_C	name of recording
rec_desc	RA_INFO_REC_GEN_DESC_C	description of the recording
rec_comment	RA_INFO_REC_GEN_COMMENT_C	comment about the recording
rec_time	RA_INFO_REC_GEN_TIME_C	start time of recording (hh:mm:ss)
rec_date	RA_INFO_REC_GEN_DATE_C	start date of recording (dd.mm.yyyy)
rec_duration_sec	RA_INFO_REC_GEN_DURATION_SEC_L	duration of the recording in seconds
rec_number	RA_INFO_REC_GEN_NUMBER_L	number of recording (0: main recording)
rec_num_sub_rec	RA_INFO_REC_GEN_NUM_SUB_REC_L	number of sub-recordings

Info Name	Info Constant	Info Description
rec_num_devices	RA_INFO_REC_GEN_NUM_DEVICES_L	number of devices used for recording
rec_num_channel	RA_INFO_REC_GEN_NUM_CHANNEL_L	number of channels
rec_path	RA_INFO_REC_GEN_PATH_C	full path of recording incl. recording name
rec_is_directory	RA_INFO_REC_GEN_DIR_L	flag if recording was saved in directory
rec_mm_per_sec	RA_INFO_REC_GEN_MM_SEC_D	mm per second (hint for displaing signal)

Table 6-6. List of info-names for device infos -- ra_dev_get_info

Info Name	Info Constant	Info Description
dev_hw_name	RA_INFO_REC_DEV_HW_NAME_C	name of the recording device
dev_hw_manufacturer	RA_INFO_REC_DEV_HW_MANUFACTURER_C	manufacturer of the recording device
dev_hw_serial_no	RA_INFO_REC_DEV_HW_SERIAL_NO_C	serial number of the recording device
dev_hw_version	RA_INFO_REC_DEV_HW_VERSION_C	hardware-version of the recording device
dev_sw_name	RA_INFO_REC_DEV_SW_NAME_C	software-name of the recording device
dev_sw_manufacturer	RA_INFO_REC_DEV_SW_MANUFACTURER_C	manufacturer of the software of the recording device
dev_sw_serial_no	RA_INFO_REC_DEV_SW_SERIAL_NO_C	serial number of the software of the recording device
dev_sw_version	RA_INFO_REC_DEV_SW_VERSION_C	version of the software of the recording device

Table 6-7. List of info-names for channel infos -- ra_ch_get_info

Info Name	Info Constant	Info Description
ch_name	RA_INFO_REC_CH_NAME_C	name of the channel
ch_desc	RA_INFO_REC_CH_DESC_C	description of the channel
ch_num_samples	RA_INFO_REC_CH_NUM_SAMPLES_L	number of samples recorded in the channel
ch_samplerate	RA_INFO_REC_CH_SAMPLERATE_D	samplerate used in the channel
ch_num_bits	RA_INFO_REC_CH_NUM_BITS_L	number of bits used for one sample

Info Name	Info Constant	Info Description
ch_amp_res	RA_INFO_REC_CH_AMP_RESOLUTION_D	value for 'bit' representing one unit
ch_unit	RA_INFO_REC_CH_UNIT_C	unit of the channel
ch_center_value	RA_INFO_REC_CH_CENTER_VALUE_D	value [unit] signal is centered
ch_center_sample	RA_INFO_REC_CH_CENTER_SAMPLE_D	center value as sample-value
ch_min_adc	RA_INFO_REC_CH_MIN_ADC_D	minimum value in ADC amp-levels
ch_max_adc	RA_INFO_REC_CH_MAX_ADC_D	minimum value in ADC amp-levels
ch_min_unit	RA_INFO_REC_CH_MIN_UNIT_D	minimum value in units
ch_max_unit	RA_INFO_REC_CH_MAX_UNIT_D	maximum value in units
ch_mm_per_unit	RA_INFO_REC_CH_MM_UNIT_D	mm per unit (hint for displaing signal)
ch_type	RA_INFO_REC_CH_TYPE_L	type of channel (see above RA_CH_TYPE_*)

Table 6-8. List of info-names for evaluation infos -- ra_eval_get_info

Info Name	Info Constant	Info Description
eval_name	RA_INFO_EVAL_NAME_C	name of evaluation (if any)
eval_desc	RA_INFO_EVAL_DESC_C	comment of evaluation
eval_add_timestamp	RA_INFO_EVAL_ADD_TS_C	timestamp when eval was added
eval_modify_timestamp	RA_INFO_EVAL_MODIFY_TS_C	timestamp when eval was last modified
eval_user	RA_INFO_EVAL_USER_C	user who added eval
eval_host	RA_INFO_EVAL_HOST_C	machine on which eval was added
eval_program	RA_INFO_EVAL_PROG_C	program which added eval
eval_is_original	RA_INFO_EVAL_ORIGINAL_L	evaluation done in recording system
eval_is_default	RA_INFO_EVAL_DEFAULT_L	evaluation which should be used
class_id_ascii	RA_INFO_CLASS_ASCII_ID_C	ASCII id of event-class
class_name	RA_INFO_CLASS_NAME_C	name of event-class
class_desc	RA_INFO_CLASS_DESC_C	description of event-class

Info Name	Info Constant	Info Description
class_num_events	RA_INFO_CLASS_EV_NUM_L	number of events in event-class

