nsMCDLibrary

Neuroshare implementation for MC_Rack data

Multi Channel Systems MCS GmbH
Aspenhaustrasse 21
72770 Reutlingen
Germany

Fon +49-71 21-90 92 5 – 0
Fax +49-71 21-90 92 5 – 11

info@multichannelsystems.com

www.multichannelsystems.com

nsMCDLibrary Version: 3.7b

Stand: 2012-09-23

Autor: Jens Pätzold

General

To access MC_Rack data files from the Neuroshare API, the data are mapped to the structure given by Neuroshare API. This API is implemented in a Windows DLL (Dynamic Link Library) or Linux or Mac (Intel) shared library by a set of predefined functions.
MC_Rack files recorded with MC_Rack Version 4.4.6 or older could be read with this version of the library.
Up to now, not all possible data streams within a MC_Rack file could be read by the Neuroshare API.
The nsMCDLibrary.dll is implemented according to the Neuroshare API Version 1.3.
This paper only handles what is specific to data from MC_Rack. For the documentation of the Neuroshare API see there: http://neuroshare.sourceforge.net/

Installation

The Neuroshare library can be downloaded from our web page at http://www.multichannelsystems.com/software/neuroshare.
All packages contain this documentation and the library for the respective system, example source files to access the Neuroshare library directly from your own program code, the sources for the “mexprog“ library that is needed for the interface to Matlab as well as the Matlab functions itself.

Table 1:...

Operating System	Package Name	Neuroshare Library Name
Windows 32bit / 64bit	nsMCDLibrary_3.7b.zip	nsMCDLibrary.dll nsMCDLibrary64.dll
Linux 32bit (build with Ubuntu 10.04 LTS)	sMCDLibrary_Linux32_3.7b.tar.gz or	nsMCDLibrary.so
Linux 64bit (build with Ubuntu 10.04 LTS)	sMCDLibrary_Linux64_3.7b.tar.gz or	nsMCDLibrary.so
Mac OSX (Intel) 32bit and 64bit 'fat' binary	nsMCDLibrary_MacOSX_3.7b.tar.gz	nsMCDLibrary.dylib

Just unzip the file anywhere on your computer.

“mexprog“ could be compiled from within Matlab for you specific OS.
The plain Matlab interface files could be directly used from within our package.
Best is to set your search path in Matlab to the Matlab_Interface folder.

Implementation Details

MC_Rack has two different recording types. Both, continuous and triggered data, can be handled with the Neuroshare library. The mapping is a little bit different for each type, so the mapping is shown for each type separately.

Continuous data mapping

Table 2: Details about continuous data mapping

Stream	Short name	Entity Type	Entity Type Number
Electrode Raw Data	elec	Analog Entity	2
Analog Raw Data	anlg	Analog Entity	2
Filtered Data	filt	Analog Entity	2
Channel Tool Data	chtl	Analog Entity	2
Digital Data	digi	Analog Entity, each binary digit as Event Entity	2 1
Spikes	spks	Segment Entity	3
Trigger	trig	Event Entity	1

Other stream are not mapped up to now.

Triggered data mapping

Table 3:Details about triggered data mapping

Stream	Short name	Entity	Entity Type Number
Electrode Raw Data	elec	Analog Entity and Segment Entity	2 3
Analog Raw Data	anlg	Analog Entity and Segment Entity	2 3
Filtered Data	filt	Analog Entity and Segment Entity	2 3
Channel Tool Data	chtl	Analog Entity and Segment Entity	2 3
Digital Data	digi	Analog Entity and Segment Entity, each binary digit as Event Entity	2 3 1
Spikes	spks	Segment Entity	3
Trigger	trig	Event Entity	1

Please be aware that some streams are mapped to two entities. The contents is essentially the same. Other stream are not mapped up to now.

Missing streams

All other possible streams as average and different parameters are up to now not implemented in the nsMCSLibrary.dll.

Entities

Event entities

Each of the 16 bits of the digital data channel is represented as one event entity. Each change on the individual digital line results on a new event in the entity with the next index. The event information is in one 8bit (byte) value. Its value is 1 for a change from 0 to 1 on the digital line and −1 (255) for a change from 1 to 0 on the digital line.
Triggers are mapped to one event entity for each trigger stream. The event is represented in two 16bit (word) values. The first one gives the slope of the Trigger, the second the Trigger value. For manual triggers the second value gives the trigger number. If trial synchronization is used the trigger event contains another two 16bit values. The 3rd one is the trial number and the 4th one is the stimulus number.

Analog entities

Raw electrode data, analog data and filtered data are mapped to analog entities. Each channel gives one analog entity in Neuroshare. The different gain is considered. The data are given in Volt of the original input.
The 16 bit of the digital data channel are represented in one analog entity as raw data with a range from 0 to 65535, with minimum step size of one. Each sample value is included in the entity.
For continuous data the first index is starting at time 0. Then all data are following with the next index for the next sample point. The time difference of each index is the reciprocal value of the sample rate.
For triggered data the indices are not always continuous in time. If you are reading many indices at one time, you get, according to the Neuroshare specifications, in `pdwContCount` the number of samples that are continuous:
„Although the samples in an analog entity are indexed, they are not guaranteed to be continuous in time and may contain gaps between some of the indexes. When the requested data is returned, `pdwContCount` contains the number of Analog items, starting from `dwStartIndex`, which do not contain a time gap“. [NeuroshareAPI-1-3.pdf]
For analog entities there is no timestamp directly included in the API for reading the data. You get the timestamp with the function `ns_GetTimeByIndex`.

