Astrodynamics Standards Shared Library

DllMain

Version 9.4

May 2024

Contents

1. INTRODUCTION 1

2. PREREQUISITES...1

3. GETTING STARTED 1

4. USER'S OWN INPUT DATA 1

1. Introduction

DllMain creates a static global data set containing all of the data pointers and function pointers needed by all other Astrodynamics Standards libraries. This library uses the static global data set to communicate among the other libraries.

If you are on Windows, the shared library files will end in ".dll". For example, "BatchDC.dll". If you are on Linux, the shared library will begin with "lib" and end in ".so", and will be all lowercase. For example, libbatchdc.so.

2. Prerequisites

There are no prerequisites for using DllMain.

3. Getting Started

To get started, please read the README.txt file that came in the root directory of your distribution. In addition to an overall description contained in the distribution, it has a description of a "**wrapper**".

To get started with **DllMain**, there is a "wrapper" specific to DllMain, under the **SampleCode** directory. Under your language of choice, you will see a "**DriverExample/wrapper**" subdirectory. The files under this directory will have all the Application Programming Interfaces (APIs) available. For DllMain specific APIs, you should see a source file labelled with "DllMain" in the file name. This will be where you will find all the APIs for that specific library. The "DriverExample" directory will also contain several examples of applications that should run by simply running the runExample.bat or runExample.sh script. You can use these examples as a starting point for building your application.

If you do not see your programming language under "SampleCode", look in the HTML documentation for the APIs. Open a browser to the "Documentation/APIDocs/index.html" file. This document will show all the APIs regardless of programming language.

The Astrodynamics Standards libraries should work with any language capable of using Dynamic Link Library (on Windows) or Shared object (on Linux) files.

4. User's own input data

Whenever the user wants to enter and later retrieve his own input data in an input text file, he can use the predefined input card "AS_MOIC" format to fulfil that. This input card allows up to 128 numeric data fields. It's a free format and the only requirement is that data fields are separated by commas or white spaces. To retrieve the input data, the user needs to call the **DllMainLoadFile**() first and later call the **GetMOICData**() to retrieve the expected input data.

Example:

```
// name of the input text file
string dllMainInputFile = "as_moic_test.inp";

// there are 9 numeric data fields the user enters via the AS_MOIC card
string myInputString = "AS_MOIC 1.123, 2.234, 3.4545, 4.656, 5.15,6.456, 7.876, 8.887, 9.165";

// store the input data in the input text file
File.WriteAllText(dllMainInputFile, myInputString);

// load the input text file
int errCode = DllMainWrapper.DllMainLoadFile(dllMainInputFile);
Assert.AreEqual(0, errCode);

// the input AS_MOIC contains 9 fields
double[] xa_moic = new double[128];
```

```
// retrieve the input data stored in the AS_MOIC card in the input text file
DllMainWrapper.GetMOICData(9, xa_moic);
Assert.AreEqual(1.123, xa_moic[0]);
Assert.AreEqual(2.234, xa_moic[1]);
Assert.AreEqual(3.4545, xa_moic[2]);
Assert.AreEqual(4.656, xa_moic[3]);
Assert.AreEqual(5.15, xa_moic[4]);
Assert.AreEqual(6.456, xa_moic[5]);
Assert.AreEqual(7.876, xa_moic[6]);
Assert.AreEqual(8.887, xa_moic[7]);
Assert.AreEqual(9.165, xa_moic[8]);
```