```
function EMU_rawData2mat(sbj_name,block_nums,nchan, microchan)
% Convert the BlackRock NS3 and NS5 files to individual channel .mat
file and save
% them into .\Indiv_chans directory
%
% New BlackRock format saves all recording channels in one .mat file,
this
% function is created to split and save the raw data file from
blackrock to indivdual
% channel files
%
%
% Input: sbj_name, i.e. YAA
%        block_num, i.e. 001
%        nchan, i.e. 128

% Output: each channel is save as one .mat file including photodiode
(*Photodiode.mat) and
% audio (*audio.mat) channels
%
%   => Example:
%       ecog_rawData2mat('YAA',1:3,128)
%
% Dependencies:
% NSP_sync.m - syncing two NSP data
% write_analogInfo.m - exporting single channel data info
% write_conversionlog.m - creating a log for the conversion
%
% First created as ecog_rawData2mat : Feb 28, 2018 - Yvonne Chen - BCM
% Various revisions - see ARCHIVE
% Current version is created: Jan 28, 2021 - Yvonne Chen - BCM

%% Number of NSPs for conversion
if nargin < 4 %no microchan input, no mic or ET sync input
    microchan = [];
end
%% HOUSE keeping
%- BlackRock Convert code path
addpath('/Volumes/ecog/Foster_Lab/CODE/Toolboxes/NPMK');
%- Code function path
addpath('/Volumes/ecog/Foster_Lab/CODE/MASTER/BASICS/Fx_rawData2mat');

%- Find subject directory and read in all file names
subdata_dir = sprintf('/Volumes/ecog/ECoG_Data/%sDatafile/
DATA',sbj_name);
allfiles = dir(subdata_dir);
scan_files = {allfiles([allfiles.isdir]).name}; %find all the
directories
scan_file_names = scan_files(~[strcmp(scan_files, '.') |
strcmp(scan_files, '..')]); %read in all file names
```

```
%loop through blocks
for block_index = 1:length(block_nums)
    %% setup data structures
    block_num = sprintf('%03d',block_nums(block_index)); %create block
name
    % if no files found, return error message
    if sum(contains(scan_file_names,sprintf('EMU-
%03d',block_nums(block_index)))) == 0 &&
sum(contains(scan_file_names,sprintf('%03d',block_nums(block_index))))
== 0%no file find
        fprintf('%03d files are not found...
\n',block_nums(block_index))
        return
    end

    % Find directory with the EMU block number and create indiv data
folder
    try %new naming system
        dir_name =
scan_file_names{contains(scan_file_names,sprintf('EMU-
%03d',block_nums(block_index)))};
    catch %catch old naming systems
        dir_name =
scan_file_names{contains(scan_file_names,sprintf('%03d',block_nums(blo
ck_index)))};
    end

    rawdata_dir = sprintf('%s/%s',subdata_dir,dir_name);
    % make individual channel directory if they did not exist
    if ~exist(sprintf('%s/Indiv_chans',rawdata_dir),'dir')
       mkdir(sprintf('%s/Indiv_chans',rawdata_dir))
    end
    data_dir = sprintf('%s/Indiv_chans',rawdata_dir);


    %% Load raw files
    % find data file names
    blockfiles = dir(rawdata_dir);
    data_file_names = {blockfiles(~[blockfiles.isdir]).name}; %find
all the directories
    %check how many NSPs used for recording - find name index
    nsp_namei_ns5 = find(contains(data_file_names,'.ns5')); %1 or 2
NSP(s)
    nsp_namei_ns3 = find(contains(data_file_names,'.ns3')); %1 or 2
NSP(s)

    rawData_name5 = data_file_names{nsp_namei_ns5(1)};
    rawData_name3 = data_file_names{nsp_namei_ns3(1)};
    % load NS5 and NS3
    fprintf('Loading NS5...\n')
    openNSx(sprintf('%s/%s',rawdata_dir,rawData_name5),'p:double');
    fprintf('Loading NS3...\n')
    openNSx(sprintf('%s/%s',rawdata_dir,rawData_name3),'p:double');

    NS3_data1 = NS3; %NS3 contains all the ECoG recording data and 
other channel info - 2000 Hz sample rate
    NS5_data1 = NS5; %NS5 contains all the microwire recording, 
auxiliary input, including photodiode, audio, etc. - 30,000 Hz sample 
rate
    clear NS3 NS5

    %Data segments
    NS3_si1 = size(NS3_data1.MetaTags.DataPoints,2); % check total 
data seements
    NS5_si1 = size(NS5_data1.MetaTags.DataPoints,2); % check total aux 
data segments

    %% Load NSP2 if needed
    if length(nsp_namei_ns3) > 1 %2 NSP recordings
        fprintf('NSP2 data detected...\n')
        rawData_name5 = data_file_names{nsp_namei_ns5(2)};
        rawData_name3 = data_file_names{nsp_namei_ns3(2)};

        % load NS5 and NS3
        fprintf('Loading NS5...\n')
        openNSx(sprintf('%s/
%s',rawdata_dir,rawData_name5),'p:double');
        fprintf('Loading NS3...\n')
        openNSx(sprintf('%s/