Segment entities

Spike streams are represented in segment entities. Spikes (or wave forms) are sampled independently for each channel. This means there is always only one source per segment entity. The number of sources is always 1. Each original channel from MC_Rack data forms one entity in Neuroshare. The different gain is considered. The data are given in Volt of the original input.
For triggered data all analog entities are also represented as segment entities. One segment contains the data of one sweep for one channel.

Neural event entities

There are no streams in MC_Rack data that are mapped to neural events.

Naming convention for entities

The entities are named as follow:

- 8 letter: stream name with:
  - 4 letters name
  - 4 digits stream number.
- Space
- 4 letters: HWID (Hardware ID)
- Space
- 4 digits: index of channel
- Space
- 8 letters: name of channel (right justified)

From here on, this applies only to event entities of digital data:

- Space
- 2 digits sub channel for events of digital data

for example:
aaaa0000 0000 0000 xxxxxxxx 00
elec0001 0001 0000   21
digi0001 0063 0001   D1
digi0001 0063 0001   D1 02

Configure Behavior

The behavior of the Dll can be configured by a file in the ini-file format. All parameters are in the section **[Settings]**.
BaseIsPressedStart configures how times are interpreted.

0: times start at 0 for each file (default and old behavior)
1: times start at the time when the start button is pressed.

Example for the configuration file:

[Settings]
BaseIsPressedStart = 1

Location of the file

Windows: The configuration file is searched in the same directory and with the same name as the Dll has, but with the extension „.ini“ (nsMCDLibrary.ini)
Linux / Apple: not implemented yet

MCS Specific Matlab Functions

All functions listed in the following can be executed by **mcs_ExampleScript**.
mcs_ExampleScript might serve as example for you and can show you how to work with the given functions.
mcs_ExampleScript generates all parameters needed to run and perform the following Matlab functions:

- mcs_Info.m
- mcs_GetEntities.m
- mcs_Graphic.m

mcs_ExampleScript uses three predefined Neuroshare functions (“**ns_...**”) and a modified

Neuroshare function that is described in the next section of the text.
mcs_SetLibrary('nsMCDlibrary.dll') will always be the first step. This function opens a Neuroshare shared Library and loads the appropriate DLL.
ns_GetLibraryInfo obtains information about the library.
ns_OpenFile opens a neural data file, specified by filename or pathway and returns a file handle called hFile, that contains an identification number. This is important when you want to call the following functions.

ns_GetFileInfo is a function that contains some information about the data file like *FileType, EntityCount, TimeStampResolution, TimeSpan, AppName,Time_Year, Time_Month, Time_Day, Time_Hour, Time_Min, Time_Sec, Time_MilliSec, FileComment.*
FileType is given as a sequence of abbreviations.
It is a combination of several properties: Layout, continuous or not, recorded streams.

For example:

FileInfo =

FileType: 'MEA cont Di,El,An'

The Channel Type is named according to the title given in MC_Rack (first two letters).
Channel Types of streams which are not mapped up to now can be neglected.
The table below shows possible designations. They can occur in various combinations.