Table 6-9. List of info-names for event-property infos -- ra_prop_get_info

Info Name	Info Constant	Info Description
prop_id_ascii	RA_INFO_PROP_ASCII_ID_C	ASCII id of event-property, must be unique
prop_value_type	RA_INFO_PROP_VALTYPE_L	type of the event-property values (short, long, double)
prop_name	RA_INFO_PROP_NAME_C	name of event-property, must be unique
prop_desc	RA_INFO_PROP_DESC_C	description of event property
prop_unit	RA_INFO_PROP_UNIT_C	unit of the property
prop_has_minmax	RA_INFO_PROP_HAS_MINMAX_L	flag if property has min-/max-values
prop_min	RA_INFO_PROP_MIN_D	normal min-value of the property
prop_max	RA_INFO_PROP_MAX_D	normal max-value of the property
prop_ignore_value	RA_INFO_PROP_IGNORE_VALUE_D	ignore value (or undef if no one is available)

Table 6-10. List of info-names for event-summary infos -- not yet implemented

Info Name	Info Constant	Info Description
sum_id_ascii	RA_INFO_SUM_ASCII_ID_C	ASCII id of event-summary, must be unique
sum_name	RA_INFO_SUM_NAME_C	name of event-summary
sum_desc	RA_INFO_SUM_DESC_C	description of event-summary
sum_num_dim	RA_INFO_SUM_NUM_DIM_L	
sum_dim_unit	RA_INFO_SUM_DIM_UNIT_C	
sum_dim_name	RA_INFO_SUM_DIM_NAME_C	
sum_num_ch	RA_INFO_SUM_NUM_CH_L	
sum_ch_num	RA_INFO_SUM_CH_NUM_L	
sum_ch_fiducial	RA_INFO_SUM_CH_FIDUCIAL_L	

Table 6-11. List of info-names for plugin infos -- ra_plugin_get_info

Info Name	Info Constant	Info Description
pl_name	RA_INFO_PL_NAME_C	name of plugin, used for identification
pl_desc	RA_INFO_PL_DESC_C	description of plugin
pl_file	RA_INFO_PL_FILE_C	filename with path of the plugin
pl_use_it	RA_INFO_PL_USE_IT_L	flag if plugin shoul be used or not
pl_type	RA_INFO_PL_TYPE_L	type of plugin
pl_version	RA_INFO_PL_VERSION_C	version of plugin
pl_build_ts	RA_INFO_PL_BUILD_TS_C	build timestamp of plugin
pl_license	RA_INFO_PL_LICENSE_L	license used for plugin
pl_num_opt	RA_INFO_PL_NUM_OPTIONS_L	number of options in the plugin
pl_num_results	RA_INFO_PL_NUM_RESULTS_L	number of values returned from plugin

Table 6-12. List of info-names for plugin-option infos -- ra_plugin_get_option_info

Info Name	Info Constant	Info Description
pl_opt_name	RA_INFO_PL_OPT_NAME_C	short name of option
pl_opt_desc	RA_INFO_PL_OPT_DESC_C	description of option
pl_opt_type	RA_INFO_PL_OPT_TYPE_L	type of option

Table 6-13. List of info-names for plugin-results infos -- ra_proc_get_result_info

Info Name	Info Constant	Info Description
proc_num_res_sets	RA_INFO_PROC_NUM_RES_SETS_L	number of available result sets
proc_num_res	RA_INFO_PROC_NUM_RES_L	number of available results per set
res_name	RA_INFO_PL_RES_NAME_C	short name of result (can be used as table header)
res_desc	RA_INFO_PL_RES_DESC_C	description of result
res_default	RA_INFO_PL_RES_DEFAULT_L	flag if result belongs to default-values (=1)
session_name	RA_INFO_SES_NAME_C	name of the session
session_desc	RA_INFO_SES_DESC_C	a description of the session

Chapter 7. Signals for Inter Plugin Communication

No special intro text.

select-event

Name

`select-event` — select evens with a specific value

Synopsis

`select-event` `(`*`event-name`*`, `*`event-value`*`);`

Arguments

`event-name - char-pointer`

name of an event-property

`event-value - double`

value of an event-property

Description

This signal will be emitted if a set of events (’event-name’) with a specific value (’event-value’) should be selected.

highlight-event

Name

`highlight-event` — highlight an event

Synopsis

`highlight-event` `(`*`event-set-name`*`, `*`event-number`*`);`

Arguments

`event-set-name - char-pointer`

name of an event-set

`event-number - long`

number of a single event

Description

This signal can be emitted from a plugin if an event was selected and this event should be emphasised in other view-plugins. The selected event has the number ’event-number’ and is in the event-set ’event-set-name’.

eval-change

Name

`eval-change` — evaluation has changed

Synopsis

`eval-change` `(`*`none`*`);`

Arguments

-

no argument

Description

This signal will be emitted if the default-evaluation has changed. The receiving plugins must re-read the evaluation data and correct their handling of the evaluation data (e.g. displaying a x/y-plot) if necessary.

start-event-change

Name

`start-event-change` — some event-change signals are coming

Synopsis

`start-event-change` `(`*`none`*`);`

Arguments

`-`

no argument

Description

This signal will be emitted if more than one event-change signal is coming. The plugin can decide if it will response to every single change or collect all changes and response to the "end-event-change" signal.

end-event-change

Name

`end-event-change` — close an start-event-change

Synopsis

`end-event-change` `(`*`none`*`);`

Arguments

`-`

no argument

Description

This signal 'closes' an "start-event-change". If the plugin collected the event-changes, now it can handle the changes.

event-change

Name

`event-change` — event-value has changed

Synopsis

`event-change` `(`*`event-name`*`, `*`event-number`*`);`

Arguments

`event-name - char-pointer`

name of an event-property

`event-number - long`

number of a single event

Description

This signal will be emitted if the value of an event-property has changed. The receiving plugins must check if the event-property is handled by the plugin and can ignore it, if it is not handled. If it is handled, the plugin can update whatever is needed (e.g. replot event-property values). If 'event-number' is -1, the whole event-property should be re-read.

