Frequently Asked Questions

Version 2.8

February 26, 2010

MLDesign Technologies, Inc.
2130 Hanover St
Palo Alto, CA 94306

support : www.mldesigner.com/support
http : www.mldesigner.com

MLDesign Technologies

Chapter 1

Frequently Asked Questions

This chapter is designed to help you find quick and short answers to your questions. All these topics are covered in detail in the MLDesigner user manual which can be opened by clicking the **Help** button in the top right corner of the GUI and selecting the option **Search Index**.

1.1 General Questions

Do I need to learn Ptolemy and BONeS to be able to design with MLDesigner? Go to Answer 2.1.

1.2 Error Messages and Their Most Common Causes

1. Cannot create library ”xxx” to location ”file:/user/xxx/yyy/zzz/xxx.mml”. Maybe you have no write permission. Cannot create model ”xxx”. You must create the library ”yyy” first. Go to Answer 1.
2. Compilation error , unable to load the primitive Dynamic. Go to Answer 2.
3. ”make: fatal error in reader: /export/home/mldesigner/mk/ config-default.mk,line 59: Unexpected end of line seen” Go to Answer 3.

1.3 Segmentation Faults

What are the most common causes of segmentation faults that cause MLDesigner to crash? Go to Answer 2.3.

1.4 Data Structures

1. When must a data structure be deleted and when is it a clone?
2. What is a memory leak?

Go to Answer 2.4.

1.5 Load Mode

What is the difference between Load Mode Dynamic and Load Mode Permanent? Go to Answer 2.5.

1.6 Plotting Systems

How do I make data available to, or save data for analysis using external programs? Go to Answer 2.6.

1.7 Setting Environment Variables

1. How do I change the environment variables so I can use another Editor to debug or change source code in my primitives? Go to Answer 2.7.
2. How do I change the `MLD_USER` environment variable. Go to Answer 2.7.
3. How do I change the environment variables so I can use an external debugger? Go to Answer 2.7.
4. Is setting the `Working Directory` entry the same as setting the `$MLD_USER` environment variable? Go to Answer 2.7.

1.8 ddd debugger and Red Hat

I am running MLDesigner under Red Hat and I am unable to debug using the standard debugger **ddd**. Go to Answer 2.8.

1.9 Linked Objects

Why must I restart MLDesigner before changes made to Linked Object take effect? Go to Answer 2.9.

1.10 Shared Libraries

What is the environment variable `MLD_SHARED` used for and how do I set this variable? Why are there invalid modules in my Shared Libraries? Go to Answer 2.10.

Chapter 2

Answers to FAQ's

2.1 Answers for the General Questions

No but it is necessary to bear in mind that the Ptolemy base type `int` is different to any **data structure** type including those like `Root.Integer`. It is only possible to connect ports of the same type. Back to Question 1.1.

2.2 Error Messages and Their Causes

1. This error message normally occurs when you do not have permission to write to the target directory or to one of the directories higher in the hierarchical structure. Another reason could be caused by a file that is write protected in the target directory when you have checked the option "Overwrite existing Files" in the appropriate Import dialog window. Back to Question 1.2.
2. There are a number of reasons for this error message.

   - Firstly your primitive contains syntactical errors and can not be compiled by the C++ compiler. In that case you will get an error dialog ("Compilation failed."), when you compile the primitive in MLD. You should open the primitive editor, try to compile it and correct the reported errors until successful compilation is confirmed.
   - Secondly a primitive with the same name already exists in the same Domain.
   - Another reason could be the primitive was already loaded with Load mode set to Permanent. If a Primitive is loaded permanent it is not possible to make any changes to its source code and then recompile the primitive. You must shutdown MLDesigner and restart before the changes take effect. You only need to load a primitive as permanent if you want it to be inherited by another primitive. A primitive that is a child of another primitive can only be recompiled if the parent is loaded as permanent.

   Back to Question 1.2.
3. This error is often caused by compiler incompatibility. What version of make do you use? You need gmake 3.74 or later. Please look at the section on system requirements in the app. **??**. Back to Question 1.2.

2.3 Segmentation Faults / System Crashes

The most common reason for system crashes with MLDesigner is programmer error in user defined primitives. When a segmentation fault occurs you should first check to see if your system contains user defined primitives. Often the crash is caused by inconsistencies in memory allocated to output datasets. The next step is to run the simulation **extern** possibly using an external debugger to debug the system. External simulations occur outside of the MLDesigner environment. See the MLDesigner manual index for more information on debugging. Back to Question 1.3.

2.4 Data Structures

1. **When to clone a particle** Generally you don't have to clone particles. In sending particles over the ports you can use operators like << or =. For data structures the situation is more complicated, because you have to take into account how it is instantiated as a data structure and how it is used.

