MAPPS - Matlab®
Programmer's Guide

eyesDx™

Exploring human behavior objectively

Contents

Overview 3

[*PB*] mxGetProjectName 4
[*PB*] mxGetStartTime 5
[*PB*] mxGetTime 6
[*PB*] mxSetTime 7

[*SP*] mxListBuses 8
[*SP*] mxGetBuses 9
[*SP*] mxCreateBus 11
[*SP*] mxUpdateBus 13
[*SP*] mxDisplayBus 14
[*SP*] mxHideBus 15
[*SP*] mxDeleteBus 16
[*SP*] mxDeleteByTime 17
[*SP*] mxDeleteByIndex 18

[*ET*] mxGetFixations 19
[*ET*] mxUpdateFixations 20
[*ET*] mxGetRois 21
[*ET*] mxGetRoiIntersections 23
[*ET*] mxListCanvases 24
[*ET*] mxUpdateOverlay 25

[*CM*] mxSignOffClient 27

Overview

The MAPPS-Matlab interface enables a user to access and manipulate data sources and visualization capabilities of MAPPS from Matlab in both post-processing, as well as live mode configurations, if licensed. The interface consists of a set of pre-compiled MEX-files that can be called directly from Matlab. Currently, the interface only supports a shared memory connection, requiring both MAPPS and the Matlab session to run on the same machine.

Other prerequisites include:

- Microsoft Windows XP, Vista, or 7 (32-bit or 64-bit)
- Matlab R2010a or later (simply not tested on earlier versions)
- Microsoft Visual C++ 2010 Redistributable Package (x86 or x64, determined by the Matlab version)
- Matching header file versions between MAPPS and the interface MEX-files (pre-set at the time when MAPPS and MEX-files are compiled)

The layout and ordering of interface calls described in this guide follows the broad functional differentiation amongst them. The first group of calls allows control over playback of data sets ([*PB*]). A second set of function calls implement bi-directional flow and display control of data encapsulated in MAPPS buses on the Signals Page (labeled with [*SP*]). The third group of operations is geared towards data types hosted on the Eye Tracking Page, including fixation, regions (or area) of interest, and video overlays (labeled with [*ET*]). A fourth group deals with maintenance of the MAPPS-Matlab client connection ([*CM*]).

Multiple Matlab clients are allowed to connect to MAPPS simultaneously. To support this capability, the first input argument to each function call requires the user to provide a data structure for identifying the calling Matlab client to MAPPS. Currently, the structure only needs to contain a field with a user-specified character string, which is used as the client name. This Client structure will be leveraged in future to control additional parameters of the connection between MAPPS and Matlab.

[*PB*] mxGetProjectName

mxGetProjectName is used to retrieve the name of the current MAPPS project or dataset.

Syntax and Arguments:

```
name = mxGetProjectName(
                         struct: Matlab client
                        );
```

with

```
string name
```

Notes:
1) If output argument `name` is not provided, the value is echoed in the Command Window of Matlab.

[*PB*] mxGetStartTime

mxGetStartTime is used to retrieve the time value of the start point of the current MAPPS dataset as a 64-bit value representing the number of 100-nanosecond intervals since January 1st, 1601 (UTC), also referred to as FILETIME.

Syntax and Arguments:

```
time = mxGetStartTime(
                       struct: Matlab client
                      );
```

with

```
int64 time
```

Notes:

2) If output argument `time` is not provided, the current time is echoed in the Command Window of Matlab.

[*PB*] mxGetTime

mxGetTime is used to retrieve the current value of the MAPPS time marker in seconds with respect to the start of the entire dataset.

Syntax and Arguments:

```
time = mxGetTime(
                  struct: Matlab client
                 );
```

with

```
single time
```

Notes:

3) If output argument `time` is not provided, the current time is echoed in the Command Window of Matlab.

[*PB*] mxSetTime

mxSetTime is used to instruct MAPPS to move the current time marker to the offset specified in seconds with respect to the start of the entire dataset.