region-change

Name

`region-change` —

Synopsis

`region-change` `(`*`rec-pos`*`, `*`rec-pos`*`);`

Arguments

`rec-pos - long`

position in recording stream

`rec-pos - long`

position in recording stream

Description

This signal will be emitted if something changed between two positions.

add-event

Name

`add-event` — an event was added

Synopsis

`add-event` `(`*`event-set-name`*`, `*`event-number`*`);`

Arguments

`event-set-name - char-pointer`

name of an event-set

`event-number - long`

number of a single event

Description

This signal will be emitted if a new event (with event-number ’event-number’) was added to event-set ’event-set-name’.

del-event-begin

Name

`del-event-begin` — start of an event-deletion

Synopsis

`del-event-begin` `(`*`event-set-name`*`, `*`event-number`*`);`

Arguments

event-set-name - char-pointer

name of an event-set

event-number - long

number of a single event

Description

This signal will be emitted before an event will be deleted. This enables plugins to get infos from the event before deletion. E.g. for templates it is necessary to know th template-number used by an event. The deleted event is event number ’event-number’ from the event-set ’event-set-name’.

del-event-end

Name

`del-event-end` — end of an event-deletion

Synopsis

```
del-event-end (event-set-name, event-number);
```

Arguments

event-set-name - char-pointer

name of an event-set

event-number - long

number of a single event

Description

This signal will be emitted after an event was deleted. The deleted event is event number 'event-number' from the event-set 'event-set-name'.

template-delete

Name

`template-delete` — a template was deleted

Synopsis

`template-delete` `(`*`event-set-name`*`, `*`template-number`*`);`

Arguments

`event-set-name - char-pointer`

name of an event-set

`template-number - long`

number of a template

Description

This signal will be emitted after a template was deleted. The deleted template is template number 'template-number' from the event-set 'event-set-name'.

template-add

Name

`template-add` — a template was added

Synopsis

`template-add` `(`*`event-set-name`*`, `*`template-number`*`);`

Arguments

`event-set-name - char-pointer`

name of an event-set

`template-number - long`

number of a template

Description

This signal will be emitted if a template was added. The added template is template number 'template-number' from the event-set 'event-set-name'.

pos-change

Name

`pos-change` — set view to a new position

Synopsis

`pos-change` `(`*`rec-pos`*`, `*`max-x-scale`*`);`

Arguments

`rec-pos - long`

position in recording stream

`max-x-scale - double`

scaling factor for x-axis for channel with highest samplerate

Description

This signal will be emitted, if a view-plugin changed the position in the recording stream which is displayed. The new position is given by 'rec-pos' and is in sampleunits of the recording. View-plugins MUST handle this signal. (TODO: think about how to handle different recordings?)

gui-event-start

Name

`gui-event-start` — start of signal activated by GUI-event

Synopsis

```
gui-event-start (none);
```

Arguments

`-`

no argument

Description

This signal will indicate that the next signal(s) emitted is(are) released by a GUI-event (e.g. mouse click).

gui-event-end

Name

`gui-event-end` — end of signal activated by GUI-event

Synopsis

```
gui-event-end (none);
```

Arguments

`-`

no argument

Description

Indicating that the GUI-event is finished. (see ’gui-event-start’).

x-resolution

Name

`x-resolution` — set new resolution of x-axis

Synopsis

```
x-resolution (x-res, gui-parent);
```

Arguments

x-res - double

resolution of the x-axis

gui-parent - long

id (e.g. the pointer) of the parent gui-element

Description

This signal will be emitted if the resolution of the x-axis of a view has changed. The new reslution is given by 'x-res'. Additionally the parent of the view is given by 'gui-parent'. Normally the view only change his x-resolution if the parent of the receiving view is the same as the sending view (sync. of splitted views).

y-resolution

Name

`y-resolution` — set new mm/unit of a channel

Synopsis

```
y-resolution (mm-per-unit, channel);
```

Arguments

mm-per-unit - double

mm/unit a channel

channel - long

channel-number of root-measurement

Description

This signal will be emitted if the mm/unit of a channel has changed. The new y-resolution is given by ’mm-per-unit’, the channel by ’channel’.

Chapter 8. libRASCH Plugins

No special intro text.

8.1. Access Plugins

ART

Name

`ART` — Handle signals recorded for ART-Study

Description

The plugin handles signals which are recorded for the ART-Study. To access the recorded data, the plugins for portilab and portapres are used.

Version

0.3.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

adi-ascii

Name

`adi-ascii` — Handle signals recorded with Powerlab and saved as ASCII files

Description

The plugin provides access to signals recorded with Powerlab from ADInstruments and exported as an ASCII file.

Version

0.1.1

Author(s)

Raphael Schneider (librasch@gmail.com)

cfs

Name

`cfs` — Handle signals saved using the CED Filing System (CFS)

Description

The plugin provides access to recordings saved in the CED Filing System (CFS) format. At the moment only equalspaced channels are supported.

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

ctg-rasch

Name

`ctg-rasch` — Handle cardiotocogram signals exported from GMT system

Description

The plugin provides access to CTG's exported from the GMT system. The exported raw data (three files: fetal heart rate child 1, fetal heart rate child 2 and uterine contraction) is packed in a directory with the extension '.ctg'. Additionally some information about the mother and the recording is stored in the file 'ctg_info.txt'.