   When to create a new particle When you instantiate a new data structure for direct usage (not to be used as parameter in a method that takes `const Type*` as parameter - methods of setType() kind), you have to use the method `DsHandler::makeNewStructure()`.

   When is a particle a clone? When you set a data structure to a Memory, Event or a DataStructMember, the parameter is cloned inside the method so there is no need for a new created object.

   When to delete a data structure: When you get a data structure using the method `DsHandler::makeNewStructure()` be sure to delete it to avoid overloading memory. Every new data structure created using the `clone ()` method must be deleted. The exception is when you place it on an output port. DataStructParticles take care to delete the data structure when the last reference to it is deleted. Do not delete `const` pointers to Type or class `derivedfrom`, returned by some methods. These are references to the class members and the object takes care of deleting them. On the contrary, `non const` pointers to Types must be deleted if you are not in the exceptional case previously mentioned.

   Do not delete particles when As an exception, don't delete the `non const Type*` returned by DataStructMember's methods `getData ()` or `fieldWithName ()` of the DataStructure class, or every time when you write or read a Memory, using `writeMemory()`
2. A **Memory Leak** occurs when new memory is allocated dynamically and never deallocated. In C++ new memory is created by the `new` operator and deallocated by the `delete` or the `delete []` operator. Memory leaks accumulate over time and can crash the program.

Back to Question 1.4.

2.5 Load Mode

While creating a new primitive model component, you can define its load mode. By selecting the load mode Dynamic, a shared library containing the primitive code is created on loading. This shared library is linked dynamically to MLDesigner so that you can reload the primitive any time. All changes made to the primitive are effective immediately. Conversely, by selecting load mode Permanent, the primitive is linked to MLDesigner on loading as it would be with a built-in

primitive. All changes made to a primitive with load mode permanent only take effect after closing and restarting MLDesigner. Back to Question 1.5.

2.6 Plotting Systems

1. There are a number of ways to display or handle datasets and results of simulations.

   - You may use SatLab to display data. (see Demos/DopplerIR, you need to start SatLab before running the demo!). It uses a Tcl script $MLD/MLD_Libraries/DE/Contrib/WiNeS.tcl.
   - You may save your data to one or more files and postprocess/display it with another tool of your choice. You could for example use a **DEPrinter** primitive to dump data to a file, then use GnuPlot, Matlab or even a spreadsheet program like StarCalc (within StarOffice) to display the information.

Back to Question 1.6.

2.7 Setting Environment Variables

Change the Default Editor

To define the editor you prefer to use when working with source code it is no longer necessary to set the environment variables. A dialog is available where a variety of setting can be changed. To define which editor you prefer click the **Settings** option in the main menu. Expand the tree by clicking the **Primitive Source Editor** item as shown in fig. 2.1. The following options are available:

1. **Use external editors with xterm.** Some editors open in a terminal and others not. Often the terminal is not required and only gets in the way. To stop the xterm from opening with your editor, click the check box to remove the tick. If the editor cannot open without a console, you will get an error message in the MLDesigner console when attempting to open a source file. In this case make sure you have a tick in the check box. Try open `vi` without a tick in this check box or set the editor variable to `emacs` in the **$EDITOR** input field.
2. **Internal.** This points to the built-in editor installed with MLDesigner .
3. **$EDITOR** This radio button sets the default editor to the one you have defined by setting your environment variable. i.e.,

   For **sh** and **bash** command shells:

   ```
   export EDITOR=emacs
   ```

   For **csh** and **tcsh** command shells:

   ```
   setenv EDITOR=emacs
   ```

   You must close and restart MLDesigner before changes take effect.
4. **User-defined** Has the same effect as the $EDITOR radio button except you do not have to close and reopen MLDesigner to get the new setting to work. You must only change the

editor name. The `@$` are variables that instruct the editor you choose to open the source code of the open primitive in your Model Editor Window.