Syntax and Arguments:

```
mxSetTime(
           struct: Matlab client,
           scalar: Offset in seconds
          );
```

[*SP*] mxListBuses

mxListBuses is used to request the layout of MAPPS buses, which is reported in terms of their fields, the data storage type of each field, and time offsets of the 1st and last sample of each bus.

Syntax and Arguments:

```
[Times, Buses] = mxListBuses(
                               struct: Matlab client
                             );
```

with

```
struct Times
          ├─ Name of bus 1 ──┬─ Time of first sample
          │                  └─ Time of last sample
          ╞═ :
          ├─ Name of bus n ──┬─ Time of first sample
          │                  └─ Time of last sample
          └─ Global_MAPPS_Time ─┬─ Time of first bus sample, overall
                                └─ Time of last bus sample, overall

struct Buses
          ├─ Name of bus 1
          │             ├─ Name of field 1 with storage type
          │             ╞═ :
          │             └─ Name of field m with storage type
          ╞═ :
          └─ Name of bus n
                        ├─ Name of field 1 with storage type
                        ╞═ :
                        └─ Name of field p with storage type
```

Notes:

1) Values reported in `Times` are defined as offsets in millisecond with respect to the start of the entire dataset, including buses, video, audio, etc.
2) Apart from storage type, fields in `Buses` contain no actual data.
3) If output arguments `Times` and `Buses` are not provided, the results are echoed in the Command Window of Matlab.

[*SP*] mxGetBuses

mxGetBuses is used to retrieve data from one or more fields of one or more buses in MAPPS. For each bus, an additional field is reported containing the offset in millisecond of each timeslice with respect to the start of the entire dataset.

Syntax and Arguments:

```
Data = mxGetBuses(
                  struct: Matlab client,
                  cellstr: Names of fields,
                  cellstr: Names of associated buses,
                  scalar (optional): Start time of requested interval
                                     in terms of millisecond offset,
                  scalar (optional): End time of requested interval in
                                     terms of millisecond offset
                  );
```

or

```
Data = mxGetBuses(
                  struct: Matlab client,
                  struct: Buses
                            ├─ Name of bus 1
                            │          ├─ Name of field 1
                            │          ╞═ :
                            │          └─ Name of field m
                            ╞═ :
                            └─ Name of bus n
                                       ├─ Name of field 1
                                       ╞═ :
                                       └─ Name of field p,
                  scalar (optional): Start time of requested interval
                                     in terms millisecond offset,
                  scalar (optional): End time of requested interval in
                                     terms millisecond offset
                  );
```

with

```
struct Data
          ├─ Name of bus 1
          │          ├─ Time offsets in millisecond for bus 1
          │          ├─ Data from field 1
          │          ╞═ :
          │          └─ Data from field m
          ╞═ :
          └─ Name of bus n
                     ├─ Time offsets in millisecond for bus n
```

```
├─ Data from field 1
├─ :
└─ Data from field p
```

Notes:

1) Time offsets in millisecond are reported for every requested bus that exists, irrespective of whether any of the requested fields for a specific bus exist.
2) To easily retrieve all data from all buses, pass `Buses`, reported by mxListBuses, as input parameter for mxGetBuses.

Example:

To retrieve data from 'Signal_1' and 'Signal_2' for 'Bus_1', as well as 'Signal_1' for 'Bus_2' from 20 seconds after the start of the dataset over a 50-second interval, use:

```
Client.Name = 'My_Matlab';
startTime = 20;  % in sec.
duration = 50;   % in sec.
Data = mxGetBuses(Client, {'Signal_1', 'Signal_2', 'Signal_1'}, ...
                  {'Bus_1', 'Bus_1', 'Bus_2'}, ...
                  startTime * 1000, (startTime + duration) * 1000);
```

or

```
Client.Name = 'My_Matlab';
startTime = 20;  % in sec.
duration = 50;   % in sec.
Buses = struct('Bus_1', struct('Signal_1', [], 'Signal_2', []), ...
               'Bus_2', struct('Signal_1', []));
Data = mxGetBuses(Client, Buses, startTime * 1000, ...
                  (startTime + duration) * 1000);
```

To retrieve only the millisecond offsets for 'Bus_1', use:

```
Client.Name = 'My_Matlab';
Data = mxGetBuses(Client, [], {'Bus_1'});
```

[*SP*] mxCreateBus

mxCreateBus is used to request MAPPS to spawn a new bus and for specifying the layout of the requested bus. The bus can contain one or more signals, listed as fields. It is assumed that these signals are sampled on the same schedule.