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

ctg-sonicaid

Name

`ctg-sonicaid` — Handle cardiotocogram signals saved with the Oxford FetalCare system

Description

The plugin provides access to CTG's handled with the Oxford FetalCare system. The information about the Mother and the recording date and time is stored in a Access database-file. Therefore a text file similar to the one in the ctg-rasch plugin will be used to handle this information (the file is called 'ctg_info.txt').

Version

0.1.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

dadisp

Name

`dadisp` — Handle signals saved a DaDisp signal file format

Description

The plugin provides access to recordings saved in a DaDisp signal file format. Because I have not used the program DaDisp but have only some files, this plugin supports only this format (files with the extension 'dsp' and a ASCII header at the beginning).

Version

0.1.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

dasylab

Name

`dasylab` — Handle signals recorded using DasyLab

Description

The plugin provides access to recordings saved in the format used by the DasyLab-Systems.

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

edf/edf+

Name

`edf/edf+` — Handle signals recorded using European Data Format(+)

Description

The plugin provides access to recordings saved in the European Data Format (EDF) and the EDF+ format.

Remarks

If the EDF+ file is a non-contiguous signal, than each data-block is handled as a single session. (Recommendation from Jesus Olivan.)

Version

0.4.0

Author(s)

Raphael Schneider (librasch@gmail.com)

ideeq-ascii

Name

`ideeq-ascii` — Handle signals recorded with IDEEQ and saved as ASCII files

Description

The plugin provides access to signals recorded with IDEEQ and exported as an ASCII file.

Version

0.1.1

Author(s)

Raphael Schneider (librasch@gmail.com)

ishne-holter

Name

`ishne-holter` — Handle signals saved using ISHNE standard output for Holter ECG

Description

The plugin provides access to recordings saved in the ’ISHNE standard output for Holter ECG’ format.

Version

0.3.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

mit-bih

Name

`mit-bih` — Handle signals recorded using the MIT/BIH Data Format

Description

The plugin provides access to recordings saved in the format used for the MIT/BIH database(s) and the databases of provides by PhysioNet (www.physionet.org).

Version

0.5.1

Author(s)

Raphael Schneider (librasch@gmail.com)

mortara-sparc

Name

`mortara-sparc` — Handle ECGs stored in the (Mortara) SPARC format

Description

The plugin provides access to ECGs stored in the (Mortara) SPARC format.

Version

0.1.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

portapres

Name

`portapres` — Handle signals recorded with Portapres-System

Description

The plugin provides access to blood-pressure data recorded with the Portapres system.

Remarks

Up to now only a small subset of the binary-file can be read. It is planned to add support to access the data exported with BeatScope.

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

poly5/tms32

Name

`poly5/tms32` — Handle signals recorded using TMS ADC system and PortiLab

Description

The plugin provides access to data recorded with the TMS ADC-system and using the PortiLab software.

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

read-rri

Name

`read-rri` — Handle RR interval-files (nearly the final design)

Description

The plugin provides access to RR interval files. It now supports the first RR-interval format specific for libRASCH.

Version

0.4.1

Author(s)

Raphael Schneider (librasch@gmail.com)

8.2. Process Plugins

ap-morphology

Name

`ap-morphology` — get systolic and diastolyc values and positions of aterial pressure waves

Description

The plugin get systolic and diastolic values and positions from bloodpressure waves. Additionally it tries to find calibration-intervals (eg in the Portapres system it is possible to enable calibration during the measurement).

Results

Table 8-1. List of results of plugin ap-morphology

Name	Description	type
EV_ID	event-id’s	long array
CH	channel-number the tuple of data belongs to	long array
SYST	systolic blood pressures	double array
DIAS	diastolic blood pressures	double array
MEAN	mean blood pressures	double array
SYST_POS	positions of systolic measurements	long array

Name	Description	type
DIAS_POS	positions of diastolic measurements	long array
FLAGS	flags	long array
IBI	inter beat intervals	double array
CALIB_BEGIN	begin of the calibration sequences [su]	long array
CALIB_END	end of the calibration sequences [su]	long array
CALIB_CH	channel-number the calibration sequence belongs to	long array
CALIB_INFO	infos about the calibration sequences	long array
DPDT_MIN	dp/dt minimum	double array
DPDT_MAX	dp/dt maximum	double array
DPDT_MIN_POS	positions of dp/dt min	long array
DPDT_MAX_POS	positions of dp/dt max	long array

Version

0.4.0

Author(s)

Raphael Schneider (librasch@gmail.com)

calibration

Name

`calibration` — measure calibration signals in a recording

Description

The plugin tries to detect and measure a calibration signal in a recording channel.

Options

Table 8-1. List of options for plugin calibration

Name	Description	type
rh	recording handle	long
ch	channel of the recording used to measure the calibration signal	long
start_pos	signal-position to start search for calibration signal [su]	long
end_pos	signal-position to stop search for calibration signal [su]	long
use_data	use the data given in ’data’	
num_data	number of the data given in ’data’	long
data	perform calibration meausure using this data	double array
samplerate	samplerate used for the values in ’data’	double
type	type of calibration signal (see CALIB_TYPE_* in ra_calibration.h)	long
difference_lag	gap between samples used for difference signal [su]	long
min_calib_amp	percentage of maximum difference used as minimal calibration amplitude	double
segment_length	grid-width used to search for continous calibration signals [seconds]	double
min_level_length	minimum duration staying on the same level [seconds]	double
calib_cycle_length	length of one calibration cycle [seconds]	double
get_cycle_length	flag if the calibration cycle length should be measured	
min_cycles	required minimum number of contiguous calibration cycles	long

