HOW TO GENERATE A DLL FROM A MATLAB FUNCTION SCRIPT TO RUN IN LABVIEW

1. Make sure you have:
   a. MATLAB compiler
   b. C/C++ compiler (eg: Open Watcom http://www.openwatcom.org/) Most people can afford to buy MS Visual C/C++

2. Make sure your MATLAB compiler and C/C++ compiler are running: Try mathworks technical note 1621 (http://www.mathworks.com/support/tech-notes/1600/1621.shtml). This note should give you enough information about how to setup your MATLAB compiler and C/C++ compiler.

3. The problem of generating a dll from MATLAB to work in LabVIEW is that the MATLAB compiler requires special data types (e.g. mxArray). It is therefore required to write a special 'wrapper function' which interfaces the MATLAB generated source code with a more 'standard' ANSI C code. The following example illustrates the procedure. The code following has been implemented using MATLAB 5.3, MATLAB Compiler 2.0, Open Watcom 1.0, and LabVIEW 6.1.

   a. Write the following m-code function and save as foo.m:

```
function y = foo(x)
y = 2*x
```

   b. Compile into 'C' code: Type the following command in the MATLAB prompt:

```
>>mcc -t -L C -W lib:foolib -h foo.m
```

   This will generate the following files:

   foo.c: contains the implementation of foo (Mfoo) and the interfacing functions (mlfFoo, mlxFoo)

   foo.h: contains the prototypes of mlfFoo and mlxFoo

   foolib.c: contains the implementations of foolibInitialize and foolibTermintate, necessary to initialize and terminate mlfFoo

   foolib.h: contains the prototypes of mlxFoo, foolibInitialize and foolibTerminate.

   Foolib.exports: contains the symbols to export in the dll.

c. Create foo_wrapper.c. This is the wrapper function which will allow the use of the C implementation of foo.m. The code is listed and commented below.

```
/* This file foo_wrapper.c */
#include "matlab.h"
#include "foodll.h"
#include "matrix.h"

//main wrapper function definition
double wrapper_main(double *in1){

   //declare variable to deliver result
   double out;

   //Create two pointers of mxArray type to store inputs and outputs
   mxArray *in1_ptr, *out1_ptr;

   //Allocate input pointer to a 1 by 1 double, real matrix
   in1_ptr = mxCreateDoubleMatrix(1,1,mxREAL);

   //Move the data from the input to the pointer
   fill(mxGetPr(in1_ptr),in1,1);

   //Initialise foo implementation
   foolibInitialize();

   //Pass values to mlfFoo and receive in mxArray type variable
   out1_ptr = mlfFoo(in1_ptr);

   //Terminate foo implementation
   foolibTerminate();

   //Move from mxArray type to double type
   fill(in1,mxGetPr(out1_ptr),1);

   //move data to output variable
   out = *in1;

   //Return value
   return(out);
}

void fill(double *out, double *in, int size){
   //This function moves data from one type to another
   int i;
   for(i=0;i<size;i++)
      out[i] = in[i];
}
```

d. Create foodll.h which contains prototypes of functions in foo_wrapper.c. The code is listed below.

```
//This file foodll.h
double wrapper_main(double *in1);
void fill(double *out, double *in, int size);
```

e. Add entry point to foolib.exports: add the following line to the file foolib.exports:

```
wrapper_main.
```

f. Build using the following command in MATLAB:

```
>>mbuild –link shared foo_wrapper.c foo.c foolib.c foolib.exports
```

4. You have now generated a file called foo_wrapper.dll, which contains the function wrapper_main.c among others. Use this function making sure the input (arguments) and output (return value) are of 8-bit double type.

5. Additional information can be found in:
   a. Mathworks solution number 25176: http://www.mathworks.com/support/solutions/data/25176.shtml
   b. http://www.mathworks.com/support/solutions/data/23932.shtml
   c. http://www.mathworks.com/support/solutions/data/27671.shtml
   d. National Instruments Application Note 087: http://zone.ni.com/devzone/conceptd.nsf/webmain/5CF9A9FFD774028586256869005FF2ED?opendocument