Syntax and Arguments:

```
mxCreateBus(
             struct: Matlab client,
             scalar: # of timeslices,
             struct: Name of bus ─┬─ Name of field 1 with storage type
                                  ├─ Name of field 2 with storage type
                                  ╞═ :
                                  └─ Name of field n with storage type
            );
```

or

```
mxCreateBus(
             struct: Matlab client,
             scalar: # of timeslices,
             string: Name of bus,
             string/cellstr: Names of fields,
             string/cellstr: MAPPS storage type of each field,
                             respectively
            );
```

Notes:

1) `struct` should contain at least one non-empty field.
2) Fields in `struct` do not have to be empty – only the storage type is derived from the contents for each field. If passing only `[]` for a field, the storage type will default to `double`.
3) Only group together fields with the same sampling schedules into single bus, and create additional buses for fields with different sampling schedules.
4) The new bus is added at the bottom of the Signals Page.

Example:

To spawn a bus, called 'My_Bus', which will contain 101 timeslices, and consist of two fields, called 'Signal_1' with `float` values and 'Signal_2' with `int` values, use:

```
Client.Name = 'My_Matlab';
nT = 101;
Bus = struct('My_Bus', struct('Signal_1', single([]), 'Signal_2', ...
              int32([])));
mxCreateBus(Client, nT, Bus);
```

or

```
Client.Name = 'My_Matlab';
nT = 101;
mxCreateBus(Client, nT, 'My_Bus', {'Signal_1', 'Signal_2'}, ...
            {'float', 'int'});
```

[*SP*] mxUpdateBus

mxUpdateBus is used to replace data of fields in the specified bus based on an interval set by the 1st and last time values of the millisecond offsets provided. The number of timeslices for each field must equal the number of millisecond offsets.

Syntax and Arguments:

```
mxUpdateBus(
             struct: Matlab client,
             numeric vector: time offsets in millisecond,
             struct: Name of bus ──┬─ Data for field 1
                                   ├─ Data for field 2
                                   ├─ :
                                   └─ Data for field n
           );
```

Notes:

1) Only buses created by the Matlab user can be updated.
2) Data for fields that do not exist are ignored.
3) In case no data is provided for fields that exist, zeros are passed for integer fields, NaNs (Not-a-Number) are passed for floating point fields, and empty strings are passed for character fields.
4) Time offsets are not checked for ascending order.
5) Time offsets are allowed to have irregularly intervals.

Example:

To update field 'Signal_1' of a previously-spawned bus, called 'My_Bus', to contain a 2 Hz sinusoid from 5 seconds after the start of the dataset over a 2-second interval at a sampling rate of 100 Hz, use:

```
Client.Name = 'My_Matlab';
startTime = 5;  % in sec.
duration = 2;   % in sec.
fz = 50;        % in Hz
f = 2;          % in Hz
T = startTime:1/fz:(startTime + duration);
S = sin(2 * pi * f * T);
mxUpdateBus(Client, T * 1000, struct('My_Bus', ...
            struct('Signal_1', S)));
```

[*SP*] mxDisplayBus

mxDisplayBus is used to instruct MAPPS to render the fields of the specified bus on the Signals Page.

Syntax and Arguments:

```
mxDisplayBus(
              struct: Matlab client,
              string: Name of bus
             );
```

Notes:

1) If the fields of the specified bus are already displayed, mxDisplayBus will have no effect.
2) If the specified bus does not exist, mxDisplayBus will fail and have no effect.