Results

Table 8-2. List of results of plugin calibration

Name	Description	type
calib_height	calibration value	double
calib_high_value	high-level value of the calibration signal	double
calib_low_value	low-level value of the calibration signal	double
calib_width	width of the calibration signal (measured using ’calib_data’)	double
calib_type	type of the calibration signal	long
calib_accuracy	accurcy of the calibration height [percent]	double
calib_out_of_range	flag if calibration cycles reached upper or lower value range	long
calib_data	data used to measure calibration (all valid segments combined)	double array
calib_cycle_pos	start- and end-position of valid calibration cycles	long array

Version

0.3.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

dawes-redman

Name

`dawes-redman` — calculate FHR variations using the Dawes/Redman criteria

Description

The plugin calculates the fetal heart rate variations using the Dawes/Redman criteria. The estimation of the baseline of the fetal heart rate is done according the descriptions in the various papers from Dawes and Redman. For the digital filter, a Butterworth filter with order 4 and corner-frequency of 0.0017Hz (approx. 0.1 min-1)is used (using the 3.75sec epoch-values results in a samplerate of approx. 0.2667Hz). The filter was designed on the website of Tony Fisher, using the mkfilter cgi-script. This can be found at http://www-users.cs.york.ac.uk/~fisher/mkfilter/ .

Results

Table 8-1. List of results of plugin dawes-redman

Name	Description	type
FHR_EPOCHS	epochs of the fetal heart rate (averages over 3.75 seconds)	double array
FHR_BASELINE	baseline of the fetal heart rate	double array
SIGNAL_LOST	signal lost in percent	double
NUM_VALID_EPOCHS	number of valid epochs	long
BASAL_FHR	basal heart rate of the fetus	double
ACCEL_10	number of accelerations > 10bpm and <= 15bpm	long
POS_ACCEL_10	positions of the accelerations between 10 and 15bpm	long array
ACCEL_15	number of accelerations > 15bpm	long
POS_ACCEL_15	positions of the accelerations > 15bpm	long array
LOST_BEATS_20	decelerations < 20 lost beats	long
POS_LOST_BEATS_20	positions of the decelerations < 20	long array
LOST_BEATS_21_100	decelerations between 20 and 100 lost beats	long
POS_LOST_BEATS_21_100	postions of the decelerations between 20 and 100	long array
LOST_BEATS_101	decelerations above 100 lost beats	long
POS_LOST_BEATS_101	postions of the decelerations above 100	long array
MINUTE_RANGE	minute range for each minute in bpm	double array
MINUTE_RANGE_MS	miniute range for each minute in msec	double array
HIGH_VARIATIONS	number of minutes with high variations	long
MIN_HIGH_VARIATIONS	the minutes with the high variations	long array

Name	Description	type
LOW_VARIATIONS	number of minutes with low variations	long
MIN_LOW_VARIATIONS	the minutes with low variations	long array
LONG_TERM_VARIATIONS_MS	long term variations for all minutes in msec	double
LONG_TERM_VARIATIONS_BPM	long term variations for all minutes in bpm	double
LONG_TERM_VARIATIONS_HIGH_MS	long term variations for minutes with high variations in msec	double
LONG_TERM_VARIATIONS_HIGH_BPM	long term variations for minutes with high variations in bpm	double
SHORT_TERM_VARIATIONS	short term variations	double

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

detect-ctg

Name

`detect-ctg` — perform detections in cardiotocograms

Description

The plugin performs detections of events in a cardiotocogram. At the moment only the position of the maximal uterine contraction will be searched.

Remarks

In the future, the plugin will be combined with the detect-simple plugin. For this step, the detect-simple plugin will be extended with options and the possibility to return the detection results not only by saving them in the evaluation file. But to get a faster result, the CTG-specific code will be first implemented here.

Options

Table 8-1. List of options for plugin detect-ctg

Name	Description	type
num_ch	number of channels set in ’ch’	long
ch	list of channels used for the event detection	long array
save_in_eval	flag if results should be saved in an evaluation	
eh	eval-handle used to store the results	
clh	event-class-handle used to store the results when region was selected	

Results

Table 8-2. List of results of plugin detect-ctg

Name	Description	type
num	number of found events	long
pos	positions of the found events in sampleunits	long array

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

detect-ecg

Name

`detect-ecg` — performs ECG beat-detection and morphology measures

Description

The file provides the following digital signal processing (DSP) routines: - FIR filter creation - filtfilt function to filter a signal without a phase-distortion - interpolation (increase in samplerate) of a signal

Options

Table 8-1. List of options for plugin detect-ecg

Name	Description	type
num_ch	number of channels set in 'ch'	long
ch	list of channels used for the event detection	long array
combine_beats	flag if nearby beats (distance below 'min_beat_distance') should be combined	
min_beat_distance	minimum allowed distance between beats [seconds]	double
save_in_eval	flag if results should be saved in an evaluation	
eh	eval-handle used to store the results	
clh	event-class-handle used to store the results when region was selected	
use_region	search beats in a given area	
region_start	start position of the search area	long
region_end	end position of the search area	long
region_start_is_beatpos	'region_start' is the beat position	
num_events	number of events set in 'events' option	long

Name	Description	type
events	heartbeat events for which the wave-boundary detection should be re-run	long array
filter_powerline_noise	flag if a power-line noise filter should be applied	
filter_baseline_wander	flag if a baseline-wander filter should be applied	
check_for_calibration	flag if for a calibration signal should be searched	
check_for_calibration	flag if for a calibration signal should be searched	
check_for_calibration	flag if for a calibration signal should be searched	
force_p_type		long
force_qrs_type		long
force_t_type		long
check_wave_at_cursor		
edit_cursor_pos		long
edit_cursor_pos		long
thresh_p		double
thresh_qrs		double
thresh_t		double

Results

Table 8-2. List of results of plugin detect-ecg

Name	Description	type
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		

Name	Description	type
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		
"ch_t_type", gettext_noop(""), RA_VALUE_TYPE_LONG_ARRAY		

Version

0.1.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

ecg

Name

`ecg` — perform ecg specific actions after reading original evaluation or doing detect using libRASCH

Description

The plugin performs a finer classification of the qrs-complexes of an ecg, using from a previous evaluation only the information of the general annotation of a beat (coming from atrium, from ventricle, paced beat or artifact). Using this plugin allows to have in libRASCH a consistent sub-classification (eg premature normal beat) independent of the original evaluation system used.