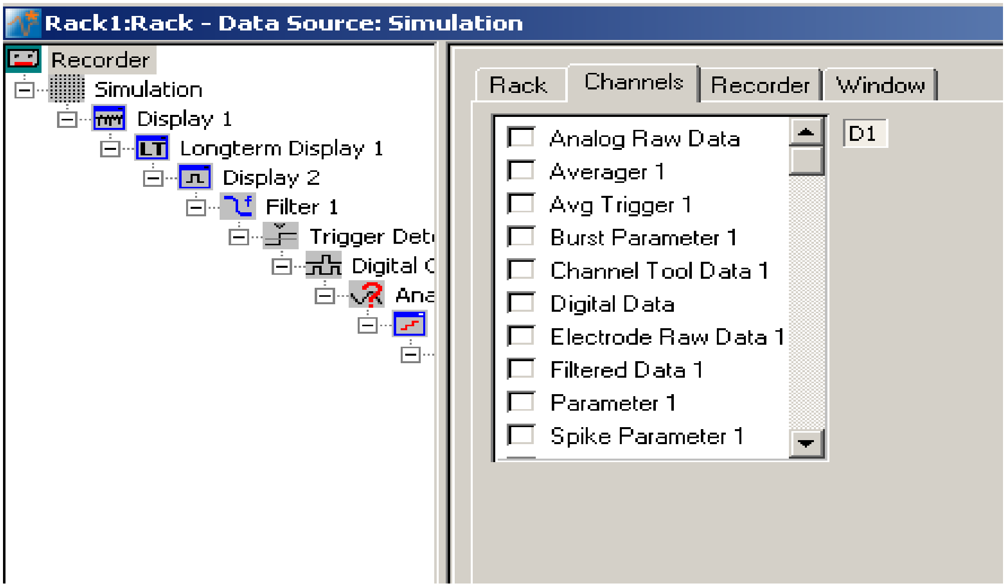

Figure 1: MC_Rack with several channels types

Table 4: Naming convention for FileType

Layout	**Recording Mode**	**Channel Type short form**	**Channel Type long form**
LIN	cont	Di	Digital Data
MEA	trig	El	Electrode Raw Data
DMEA	trig_ss	An	Analog Raw Data
Conf	xxx	Fi	Filtered Data
XXX		Sp	Spikes
		Tr	Trigger
		Ch	Channel Tool Data

mcs_SetLibrary.m

Rough abstract of functionality:

mcs_SetLibrary is a further implementation of **ns_SetLibrary. mcs_SetLibrary** supplements **ns_SetLibrary** through an optimized sequence of events if **ns_SetLibrary** is not able to open the required file. **mcs_SetLibrary** returns more information in the form of error and status messages and in case of failure, the function parse the Matlab path and tries to open the file using different paths.

Usage:

[ns_RESULT] = mcs_SetLibrary(filename)

Description:

Opens the dynamic linked library specified by filename or in case of failure tries to open it using a different path

Parameters:

filename — Pointer to a null-terminated string that specifies the name of the file to open.

Return Values:

ns_RESULT	This function returns 0 if the file is successfully opened. Otherwise the following error message is generated:'*ns_SetLibrary does not find the library:...* ' and maybe '*Try to open …*' and '*Success'*

mcs_Info.m

Rough abstract of functionality:

mcs_Info is a function to show information about the streams that are in a certain file given as *stream, type, n, TimeSpan and infotext*.
The output contains the stream name as a eight digits sequence, the *type* of entity, *n* is the number of channels per stream, *TimeSpan* gives you information about the duration of data collection and *infotext* contains optional information about the data type.
mcs_Info uses **ns_GetFileInfo** to receive the information *EntityCount* saved in *FileInfo*. EntityCount is necessary to call the next function **ns_GetEntityInfo** and to get *EntityLabel* and *EntityType* as a part of *entity*. *EntityLabel* includes some information about the stream that is connected to *EntityType* in a structure *entities*.
The function returns a table with information if the number of output arguments is 0. Otherwise the information is given as a matrix. That might be helpful if you like to work on with the given values.

Usage:

[varargout] = mcs_Info(hFile, varargin)

Description:

Generates information about entities. This information is returned in a structured output via CommandWindow (matrix or table).

Parameters:

hFile	Handle/Indentification number to an open file.
varargin	Variable length input argument list,"variable argument in"

Return Values:

- without return value: information is shown on the display as a table.
- with return value: cell matrix of this information.

mcs_GetEntities.m

Rough abstract of functionality:

mcs_GetEntities is a function to generate a vector of entity numbers of entities with a given characteristic.
mcs_GetEntities uses **ns_GetFileInfo** to produce the structure FileInfo. FileInfo contains information about EntityCount, needed to call **ns_GetEntityInfo. ns_GetEntityInfo** retrieves general entity information and type of entity saved in the structure EntityInfo that contains EntityLabel, EntityType and ItemCount.

Usage:

[entities] = mcs_GetEntities (hFile,stream_name, varargin)

Description:

Generates a vector of entity numbers that belongs to the stream with the name stream_name and possible of a certain entity type given in varargin from the data file referenced by hFile.

Parameters:

hFile	Handle/Indentification number to an open file
stream_name	e.g. 'elec0001', 8 letter: stream name with: 4 letters name, 4 digits stream number.
varargin	Variable length input argument list, "variable argument in"(optional) number of entity type

Return Values:

entities	Array of entity numbers

mcs_Graphic.m

Rough abstract of functionality:

mcs_Graphic plots analog and triggered entities. **mcs_Graphic** uses **ns_GetEntityInfo** to get information about the EntityType. With this information, it is possible to switch between analog (EntityType = 2)and segment (EntityType = 3) data.
In case of analog data, **ns_GetAnalogInfo** is called to get information about the location locX and locY.
In case of segment data, **ns_GetSegmentInfo** and **ns_GetSegmentSourceInfo** are called to get the location of the data as well.
In both cases **mcs_Graphic** manages x- and y-data shown as a plot.
Depending on chosen call of the function, you can select entities/ channels of interest. Therefore it is possible to pick them out using EntityID .

Usage:

mcs_Graphic(hFile, EntityID, firstItem, ItemCount)

Description:

Generates a graphical output for analog and segment data

Parameters:

hFile	Handle/Indentification number to an open file.
EntityID	Identification number(s) of the entity in the data file
firstItem	start
itemCount	number of items

Return Values: /

Building mexprog from Sources

xyz

FAQ

xyz

Missing

The function `ns_GetLastErrorMsg` is not implemented yet.