[*SP*] mxHideBus

mxHideBus is used to instruct MAPPS to remove the fields of the specified bus from being displayed on the Signals Page.

Syntax and Arguments:

```
mxHideBus(
           struct: Matlab client,
           string: Name of bus
          );
```

Notes:

1) If the fields of the specified bus are not rendered when mxHideBus is called, it will have no effect.
2) If the specified bus does not exist, mxHideBus will fail and have no effect.
3) It might be a good idea to hide buses used as temporary storage buffers in order to minimize clutter on the Signals Page.

[*SP*] mxDeleteBus

mxDeleteBus is used to purge the specified bus from the current dataset in MAPPS.

Syntax and Arguments:

```
mxDeleteBus(
             struct: Matlab client,
             string: Name of bus
            );
```

Notes:

1) Only buses created by the Matlab user can be deleted.
2) If the specified bus does not exist, mxDeleteBus will fail and have no effect.

[*SP*] mxDeleteByTime

mxDeleteByTime is used to instruct MAPPS to purge all timeslices included in the specified interval from the fields of the specified bus. The interval is inclusive of the start and end times, and is specified as millisecond offsets with respect to the start of the entire dataset.

Syntax and Arguments:

```
mxDeleteByTime(
                struct: Matlab client,
                string: Name of Bus,
                scalar: Inclusive start time of specified interval in
                        terms of millisecond offset,
                scalar: Inclusive end time of specified interval in
                        terms of millisecond offset
               );
```

Notes:

1) Only data from buses created by the Matlab user can be deleted.
2) If the specified bus does not exist, mxDeleteByTime will fail and have no effect.

[*SP*] mxDeleteByIndex

mxDeleteByIndex is used to instruct MAPPS to purge all timeslices of which the array indices fall within the specified interval from the fields of the specified bus. The interval is inclusive of the start and end indices.

Syntax and Arguments:

```
mxDeleteByIndex(
                 struct: Matlab client,
                 string: Name of Bus,
                 scalar: Inclusive start index of specified interval,
                 scalar: Inclusive end index of specified interval
                );
```

Notes:

1) Only data from buses created by the Matlab user can be deleted.
2) If the specified bus does not exist, mxDeleteByIndex will fail and have no effect.
3) If the specified start or end index falls outside the indices of the existing timeslices of the bus, mxDeleteByIndex will fail and have no effect.

[*ET*] mxGetFixations

mxGetFixations is used to retrieve fixation data for the specified subject from MAPPS. Fixation data are normalized with respect to the underlying canvas resolution. Time is specified as millisecond offsets with respect to the start of the entire dataset.

Syntax and Arguments:

```
Data = mxGetFixations(
                        struct: Matlab client,
                        scalar: Subject index,
                        scalar (optional): Start time of requested
                                           interval in terms of
                                           millisecond offset,
                        scalar (optional): End time of requested
                                           interval in terms of
                                           millisecond offset
                       );
```

with

```
struct Data
          ├─ Canvas (display) index for fixations 1..n
          ├─ Start time for fixations 1..n in millisecond offset
          ├─ End time for fixations 1..n in millisecond offset
          ├─ Normalized x coordinate [0..1] for fixations 1..n
          └─ Normalized y coordinate [0..1] for fixations 1..n
```

Notes:
1) Canvas indices correspond with the order of canvases reported by mxListCanvases.
2) The bottom left-hand corner is defined as the origin of a MAPPS canvas.
3) If output argument `Data` is not provided, the result is echoed in the Command Window of Matlab.

[*ET*] mxUpdateFixations

mxUpdateFixations is used to replace fixation data for the specified subject based on an interval defined by the 1st and last time values of the millisecond offsets provided.