Results

Table 8-1. List of results of plugin ecg

Name	Description	type
QRS_TEMPORAL	temporal setting of beat	
QRS_ANNOT	annotation of QRS complex	long array
ECG_FLAGS	ecg flags	long array
RRI	RR interval	double array
RRI_ANNOT	annoation of RR interval	long array
RRI_REFVALUE	reference rri representing the current heart-rate	double array
RRI_NUM_REFVALUE	number of rri's used for calculation of reference value	
IDX_RRI	event indeces the RR intervals belongs to	long array
p_width		double array
qrs_width		double array
t_width		double array
pq		double array
qt		double array
qtc		double array
qta		double array
resp_phase		double array
resp_ch		long array
NUM_ALL	number of all QRS complexes	long

Name	Description	type
NUM_UNKNOWN	number of un-classified QRS complexes	long
IDX_UNKNOWN	event indeces of un-classified QRS complexes	long array
NUM_SINUS	number of sinus beats	long
IDX_SINUS	event indeces of sinus beats	long array
NUM_NORMAL	number of normal sinus-beats	long
IDX_NORMAL	event indeces of normal sinus-beats	long array
NUM_SVPC	number of SVPCs	long
IDX_SVPC	event indeces of SVPCs	long array
NUM_SINUS_VPC	number of premature sinus-beats which following beats come late (perhaps VENT?)	long
IDX_SINUS_VPC	event indeces of premature sinus-beats which following beats come late (perhaps VENT?)	long array
NUM_PAUSE	number of pause	long
IDX_PAUSE	event indeces of pause	long array
NUM_VENT	number of VPCs	long
IDX_VENT	event indeces of VPCs	long array
NUM_VENT_SINGLE	number of single VPCs	long
IDX_VENT_SINGLE	event indeces of single VPCs	long array
NUM_VENT_SINGLE_PREM	number of VPCs which are premature	long
IDX_VENT_SINGLE_PREM	event indeces of VPCs which are premature	long array
NUM_VENT_SINGLE_PREM_COMP	number of VPCs which are premature and have a compensatory pause	long
IDX_VENT_SINGLE_PREM_COMP	event indeces of VPCs which are premature and have a compensatory pause	long array
NUM_INTERP	number of interpolated VPCs	long
IDX_INTERP	event indeces of interpolated VPCs	long array
NUM_ESACPE	number of escape beats	long
IDX_ESCAPE	event indeces of escape beats	long array
NUM_PACED	number of paced beats	long
IDX_PACED	event indeces of paced beats	long array

Name	**Description**	**type**
NUM_ARTIFACT	number of artifacts	long
IDX_ARTIFACT	event indeces of artifacts	long array
IDX_ARTIFACT	event indeces of artifacts	long array
IDX_ARTIFACT	event indeces of artifacts	long array
NUM_ARTIFACT_TYPE2	number of automatic detected artifacts: perhaps detected T-wave	long
IDX_ARTIFACT_TYPE2	event indeces of automatic detected artifacts: perhaps detected T-wave	long array
NUM_ARTIFACT_TYPE3	number of automatic detected artifacts: perhaps overlooked beat after a VPC	long
IDX_ARTIFACT_TYPE3	event indeces of automatic detected artifacts: perhaps overlooked beat after a VPC	long array
MIN_HR	min. heart rate	double
IDX_MIN_HR	start event index of min. heart rate	long
MAX_HR	max. heart rate	double
IDX_MAX_HR	start event index of max. heart rate	long
MEAN_HR	mean heart rate	double
NUM_BRADY	number of bradycardia	long
IDX_BRADY	start event indeces of the bradicardia	long array
LEN_BRADY	length (#rri's) of the bradicardia	long array
HR_BRADY	heart rate of the bradicardia	double array
NUM_TACHY	number of tachycardia	long
IDX_TACHY	start event indeces of the tachycardia	long array
LEN_TACHY	length (#rri's) of the tachycardia	long array
HR_TACHY	heart rate of the tachycardia	double array
NUM_SSALVO	number of supraventricular salvos	long
IDX_SSALVO	start event indeces of supraventricular salvos	long array
LEN_SSALVO	length of supraventricular salvos	long
HR_SSALVO	heart rate of supraventricular salvos	double