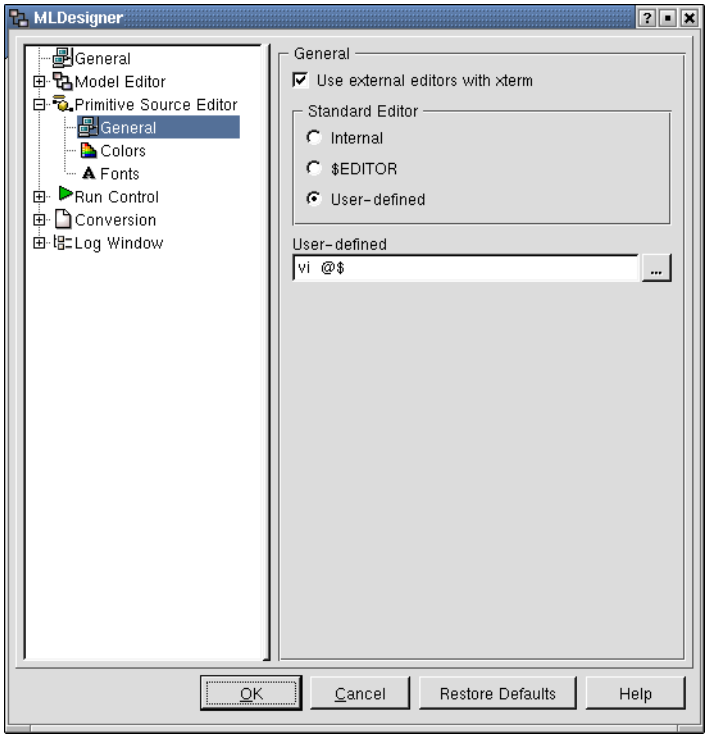

Figure 2.1: Generated hypertext documentation

■ **NOTE:** The built-in editor is used as default editor in cases where compile errors occur. The reason is that you can highlight errors in the built-in editor error console and the cursor will be automatically placed in the correct line of the source code editor.

Back to Question 1.7.

Set the `MLD_USER` variable

It is possible to set your `MLD_USER` environment variable to point to a project library or external library. Lets assume you want to work on a project called `MLD.project`. This project is the $MLD_USER directory of another user. You want to access the systems and share libraries that exist in the other user's environment. Enter the following command where you would normally open MLDesigner:

- for **bash** or **sh** shells

```
export MLD_USER=/home/user/MLD.project
```

- for **tcsh** and **csh** shells

```
setenv MLD_USER /home/user/MLD.project
```

You could also do the same locally on your own computer if you wanted to separate libraries and projects. You can create a new directory `/MLDProject` in your home directory. Set your `MLD_USER` environment variable to point to the new directory. When you open MLDesigner again you will see the tree view with MLDesigner libraries and no user libraries. You could then create a top level library with read and write rights for a workgroup on your network.
Back to Question 1.7.

Use an External Debugger

Depending on the type of shell you are using, enter one of the following at the prompt:

```
export MLD_PREBIN=ddd (sh and bash)
```

or

```
setenv MLD_PREBIN ddd (csh and tcsh)
```

The **ddd** entry refers to the ddd debugger supplied with every Linux package and can be replaced with a call referring to your favorite debugger.
You must now start MLDesigner as normal. The debugger you have chosen will start. If you are using **ddd** proceed as follows:

- Click the **View** menu and choose **Command Tool**.
- Click **Run** to start MLDesigner . This can take longer than normal.
- You can now work as normal with MLDesigner .

Back to Question 1.7.

Differences Between `$MLD_USER` and Working Directory

`$MLD_USER` and **Working Directory** are not related to each other at all. The variable `$MLD_USER` defines where MLDesigner looks for user defined models and libraries where the option **Working Directory** defines which directory is the working directory for the command console window after starting MLDesigner. Both these topics are explained in the MLDesigner manual which can be opened by clicking on the Search Index option of the Help menu found in the top right corner of the GUI. Find the index entries *Command Console* or *Working Directory* to get more detail on the topic.
Back to Question 1.7.

2.8 Using ddd Debugger under Red Hat

It is not possible to debug MLDesigner using **ddd** because Red Hat was developed completely using GCC 2.96 contrary to the recommendations of the Free Software Foundation (see http://www.fsf.org/software/gcc/gcc-2.96.html).

As a result ddd compiled under GCC 2.96 needs a symbol `dynamic_cast_2` defined in `libstdc++-libc6.2-2.so.3`.

With a SuSE system compiled using GCC 2.95 running under Red Hat, the system crashes when accessing the system internal `libstdc++-libc6.2-2.so.3`. This is apparently caused by a change in the library interface.

At the moment there are no solutions to the problem and we can only suggest you to use **gdb** or **xxgdb** as an alternative debugger.
Back to Question 1.8.

2.9 Linked Objects

Files containing object code must be loaded before the primitive that calls the external functions in the object file is compiled. All changes made to the external library will only be actualized when MLDesigner is shutdown and restarted. The reason is that it is not possible to delete the relevant primitive from memory without conflicts arising in the kernel's reload mechanism. The result would be a primitive containing a mixture of new and old code.
Back to Question 1.9.

2.10 Shared Libraries

The Shared Libraries directory was introduced to make it easier for design teams to exchange models and work on group projects.
It is possible to develop a library within the environment specified by the `$MLD_USER` environment variable and then move this library to the shared environment specified by the `$MLD_SHARED` environment variable. However, since the `$MLD_USER` variable is dynamic (because it can be different for every user) the following prerequisite applies.

- All modules and files needed by systems in the library must be located within this library, or in a location that never changes, such as the directory to which $MLD_SHARED and $MLD point.

The reason for this is that the `systemName.mml` file contains references to all model elements needed for the system to function. If these variables change then MLDesigner will not be able to locate the missing model elements.
Back to Question 1.10.