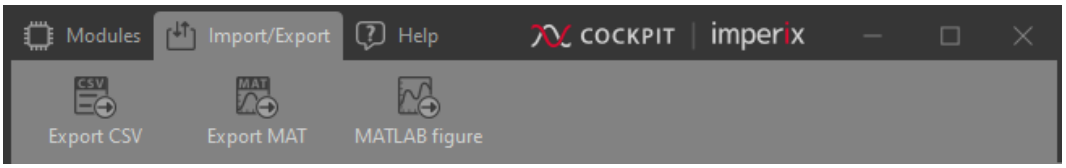
Working with MAT files exported from Cockpit

PN141 | Posted on February 27, 2024 | Updated on August 25, 2025

Mateja ILIĆ
Software Development Engineer
imperix

Table of Contents

As described in the Cockpit user guide, imperix Cockpit provides the user with the option to export monitored signals in different forms. This article provides additional information on working with data exported from Cockpit in the form of MAT files. Choosing the “Export MAT” or “MATLAB figure” options from the top bar or a context menu results in the monitored data being stored in the MAT file format.



Import/Export menu of Cockpit’s top bar

MAT is a binary file format used by MATLAB to save workspace variables, which makes it easy to load their contents and manipulate them in the MATLAB working environment. While MATLAB defines several different versions of MAT files, all MAT files created from Cockpit are in MAT-File Version 7.3, starting from Cockpit version 2024.1.

This format is based on the HDF5 standard for hierarchical storing of large amounts of data. In MAT 7.3 files, the data is compressed by default and stored in individually accessible chunks. This allows for partial saving, by appending more chunks to the file, and partial loading, by decompressing only the chunks that contain the variable to access.

Put simply, the compression and organization of MAT 7.3 files make the process of exporting from Cockpit and loading into the MATLAB Workspace more efficient in terms of time and memory in most cases compared to other file versions. To make the most of MAT files exported from Cockpit, different ways to manipulate them are listed below.

Loading MAT files

The easiest way to load a MAT file is by double-clicking on it from the MATLAB file explorer. This is equivalent to using the MATLAB `load` function with a relative file path provided.

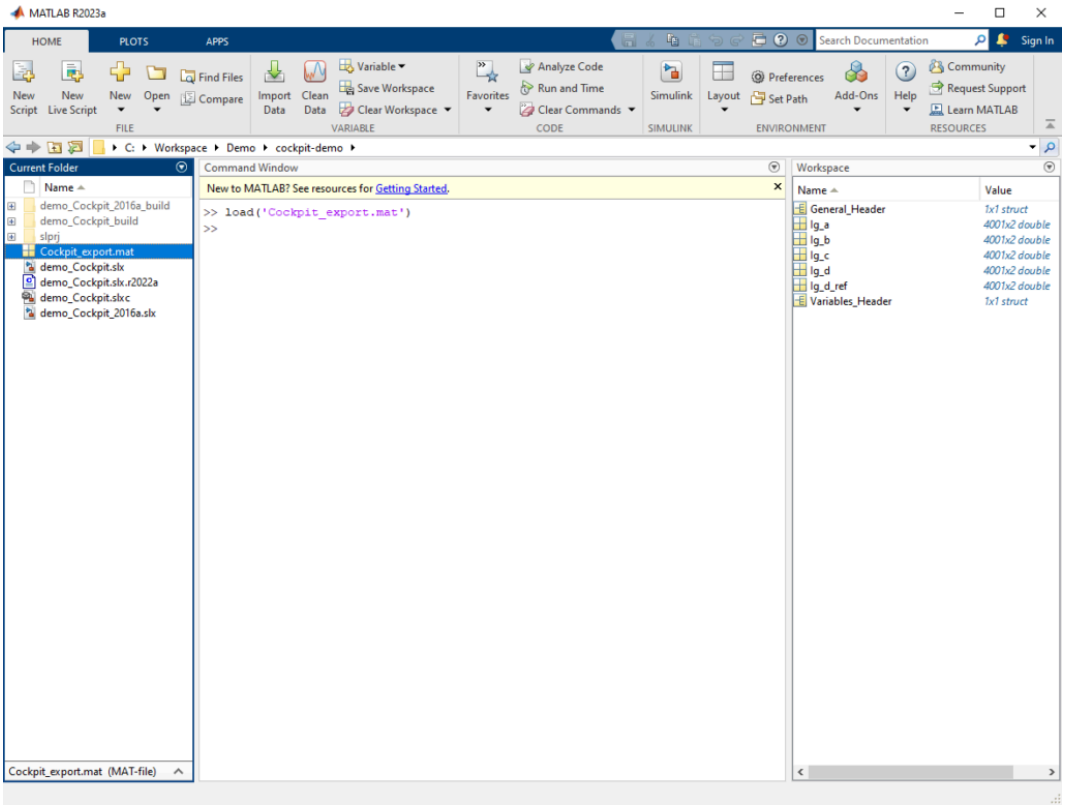


Figure 1: Loading a MAT file to a MATLAB Workspace

There are, however, more sophisticated ways to load data from MAT files. This may be useful, for example, when the file containing signals from the Cockpit Rolling Plot module after monitoring them over a long period of time. Unpacking such a file and loading all of the saved variables into MATLAB might take a while, overwhelming MATLAB's available memory. Loading such files could cause MATLAB to generate `Out of Memory` errors, become unresponsive, or crash. This can be avoided by loading only selected variables, or even parts of variables.