Name	Description	type
NUM_SVT	number of supraventricular tachycardia	long
IDX_SVT	start event indeces of supraventricular tachycardia	long array
LEN_SVT	length of supraventricular tachycardia	long
HR_SVT	heart rate of supraventricular tachycardia	double
NUM_SSALVO_SVT	number of supraventricular salvos/SVT's	long
IDX_SSALVO_SVT	start event indeces of supraventricular salvos/SVT's	long array
LEN_SSALVO_SVT	length of supraventricular salvos/SVT's	long
HR_SSALVO_SVT	heart rate of supraventricular salvos/SVT's	double
NUM_IVR	number of ideoventricular rhythms	long
IDX_IVR	start event indeces of ideoventricular rhythms	long array
LEN_IVR	length (#rri's) of ideoventricular rhythms	long array
HR_IVR	heart rate of ideoventricular rhythms	double array
NUM_COUPLET	number of couplets	long
IDX_COUPLET	event indeces of couplets	long array
NUM_BIGEMINY	number of bigeminy	long
IDX_BIGEMINY	start event indeces of the bigeminy	long array
LEN_BIGEMINY	length (#rri's) of the bigeminy	long array
NUM_TRIGEMINY	number of trigeminy	long
IDX_TRIGEMINY	start event indeces of the trigeminy	long array
LEN_TRIGEMINY	length (#rri's) of the trigeminy	long array
NUM_SALVO	number of salvos	long
IDX_SALVO	start event indeces of the salvos	long array
LEN_SALVO	length (#rri's) of the salvos	long
HR_SALVO	heart rate of the salvos	double
NUM_VT	number of VTs	long
IDX_VT	start event indeces of VTs	long array
LEN_VT	length (#rri's) of the VTs	long

Name	Description	type
HR_VT	heart rate of the VTs	double
NUM_SALVO_VT	number of salvos/VTs	long
IDX_SALVO_VT	start event indeces of the salvos/VTs	long array
LEN_SALVO_VT	length (#rri's) of the salvos/VTs	long
HR_SALVO_VT	heart rate of the salvos/VTs	double

Version

0.5.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

fiducial-point

Name

`fiducial-point` — finds fiducial points

Description

The plugin finds the fiducial-points of the QRS-complexes in an ecg.

Options

Table 8-1. List of options for plugin fiducial-point

Name	Description	type
use_class	align events stored in option 'clh'	

Name	Description	type
clh	handle to the event-class holding the events which should be processed	
pos_prop	ASCII-id of the event-property holding the positions (optional)	char *
save_in_class	flag if the fiducial-points should be saved in the event-class	
num_data	number of event-id's/sample-positions	long
data	event-id's/sample-positions	long array
data_is_pos	flag if values in 'data' are event-id's (=0) or sample positions (=1)	
ch	channel where fiducial point should be searched	long
win_len	area (+-'win_len') where the fiducial-point will be searched [sec]	double
corr_len	length of correlation [sec]	double

Results

Table 8-2. List of results of plugin fiducial-point

Name	Description	type
pos	positions after the alignment	long array

Version

0.3.0

Author(s)

Raphael Schneider (librasch@gmail.com)

freq-analysis

Name

`freq-analysis` — perform frequency analysis

Description

Options

Table 8-1. List of options for plugin freq-analysis

Name	Description	type
use_events	!=0: use events, ==0: use signal raw data	
clh	event class of the events	
prop_value	use these event-property for frequency analysis	char *
use_event_pos	flag if positions should be used from the events (=1) or from 'prop_pos' (=0)	
use_event_pos	flag if positions should be used from the events (=1) or from 'prop_pos' (=0)	
use_event_pos	flag if positions should be used from the events (=1) or from 'prop_pos' (=0)	
num_events	use the events listed in events	long
events	list of events to use	long array
num_ignore_events	number of events which should be ignored	long
ignore_events	list of events to ignore	long array
rh	recording handle	
ch	channel which should be used	long
ch	channel which should be used	long
ch	channel which should be used	long
use_start_end_pos	use raw-data/events between start_pos and end_pos	
start_pos	start-pos in sample-units	long
end_pos	end-pos in sample-units	long

Name	Description	type
num_values	number of values given in 'value'	long
values	perform frequency analysis using this data	double array
pos	positions of the values given in 'values'	double array
pos_samplerate	samplerate of the position values	double
samplerate	samplerate to use when freq.analysis should be done time based	double
smooth_data	flag if input-data should be smoothed	
smooth_method	at the moment only a boxcar-filter is available	char *
smooth_width	width of the smooth-filter	double
window	window function used	char *
remove_mean	remove the mean value before the frequency analysis	
remove_trend	remove the trend (first order) before the frequency analysis	
method	method used for the frequency analysis	char *
num_freq	number of frequency components	long
smooth_spec	flag if spectrum should be smoothed	
save_in_eval	save results in the evaluation file (not supported yet)	

Results

Table 8-2. List of results of plugin freq-analysis

Name	Description	type
FREQ_AXIS	frequency values (the x-axis)	double array
PSD	Power Spectrum Density	double array
REAL_PART	real part of the frequency spectrum	double array
IMG_PART	imaginary part of the frequency spectrum	double array

Version

0.3.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

hrv

Name

`hrv` — calculate heart rate variability (HRV) parameters

Description

The plugin calculates the heart rate variability (HRV) parameters. In the time domain the parameters, which are recommended from the Task Force for HRV, are calculated. The parameters from the frequency domain are calculated on the power-spectrum of the complete tachogram using FFT.