%s',rawdata_dir,rawData_name3),'p:double');

        NS3_data2 = NS3; %NS3 contains all the ECoG recording data and 
other channel info - 2000 Hz sample rate
        NS5_data2 = NS5; %NS5 contains all the microwire recording, 
auxiliary input, including photodiode, audio, etc. - 30,000 Hz sample 
rate
        clear NS3 NS5

        %Data segments
        NS3_si2 = size(NS3_data2.MetaTags.DataPoints,2); % check total 
data seements
        NS5_si2 = size(NS5_data2.MetaTags.DataPoints,2); % check total 
aux data segments

        %% Sync 2 NSPs
        NSP_counts = 2; %2 NSPs recording
        %- Run NSP sync - out NS3 and NS5 data length info for
```

```
conversion
        %and trime type NSP vs. PD
        [NS3_length, NS5_length, trim_type] =
NSP_sync(NS3_data1,NS3_data2,NS5_data1,NS5_data2);
    else
        % 1 NSP recording
        NSP_counts = 1; %1 NSP recording

        %EB added to handle the NSP1 ith two sections:
        % 1 NSP recording but 2 NSP active: there will be two sections
of
        % data!
        if contains(data_file_names{nsp_namei_ns3},'_NSP')
            NS3_length{1} = 1:NS3_data1.MetaTags.DataPoints(2); % EB
06/21 added and commented for blocks 057-058 of YDT (1 NSP rec but 2
sections)
            NS5_length{1} = 1:NS5_data1.MetaTags.DataPoints(2);
        else
            NS3_length{1} = 1:NS3_data1.MetaTags.DataPoints;
            NS5_length{1} = 1:NS5_data1.MetaTags.DataPoints;
        end
    end % end of NSP syncing


    %% CHECK for data for packet drop
    if NS5_si1 == 1 && length(nsp_namei_ns5) == 1 % 1 NSPs and no
packet drop
        %Full Data length
%         data_length = 1:NS3_data1.MetaTags.DataPoints;
%         aux_length = 1:NS5_data1.MetaTags.DataPoints;
    end

    %Log data info
    %- Conversion log
    log_name = sprintf('%s/
%sData_%s_conversion_log_output.txt',data_dir,sbj_name,block_num);
    fid_log = fopen(log_name,'wt');
    write_conversionlog
    %% Converting raw data to single channel .mat files
    %-------------------------------%
    % loop through EEG data channels %
    %-------------------------------%
    disp ('Converting to single channel data ...')
    for ee = 1:nchan
        if ee <= 256 %NS3_data1.MetaTags.ChannelCount
            CurrentData = NS3_data1; %currently converting data from
NSP1
            chanID = ee; %keeping track of channel count - to match
ELectrodeID in ElectrodeInfo
            data_length = NS3_length{1};
            si = NS3_si1; %when there are more than 1 segment of data
        else
            CurrentData = NS3_data2; %currently converting data from 
NSP2
            chanID = ee - 256; %keeping track of channel count -  to 
match ELectrodeID in ElectrodeInfo
            data_length = NS3_length{2};
            si = NS3_si2;
        end

        % define analogTrace
        % - find the correct row index to covert
        % Match electrode ID
        convert_idx = 
find(cell2mat({CurrentData.ElectrodesInfo.ElectrodeID})==chanID);
        if ~isempty(convert_idx) %if electrode recorded
            if size(CurrentData.MetaTags.DataPoints,2) == 1 % if only 
one seg data to convert
                analogTraces = 
CurrentData.Data(convert_idx,data_length)/4;
            else % More segments
                analogTraces = CurrentData.Data{1,si}
(convert_idx,data_length)/4;
            end


            % generate single channel data info - create analogInfos
            % Find label for the channel
            eleclabel = CurrentData.ElectrodesInfo(convert_idx).Label;
            analogInfos = 
write_analogInfo(CurrentData,convert_idx,eleclabel,'data');

            % save individual data set
            save(sprintf('%s/%sDatafile%s_ch%.d.mat',data_dir, 
sbj_name, block_num,ee),'analogTraces', 'analogInfos');
            log_str = sprintf('Saving block %s: channel %g -- %s; size 
%g %g\n',block_num,ee,eleclabel,data_length(1), data_length(end));

            clear analogInfos analogTraces
        else %if electrode did not record
            log_str = sprintf('!! No recording !! block %s: channel 
%g\n',block_num,ee);
        end
        %- Print to conversion log
        Print2Log(fid_log,log_str);
    end %end of the data channel loop
    %% Converting Aux data to single channel .mat files
    %-----------------------------%
    % Loop through Aux channels  %
    %-----------------------------%
    disp ('Converting to auxiliary channel data ...')