To work with MAT files without loading the stored data, the file should be interfaced with through the MAT-file object, created using the `matfile` function:

```
matObj = matfile('Cockpit_export.mat');Code language: Matlab (matlab)
```

Calling the `whos` function on the MAT-file object will list out the names, dimensions and memory sizes of the data contained in the file. From here, any variable can be chosen and loaded into the Workspace like so:

```
Ig_a = matObj.Ig_a;Code language: Matlab (matlab)
```

Loading parts of a given variable can be done by simply indexing it. Using the `end` keyword should be avoided so as not to inadvertently load the whole variable. If access to the last element of a dimension is needed, the following pattern can be used:

```
[nrows,ncols] = size(matObj,'Ig_b');
last_timestamp = matObj.Ig_b(nrows,1);Code language: Matlab (matlab)
```

Working with MAT files generated by Cockpit

Choosing the “Export MAT” option generates a MAT file with a structure that can be seen in Figure 1. The signals are exported as variables with the same name they had in Cockpit, in the format of a `N x 2` matrix, where `N` is the number of samples in the exported interval; the first column contains the signal timestamps, and the second the signal values. The `General_Header` structure contains the project name, export date, target IP, and other general information, while the `Variables_Header` contains information related to the visual representation of the exported signals, as plotted in Cockpit (Scale, Offset, Color…).

Plotting the exported data

After the data is saved, MATLAB’s post-processing and plotting capabilities can be used to generate the desired figure. While the initial data exploration is typically done through the MATLAB Command Window, once the desired results are obtained the commands can be transferred into a script that would be applied to all future exports. Using custom MATLAB scripts in this manner will keep the visualizations consistent across multiple experiments.

When choosing the “MATLAB Figure” export, MATLAB scripts are executed automatically after the MAT file is exported. The scripts serve to:

1. Take the exported data and recreate the signals, as they were plotted in Cockpit, in a MATLAB figure
2. Overwrite the original exported MAT file with one that contains data reshaped to a form more centered around plotting

The resulting MAT file contains essentially the same information, just in a slightly different layout. Every signal is represented with a structure that contains both the timeseries data and data necessary for recreating the exported plots in a MATLAB Figure:

```
Name: 'Ig_a'
Scale: 1
Offset: 0
Unit: '-'
Plot: 'Plot 1'
Color: '#6c4092'
Module: 'Scope'
ModuleType: 'Scope module'
Time: [4001×1 double]
Data: [4001×1 double]Code language: Matlab (matlab)
```

The MATLAB scripts used when exporting as MATLAB figure can be found in the `cockpit/matlabScripts` folder of your SDK installation. They can be used as a jumping-off point for your own custom scripts, or even modified directly. To ensure that your custom script is executed with the “MATLAB Figure” export option, make sure the name of the scripts directory and the main script, `handleCockpitExport.m` remain the same.

Saving the results as MAT files

After loading and processing the Cockpit data in MATLAB, it can also saved again in another file, or even in the same file, as demonstrated in the following example:

```
matObj = matfile('Cockpit_export.mat');

%% Saving in a separate file
Ig_d_filt = matObj.Ig_d;
Ig_d_filt(:,2) = smoothdata(Ig_d_filt(:,2)); % example processing - smoothing filter
Ig_d_ref = matObj.Ig_d_ref;
% to create a new MAT file
save("manipulated_var", "Ig_d_filt", '-v7.3');
% to add a new variable to an existing file
save("manipulated_var", "Ig_d_ref", '-append');

%% Saving in the same file
% to enable saving to the file accessed through the MAT-file Object
matObj.Properties.Writable = true;
matObj.Ig_d_filt = Ig_d_filt; % to add a new variable though the MAT-file Object

%% Updating values in a file through partial loading and saving
variables_header = data_obj.('Variables_Header');
vars_to_update = {'Ig_a', 'Ig_b', 'Ig_c'};
for i = 1:length(vars_to_update)
  if any(strcmp(variables_header.Name, vars_to_update{i}))
    matObj.(vars_to_change{i})(:,2) = smoothdata(matObj.(vars_to_update{i})(:,2));
  end
endCode language: Matlab (matlab)
```

Note that partial loading and saving in a loop, especially one element at a time, might be counter-productive, as repeated access to the file has its own overhead that can decrease the performance of your code. For large files, MATLAB documentation recommends reading and writing as much data into memory as possible at a time to avoid unnecessary overhead.

References

[1] https://mathworks.com/help/matlab/import_export/mat-file-versions.html

[2] https://mathworks.com/help/matlab/ref/matlab.io.matfile.html

[3] https://mathworks.com/help/matlab/import_export/load-parts-of-variables-from-mat-files.html