Results

Table 8-1. List of results of plugin hrv

Name	Description	type
SDNN	standard deviation of normal-to-normal intervals	double
HRVI	HRV-Index	double
SDANN	standard deviation of averaged normal-to-normal intervals	double
rmssd	root mean of squared sucsessive differences	double
pNN50		double
TP	total power	double
ULF	ultra low frequency power	double
VLF	very low frequency power of short-term recordings	double

Name	**Description**	**type**
LF	low frequency power	double
LF_NORM	normalised low frequency power	double
HF	high freuqency power	double
HF_NORM	normalised high frequency power	double
LF_HF_RATIO	LF/HF ratio	double
POWER_LAW	power law behavior	double
TACHO_INDEX	Event numbers used for HRV calculations	long array
USER_BAND	frequency power in a user-selected frequency band	double
SD1	SD1 of the Poincare Plot	double
SD2	SD2 of the Poincare Plot	double
DFA	overall DFA Alpha	double
DFA_OFFSET	offset of the overall DFA Alpha slope	double
DFA1	DFA Alpha1	double
DFA1_OFFSET	offset of the DFA Alpha-1 slope	double
DFA2	DFA Alpha2	double
DFA2_OFFSET	offset of the DFA Alpha-2 slope	double
DFA_USER	DFA Alpha of user-defined range	double
DFA_USER_OFFSET	offset of the user-defined DFA Alpha slope	double
DFA_X	x-axis for DFA plot	double array
DFA_Y	y-axis for DFA plot	double array

Version

0.4.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

template

Name

`template` — combine events in templates

Description

The plugin group events which have similar raw-data (e.g. QRS-complexes which have a similar morphology).

Options

Table 8-1. List of options for plugin template

Name	Description	type
rh	recording handle of the data which should be processed	
sh	event summary handle of already available template event-class	
use_class	align events stored in option 'clh'	
clh	handle to the event-class holding the events which should be processed	
pos_prop	ASCII-id of the event-property holding the positions (optional)	char *
pos_ch	channel-number 'pos_prop' belongs to (only needed when 'pos_prop' is used)	long
save_in_class	flag if the fiducial-points should be saved in the event-class	
num_data	number of event-id's/sample-positions	long
data	event-id's/sample-positions	long array
data_is_pos	flag if values in 'data' are event-id's (=0) or sample positions (=1)	
templ_name	name of event-type which holds template numbers	char *
templ_corr	name of event-type which holds template correlation	char *

Name	Description	type
corr_win_before	correlation window size before the reference position [sec]	double
corr_win_after	correlation window size after the reference position [sec]	double

Version

0.4.0

Author(s)

Raphael Schneider (librasch@gmail.com)

8.3. GUI/View Plugins

ch-select-dlg

Name

`ch-select-dlg` — dialog for selecting channels which will be used for processing

Description

Options

Table 8-1. List of options for plugin ch-select-dlg

Name	Description	type
num_ch	number of channels given in ’ch_type_filter’	long
ch_type_filter	list of channel types which have to be shown	long array

Results

Table 8-2. List of results of plugin ch-select-dlg

Name	Description	type
CH	list of selected channels	long array

Version

0.3.0

Author(s)

eval-dlg

Name

`eval-dlg` — a dialog showing all evaluations for a measurement

Description

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

plugin-info-dlg

Name

`plugin-info-dlg` — a dialog showing the list of all loaded plugins

Description

Version

0.3.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

sig-sel-dlg-mfc

Name

`sig-sel-dlg-mfc` — dialog to choose signales which can be handled with libRASCH (MFC version)

Description

The plugin provides a dialog which allows to choose a measurement. The dialog shows all supported measurements in a directory, which can be selected.

Options

Table 8-1. List of options for plugin sig-sel-dlg-mfc

Name	Description	type
initial_path	open dialog showing this path	char *

Results

Table 8-2. List of results of plugin sig-sel-dlg-mfc

Name	Description	type
SEL_FILE	selected file (incl. path)	char *

Version

0.3.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

simple-calc-dlg

Name

`simple-calc-dlg` — dialog to perform calculations using process plugins

Description

The plugin provides a dialog to perform calculations done process plugins. Only process plugins which return values (and not only store the results in the evaluation file like the beat-detect plugin) can be used (and only these can be selected).

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

annot-comment-view

Name

`annot-comment-view` — view listing annotations/comments

Description

The plugin list the annotations/comments saved in an evaluation.

Options

Table 8-1. List of options for plugin annot-comment-view

Name	Description	type
eh	eval-handle	

Version

0.1.0

Author(s)

Raphael Schneider (librasch@gmail.com)

cont-ap-view

Name

`cont-ap-view` — view for continues arterial pressure recordings

Description

Version

0.2.0

Author(s)

ctg-view

Name

`ctg-view` — view for cardiotocograms

Description

The plugin allows to view ecg-stripes. The plugin use the ts-view plugin for showing the raw-data.

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

ecg-view

Name

`ecg-view` — view for ecg's

Description

The plugin allows to view ecg-stripes. The plugin use the ts-view plugin for showing the raw-data.

Version

0.2.0

Author(s)

Raphael Schneider (librasch@gmail.com)

ev-summary-view

Name

`ev-summary-view` — view for event summaries

Description

The plugin shows event-summaries of an evaluation.

Options

Table 8-1. List of options for plugin ev-summary-view

Name	Description	type
sh	event-summary-handle	

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

plot-view

Name

`plot-view` — plot evaluation data

Description

The plugin plot evaluation data in a window and allows to control which data and in which format the data is plotted.

Version

0.2.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

rasch-view

Name

`rasch-view` — general view for libRASCH, handling all views

Description

This plugin is a general view-plugin, which tries to use for every supported type of signal (eg ecg, bloodpressure) the corresponding view-plugin. For ever channels which is not handled by a specific view-plugin, the default plugin is used (eg ts-view for time-series signals).

Remarks

Win32 specific: This view needs as parent a CMDIFrameWnd. The plugin creates a CMDIChildWnd and this will be returned. (The other view plugins don't create the CMDIChildWnd by themself, these plugins need a CMDIChildWnd as parent.)

Version

0.3.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)

ts-view

Name

`ts-view` — view for time-series

Description

The plugin shows the raw-data for time-series signals. This plugin is the default-plugin for all time-series signals.

Version

0.5.0

Author(s)

Raphael Schneider (rasch@med1.med.tum.de)