    aux_labels = {'Photo';'Audio'}; % can to added for additional data
    for aa = 1:length(aux_labels)
        curr_aux_label = aux_labels{aa}; %get the channel label

        for nspi = 1:NSP_counts %loop through 2 NSPs if needed
            CurrentAux = eval(sprintf('NS5_data%g',nspi));
            convert_aux_idx =
find(contains({CurrentAux.ElectrodesInfo.Label},curr_aux_label));
            aux_length = NS5_length{nspi};
            ai = eval(sprintf('NS5_si%g',nspi));

            if ~isempty(convert_aux_idx) %if Aux recorded
                %generate analogTrace
                if size(CurrentAux.MetaTags.DataPoints,2) == 1 % if
only one seg data to convert
                    analogTraces =
CurrentAux.Data(convert_aux_idx,aux_length)/4;
                else % More segments
                    analogTraces = CurrentAux.Data{1,ai}
(convert_aux_idx,aux_length)/4;
                end

                %generate single channel data info
                analogInfos =
write_analogInfo(CurrentAux,convert_aux_idx,curr_aux_label,'aux');

                %save aux data
                if nspi == 1
                    save(sprintf('%s/%sDatafile%s_%s.mat',data_dir,
sbj_name, block_num,curr_aux_label),'analogTraces', 'analogInfos');
                else
                    save(sprintf('%s/
%sDatafile%s_%s_NSP%g.mat',data_dir, sbj_name,
block_num,curr_aux_label,nspi),'analogTraces', 'analogInfos');
                end
                log_str = sprintf('Saving block %s: channel
%s\n',block_num, curr_aux_label);

                clear analogTraces analogInfos
            else %no recording
                log_str = sprintf('!! No recording !! block %s:
channel %s\n',block_num, curr_aux_label);
            end %end of empty channel check
            %- Print to conversion log
            Print2Log(fid_log,log_str);
        end % end of the NPS loop
    end %end of the aux channel loop

    %% Converting microwire data to single channel .mat files
    %--------------------------------%
    % loop through microwire channels %
    %--------------------------------%
    if ~isempty(microchan) % skip if no microchan to convert
        disp ('Converting to microwire channel data ...')
        %total_chan_counter = nchan + length(microchan); %total
channel recorded

        for mm = 1:length(microchan)
            %if mm < 256 - only 1 NSP & if > 256 - 2 NSPs
            if mm <= 256 %1 NSP
                MicroData = NS5_data1;
                micro_length = NS5_length{1};
                mi = NS5_si1; %when there are more than 1 segment of
data
            else %2 NSPs
                MicroData = NS5_data2;
                micro_length = NS5_length{2};
                mi = NS5_si2;
            end

            convert_micro_idx =
find(cell2mat({MicroData.ElectrodesInfo.ElectrodeID})==microchan(mm));
            if ~isempty(convert_micro_idx) %if electrode recorded
                % Define analogTrace
                if size(MicroData.MetaTags.DataPoints,2) == 1 % if
only one seg data to convert
                    analogTraces =
MicroData.Data(convert_micro_idx,micro_length)/4;
                else % More segments
                    analogTraces = MicroData.Data{1,mi}
(convert_micro_idx,micro_length)/4;
                end

                % generate single channel data info - create
analogInfos
                % Find label for the channel
                meleclabel =
MicroData.ElectrodesInfo(convert_micro_idx).Label;
                analogInfos =
write_analogInfo(MicroData,convert_micro_idx,meleclabel,'micro');

                % save individual data set
                save(sprintf('%s/
%sDatafile%s_micro_ch%.d.mat',data_dir, sbj_name,
block_num,mm),'analogTraces', 'analogInfos');
                log_str = sprintf('Saving block %s: mircowire channel
%g -- %s; size %g %g\n',block_num,mm,meleclabel,micro_length(1),
micro_length(end));
```

```
                clear analogInfos analogTraces
            else %if electrode did not record
                log_str = fprintf('!! No recording !! block %s:
microwire channel %g\n',block_num,mm);
            end %end of empty data check

            %- Print to conversion log
            Print2Log(fid_log,log_str);
        end %end of the microwire channel loop
    end % end of the microwire conversion

    %% Block Complete
    log_str = sprintf('----Block %s complete----\n',block_num);
    %- Print to conversion log
    Print2Log(fid_log,log_str);
    fclose(fid_log);
end % end of the block loop

end

function Print2Log(fid,log_str)
%- Print additional lines to data conversion log
    fprintf('%s', log_str);
    fprintf(fid, '%s', log_str);
end
```