An Example of DDS I/O in Octave and Matlab

Copy-paste the following code in **dds_str.m** file (read DDS definition “defname” from dictionary “fname” as a text string):

```
function valstr=dds_str(defname,fname)
  [status,valstr]=system(["perl -e 'while (<>) {if ($_=~m/
(^\\s*",defname,"\\s*=\\s*)(.*)$/){$res=$2};} print $res;' <
",fname]);
```

and this code into **dds_num.m** file (read DDS definition “defname” from dictionary “fname” as a number):

```
function val=dds_num(defname,fname)
  [status,valstr]=system(["perl -e 'while (<>) {if ($_=~m/
(^\\s*",defname,"\\s*=\\s*)(.*)$/){$res=$2};} print $res;' <
",fname]);val=str2num(valstr);
```

Don't put line breaks inside the double quoted strings.

Alternatives can be coded using awk, grep, or Octave/Matlab internal I/O but I prefer Perl. Note the above function can read even malformed DDS definitions (e.g., those containing a space before =)

Save the above files in the current directory (where you will subsequently run Octave from)

Now assume we have a dictionary 'sampledict' defining a 3D fcube with sample type as float4 (big-endian ieee float) Here we demonstrate how to create a new binary, and read and modify the dictionary file.

Run octave. The following command will print axis names (text in italic is Octave output; where output is not required it can be suppressed by appending a semicolon to the command)

```
dds_str(“axis”,”sampledict”)
ans = z y x
```

Type

```
nz=dds_num("size.z","sampledict");
ny=dds_num("size.y","sampledict");
nx=dds_num("size.x","sampledict");
```

dds_str and dds_num can be used for reading any alphanumeric definitions from a DDS dictionary.

Now we will create an Octave array to hold our output:

```
v=zeros(nz,ny,nx);
```

Note that Octave and Matlab store data by columns, like Fortran does and is expected by the DDS. At this point we populate v() with our data – for this simple example we will assign a value 100 to a sub-matrix

```
v(nz/2:nz*3/4,:,:)=100;
```

Now we will create a new binary file, write data to it and associate it with the dictionary.

This command creates an output binary of big-endian ieee floats (float4 in DDS notation; for little-endian float4x use “ieee-le”)

```
vf=fopen("data_@","w","ieee-be");
```

The DDS sample type for our original data can be found out using e.g. “dds_in < sampledict” command.

Now write the data

```
fwrite(vf,v,"float")
ans =  25000
```

and close the file

```
fclose(vf);
```

In order for the DDS to know where the new binary is, we will need to append a new “data=” definition at the end of the file. This can be achieved by e.g.

```
system("echo data= data_@ >> sampledict");
```

The above command can be used to write any new definitions or update old ones in a dictionary file.

For reading a binary, “r” mode should be used with fopen and the call to fwrite substituted with

```
[v, count]=fread(vf,[nz,ny*nx],"float");
```

A sample DDS dictionary for an arbitrary fcube can be used as a template for creating new DDS files.

That's it.

For more information on the Data Dictionary System (DDS) go to http://www.freeusp.org/DDS/index.html

Contact musa'AT'maharramov.com if you have any questions or need help.