CBMEX

cbmex provides an online interface between Matlab and the Cerebus Central app. With it you can read spike timestamps and optionally, continuous sample data. It also allows file recording to be started and stopped programmatically based on your analysis of the data. Digital outputs can be also be controlled programmatically through this interface.

The interface must be initialized by using the open command. In order to start data collection, the trialconfig command must be executed with the parameter set to 1. Collected data can then be read online from the NSP during which file recording can be started/stopped and the digital output ports can be controlled. When complete close the interface with the close command.

OPEN

Usage:

cbmex('open')
cbmex('open', [interface], [thread_priority], [force_loopback])

Description:

Call this before doing anything else. Initializes cbmex, connects to Central, and prints out the current cbmex version number.

Parameters:

Interface:	(optional) should be 1 (use central) or 2 (use UDP). Defaults to 1.
thread_priority:	(optional) should be one of: 0 - Normal 1 - Above normal 2 - Highest 3 - Time critical Default: 0
force_loopback:	(optional) is only meaningful if 'interface' is udp (2), and should be 1 or 0. Defaults to 0.

CLOSE

Usage:

cbmex('close')

Description:

Call this to disconnect from central.

TIME

Usage:

cbmex('time')

Description:

Returns the current NSP time in seconds. Note: The timer starts over if Reset is pressed on Central and when recording starts

FILECONFIG

Usage:

cbmex('fileconfig', filename, comments, action)

Description:

Starts or stops file recording to the filename specified including the comments provided. Filename contains the full path to the filename.
Note: if the file application is not running it is opened but recording will not start.

Parameters:

Action:	0 = file recording is stopped
	1 = file recording is started

DIGITALOUT

Usage:

cbmex('digitalout', channel, value)

Description:

Sends a value to the Digital Out ports on the NSP as long as the port is not configured to monitor a channel or set for timed output.

Parameters:

Channel:	Channel number of the digital out port to affect
	153 = dout1
	154 = dout2
	155 = dout3
	156 = dout4
Value:	1 sets dout to ttl high and 0 sets dout to ttl low

TRIALCONFIG

Usage:

cbmex('trialconfig', active)

Description:

Sets cbmex to start or stop recording data.

Parameters:

Active:	1 = this says "flush your data cache and start collecting data right now".
	0 = this says "stop collecting data right now".

Return Value:

If you specify a left-hand-side (return) argument, you'll get either 1 or 0. 1 indicates that the dll is currently active (recording data), 0 indicates inactive.

TRIALDATA

Usage:

[spike_data, data_start_time, lfp_data] = cbmex('trialdata', clear_buffer)

Description:

Returns collected data since collection started or the last time the data buffer was cleared. Data collection starts only after the trialdconfig command is executed with the parameter set to 1.

The cache will hold up to 1048576 events. The continuous sample cache will hold 100000 samples per channel (about 3 seconds). Matlab will of course be a pain if you let it get that large. Note that it's your responsibility to flush the data cache once in a while, by calling :

cbmex('trialconfig', 1)
or
cbmex('trialdata', 1)

Note: If the buffer fills up, no more data will be added to the channel.

Parameters:

clear_buffer: tells the library whether it should also clear all the data and reset its recording time to the current time.

Return Value:

spike_data: Returns a matrix of events, where the columns are :
For spike channels :
'channel name' [unclassified timestamps] [u1 timestamps] [u2 timestamps] [u3 timestamps] ...
For digital input channels:
'channel name' [timestamps] [values] ...remaining columns are empty...

data_start_time: (optional) Time (in seconds) that the data buffer was most recently cleared. If this call to 'trialdata' cleared the buffer (if 'clear_buffer' is 1), you get the time at which the previous buffer started (different from previous cbmex behavior).

lfp_data: (optional) Provides continuous sample data... each row in this matrix looks like :
[channel number] [sample rate (in samples / s)] [values]

Note: If you change the sampling rate on a given channel, its values are erased so the next time you call cbmex('trial_data'), be aware that sample values on different channels may not 'line up'.

Example scripts:

Example 1:

```
% Author and Date: Ehsan Azar  14 Sept 2009
% Copyright:       Blackrock Microsystems
% Purpose: Realtime spectrum display. All sampled channels are displayed.

close all;
clear variables;

f_disp = 0:0.1:15;       % the range of frequency to show spectrum over.
% Use f_disp = [] if you want the entire spectrum

collect_time = 1; % collect samples for this time
display_period = 5; % display spectrum every this amount of time

cbmex('open'); % open library

proc_fig = figure; % main display
set(proc_fig, 'Name', 'Close this figure to stop');
xlabel('frequency (Hz)');
ylabel('magnitude (dB)');

cbmex('trialconfig', 1); % empty the buffer

t_disp0 = tic; % display time
t_col0  = tic; % collection time
bCollect = true; % do we need to collect
 % while the figure is open
while (ishandle(proc_fig))

    if (bCollect)
        et_col = toc(t_col0); % elapsed time of collection
        if (et_col >= collect_time)
            [spike_data, t_buf1, lfp_data] = cbmex('trialdata',1); % read some
data
            nGraphs = size(lfp_data,1); % number of graphs
            % if the figure is still open
            if (ishandle(proc_fig))
                % graph all
                for ii=1:nGraphs
                    fs0 = lfp_data{ii,2};
                    % get the ii'th channel data
                    data = lfp_data{ii,3};
                    % number of samples to run through fft
                    collect_size = min(size(data), collect_time * fs0);
                    x = data(1:collect_size);
                    %uncomment to see the full rang
                    if isempty(f_disp)
                        [psd, f] = periodogram(x,[],'onesided',512,fs0);
                    else
                        [psd, f] = periodogram(x,[],f_disp,fs0);
                    end
                    subplot(nGraphs,1,ii,'Parent',proc_fig);
                    plot(f, 10*log10(psd), 'b');title(sprintf('fs = %d t = %f',
fs0, t_buf1));
                    xlabel('frequency (Hz)');ylabel('magnitude (dB)');
                end
                drawnow;
```

```
            end
            bCollect = false;
        end
    end

    et_disp = toc(t_disp0); % elapsed time since last display
    if (et_disp >= display_period)
        t_col0  = tic; % collection time
        t_disp0 = tic; % restart the period
        bCollect = true; % start collection
    end
end
cbmex('close'); % always close
```

Example 2:

```
% Author & Date:    Hyrum L. Sessions   14 Sept 2009
% Copyright:        Blackrock Microsystems
% Purpose:          Read serial data from the NSP and compare with a
%                   predefined value.  If it is the same, generate a
%                   pulse on dout4

% This script will read data from the NSP for a period of 30 seconds.  It
% is waiting for a character 'd' on the Serial I/O port of the NSP.  If
% received it will generate a 10ms pulse on Digital Output 4

% initialize
close all;
clear variables;

run_time = 30;      % run for time
value = 100;        % value to look for (100 = d)
channel_in = 152;   % serial port = channel 152, digital = 151
channel_out = 156;  % dout 1 = 153, 2 = 154, 3 = 155, 4 = 156

t_col = tic;        % collection time

cbmex('open');              % open library
cbmex('trialconfig',1);     % start library collecting data

start = tic();

while (run_time > toc(t_col))
    pause(0.05);    % check every 50ms
    t_test = toc(t_col);
    spike_data = cbmex('trialdata', 1);  % read data
    found = (value == spike_data{channel_in, 3});
    if (0 ~= sum(found))
        cbmex('digitalout', channel_out, 1);
        pause(0.01);
        cbmex('digitalout', channel_out, 0);
    end
end

% close the app
cbmex('close');
```