Syntax and Arguments:

```
mxUpdateFixations(
                    struct: Matlab client,
                    scalar: Subject index,
                    struct: ┬─ numeric array: DisplayIndex: Canvas
                            │      indices for fixations 1..n
                            ├─ numeric array: From_MAPPS_time: Start
                            │      offsets for fixations 1..n
                            ├─ numeric array: To_MAPPS_time: End
                            │      offsets for fixations 1..n
                            ├─ numeric array: MidPointX: Normalized
                            │      x coordinates for fixations 1..n
                            └─ numeric array: MidPointY: Normalized
                                   y coordinates for fixations 1..n
                  );
```

Notes:

1) Canvas indices correspond with those reported by mxListCanvases.
2) The bottom left-hand corner is defined as the origin of a MAPPS canvas.
3) The required field names for `struct` containing the fixation data are not case sensitive.
4) In order to view the fixation data in MAPPS, select 'External' from the Algorithm menu in Display Settings on the Eye Tracking Page, and ensure that the Show fixations checkbox is selected.

Example:

To update fixation data, consisting of 2 fixations on the 1st canvas, for subject 1, use:

```
Client.Name = 'My_Matlab';
CanvasIndex = [0, 0];     % index of 1st canvas = 0
StartTime = [1.5, 2.8];   % in sec.
EndTime = [2.0, 3.5];     % in sec.
MidPointX_Normalized = [0.1, 0.6];
MidPointY_Normalized = [0.1, 0.2];

Fix = struct('DisplayIndex', CanvasIndex, ...
             'from_mapps_time', StartTime * 1000, ...
             'To_MAPPS_time', EndTime * 1000, ...
             'MidPointX', MidPointX_Normalized, ...
             'MidPointy', 1 - MidPointY_Normalized);
mxUpdateFixations(Client, 1, Fix);
```

[*ET*] mxGetRois

mxGetRois is used to request the attributes of a specific or all regions (or areas) of interest (ROI or AOI)in the current dataset in MAPPS, in terms of its name/their names, shape or type, as well as location and size data for all specified (not interpolated) frames. Time is specified as millisecond offsets with respect to the start of the entire dataset.

Syntax and Arguments:

```
ROIs = mxGetRois(
                struct: Matlab client
                );
```

or

```
ROIs = mxGetRois(
                struct: Matlab client
                scalar: ROI index
                );
```

with

```
struct ROIs
          ├─ Name of ROI 1
          │            ├─ Canvas index
          │            ├─ Name of ROI shape/type
          │            └─ Frame ─┬─ Data for frame 1
          │                      ├─ Data for frame 2
          │                      ╞═ :
          │                      └─ Data for frame n
          ╞═ :
          └─ Name of ROI n
                       ├─ Canvas index
                       ├─ Name of ROI shape/type
                       └─ Frame ─┬─ Data for frame 1
                                 ├─ Data for frame 2
                                 ╞═ :
                                 └─ Data for frame p
```

and

```
struct Frame(i)
          ├─ Time offset in millisecond
          ├─ Normalized width
          ├─ Normalized height
          ├─ (X0): Normalized x coordinate of bottom left hand corner
          │             (for 'rectangle' shape)
          ├─ (Y0): Normalized y coordinate of bottom left hand corner
```

```
│            (for 'rectangle' shape)
├─ (X1): Normalized x coordinate of top right hand corner
│            (for 'rectangle' shape)
├─ (Y1): Normalized y coordinate of top right hand corner
│            (for 'rectangle' shape)
├─ (Xm): Normalized x coordinate of center point
│            (for 'ellipse' shape)
├─ (Ym): Normalized y coordinate of center point
│            (for 'ellipse' shape)
├─ (RadiusX): Normalized length of the semi x axis
│            (for 'ellipse' shape)
├─ (RadiusY): Normalized length of the semi y axis
│            (for 'ellipse' shape)
└─ 2D normalized vertices (for 'polygon' type)
```

Notes:

1) Canvas indices correspond with those reported by mxListCanvases.
2) The bottom left-hand corner is defined as the origin of a MAPPS canvas.
3) If output argument `ROIs` is not provided, a summary of the results are echoed in the Command Window of Matlab.

