ME 477 Embedded Computing
Saving myRIO C data to a MATLAB file

The following C functions write data of types double or char to a MATLAB ".mat" file. They are included in the ME477Library.[1] Note: You must add `#include "matlabfiles.h"` to your code.

Use these functions to open a named file *on the myRIO*, and successively add any number of data arrays and strings to the file. Finally, close the file.

Open a .mat file The prototype for the open function is

```
MATFILE *openmatfile(char *fname, int *err);
```

where `fname` is the filename, and `err` receives any error code. The function returns a structure for containing the MATLAB file pointer.

A typical call might be:

```
mf = openmatfile("Lab.mat", &err);
 if(!mf) printf("Can't open mat file %d\n", err);
```

For ME 477, **always** use the file name: "Lab.mat". Notice the use of pointers.

Add a matrix The prototype of the function for adding a matrix to the MATLAB file is

```
int
matfile_addmatrix( MATFILE *mf,
                   char *name,
                   double *data,
                   int m,
                   int n,
                   int transpose);
```

where `mf` is the MATLAB file pointer from the open statement, `name` is a char string containing the name that the matrix will be given in MATLAB, `data` is a C data array of type `(double)`, `m` and `n` are the array dimensions, `transpose` takes value of 0 or 1 to indicate where the matrix is to be transposed.

For example, to add a **1-D matrix** the call might be

```
matfile_addmatrix(mf, "vel", buffer, IMAX, 1, 0);
```

Or, to add a **single variable** the call might be

```
double Npar;
Npar = (double)N;
matfile_addmatrix(mf, "N", &Npar, 1, 1, 0);
```

Again, notice the use of pointers, and the cast to `double`.

Add a string The prototype of the function for adding a string to the MATLAB file is

```
int
matfile_addstring( MATFILE *mf,
                   char *name,
                   char *str);
```

where `mf` is the MATLAB file pointer from the open statement, `name` is a char string containing the name that the matrix will be given in MATLAB, and `str` is the string.

For example, to add a **string** the call might be

```
matfile_addstring(mf, "myName",  "Bob Smith");
```

Close the file After all data have been added, the file must be closed. The prototype of the function for closing the MATLAB file is

```
 int matfile_close(MATFILE *mf);
```

where `mf` is the MATLAB file pointer from the open statement.

For example, to close the MATLAB file the call might be

```
matfile_close(mf);
```

Example Code Putting these ideas together:

```
mf = openmatfile("Lab.mat", &err);
 if(!mf) printf("Can't open mat file %d\n", err);
matfile_addstring(mf, "myName",  "Bob Smith");
matfile_addmatrix(mf, "N", &Npar, 1, 1, 0);
matfile_addmatrix(mf, "M", &Mpar, 1, 1, 0);
matfile_addmatrix(mf, "vel", buffer, IMAX, 1, 0);
matfile_close(mf);
```

Transfer file to MATLAB After the `Lab.mat` file has been created, it can be transferred directly to MATLAB.

1. In the right pane of the Remote Systems Explorer perspective, select `172.22.11.2`, and click "Refresh information of selected resource".
2. Double click on the MATLAB data file: 172.22.11.2→Sftp Files→My Home→`Lab.mat`
3. The `Lab.mat` file will be opened in MATLAB on your laptop. Use MATLAB's `whos` command to list all of the named variables in the workspace.
4. In MATLAB navigate to a convenient folder on your laptop. Then, issue the "`save('Lab.mat')`" command to save the MATLAB workspace locally.

   The file can later be opened from a MATLAB script, using the command `load('Lab.mat')`, for plotting or analysis.

[1] http://www.malcolmmclean.site11.com/www/MatlabFiles/matfiles.html