[*ET*] mxGetRoiIntersections

mxGetRoiIntersections is used to retrieve region (or area) of interest intersections based on eye tracking data for the specified subject from MAPPS. Time is specified as millisecond offsets with respect to the start of the entire dataset.

Syntax and Arguments:

```
Data = mxGetRoiIntersections(
                              struct: Matlab client,
                              scalar: Subject index,
                              scalar (optional): Start time of
                                        requested interval in terms
                                        of millisecond offset,
                              scalar (optional): End time of
                                        requested interval in terms
                                        of millisecond offset
                             );
```

with

```
struct Data
          ├─ Time for intersections 1..n in millisecond offset
          ├─ Region (or area) of interest index for intersections 1..n
          └─ Name of region (or area) of interest for intersections
                1..n
```

Notes:

1) Region (or area) of interest indices correspond with the order of regions (or areas) of interest reported by mxListRois.
2) If output argument `Data` is not provided, the result is echoed in the Command Window of Matlab.

[*ET*] mxListCanvases

mxListCanvases is used to request the attributes of video canvases of the current dataset in MAPPS, in terms of their names, as well as the width and height of each canvas.

Syntax and Arguments:

```
Canvases = mxListCanvases(
                           struct: Matlab client
                          );
```

with

```
struct Canvases
          ├─ Name of canvas 1 ─┬─ Width in pixels
          │                    └─ Height in pixels
          ═ :
          └─ Name of canvas n ─┬─ Width in pixels
                               └─ Height in pixels
```

Notes:

4) If output argument `Canvases` is not provided, the results are echoed in the Command Window of Matlab.

[*ET*] mxUpdateOverlay

mxUpdateOverlay is used to instruct MAPPS to overlay the grayscale or color RGB image provided onto the specified canvas.

Syntax and Arguments:

```
Overlay = mxUpdateOverlay(
                           struct: Matlab client,
                           string: Name of canvas,
                           numeric array: Image as 2D matrix (grey-
                                          scale) or 3D matrix (RGB)
                                          with default Matlab ordering
                                          of dimensions:
                                          height x width x color,
                           boolean: Flip image vertically?
                          );
```

or

```
Overlay = mxUpdateOverlay(
                           struct: Matlab client,
                           string: Name of canvas,
                           numeric array: Image as 2D matrix (grey-
                                          scale) or 3D matrix (RGB),
                           boolean: Flip image vertically?,
                           string: 2/3 character sequence for
                                   specifying ordering of dimensions:
                                   height(H) x width(W) x color(C)
                          );
```

with

`numeric array Overlay` = 2D or 3D Matlab representation of the image passed to MAPPS

Notes:

1) The bottom left-hand corner is defined as the origin of a MAPPS canvas. To match the layout of a MAPPS canvas, it is necessary to vertically flip an image that has its upper left-hand corner defined as the origin.
2) In order to activate the overlay function in MAPPS, select 'Custom' from the Heatmap Mode menu in Display Settings on the Eye Tracking Page.
3) The range of grayscale and RGB values is [0, 255], and the most efficient storage type is `uint8`.
4) For RGB images, a fourth value [0, 255] can be added to the color dimension to specify the alpha (transparency) channel.
5) The resolution of the image does not have to match that of the specified canvas. MAPPS will stretch and/or shrink the image in the horizontal and vertical dimensions to fit the canvas.

Example:

To overlay a canvas, called say ‘Right_Display’, with a red-filled image, use:

```
Client.Name = 'My_Matlab';
width = 640;
height = 480;
Red = zeros(height, width, 3);
Red(:, :, 1) = 255;
mxUpdateOverlay(Client, 'Right_Display', Red, false, 'hwc');
```

[*CM*] mxSignOffClient

mxSignOffClient is used to disconnect the Matlab client from MAPPS, to release shared memory, and to remove the client status bar from the Signals Page.

Syntax and Arguments:

```
mxSignOffClient(
                 struct: Matlab client
                );
```