Specification

MLCad specific extensions to the Ldraw commands

MLCad INI-File

Interfaces for external applications

1. CONTENTS

2. GENERAL NOTES

This documentation is always under development. Please check the actual documentation on http://www.lm-software.com/mlcad regulary.

3. LICENSE AGREEMENT

The commands described in this document are special extensions to the Ldraw command syntax. Other developers may include the same functionality in their own applications to be compatible to MLCad.

A software may be called MLCad compatible if all commands listet in this document have been implemented. If only parts of the commands are implemented in a third party software, its a good idea to clearly state the commands implemented, or the commands not implemented. You should also meantion the version of MLCad you are compatible to.

If this document is referenced, the following hyper link should be used:
http://www.lm-software.com/mlcad

This specification document shell not be provided to third parties on any other location than the web site mentioned above neither electronically nor on paper, without written acceptance from the author of this document. This restriction shell make sure that only the latest specifications are used.

The use of this commands in ldraw models or part files is free and unrestricted and does not require any reference or notice.
The license is valid for third party application developers implementing MLCad command extensions only.

4. CONDITIONS

This document describes the commands available in MLCad V2.12 and above. Some of the commands are available beginning with higher versions only, in this case this is indicated in the description of the command.

5. DOCUMENT HISTORY

Version	*Release Date*	*Changes, Comments*
V1.0	14.12.2002	First document released
V2.0	24.06.2004	Updated description of the buffer exchange command Added arrow and plate pseudo parts Added Hide command New sections for MLCad.ini Smaller corrections

6. MLCAD SPECIFIC LANGUAGE EXTENSIONS TO DAT/LDR

6.1 View Rotation Commands

The rotation view commands are partially taken over from LDLite and have been extended to allow more flexibility. Generally they are an extension to the plain STEP command of Ldraw. Drawing stops after executing the command. If you want to make the view rotation visible a plain STEP command should be used immediately before the rotation step command.

There are four different rotation step commands:

6.1.1. Relative Rotation Steps

Relative rotation steps are based on the actual angles of the individual viewing areas. The model will rotated by the specified rotation angles relative to the current view angle.
That means if a view shows the model from it's front side, and the rotation step will rotate the model by 90° clockwise then after executing the step command you can see the model from the right side. If the current view angle is top then after executing the command, you will still see the model from top, but rotated 90° clockwise.

The syntax for this command is:

0 ROTSTEP <x-angle> <y-angle> <z-angle> [REL]

The keyword REL is optional but should be specified for clearness.

x-angle, y-angle and z-angle are the individual rotation angles for the different axes in degree (-360 to 360).

6.1.2. Additative Rotation Steps

This kind of rotation step turns the model by the specified angles, taking the current view angle into account. The command can be used to continuously rotated the model by specific angle. For example if this command is executed four times on a front view, and the model is rotated 90° clockwise on the y-axle then you will see the model from each of the four sides.

The syntax for this command is:

0 ROTSTEP <x-angle> <y-angle> <z-angle> ADD

x-angle, y-angle and z-angle are the individual rotation angles for the different axes in degree (-360 to 360).

6.1.3. Absolute Rotation Steps

The last type of rotation steps is similar to relative rotation steps, but it ignores the current view angles so that after executing this command each view area will show the same image.

The syntax for this command is:

0 ROTSTEP <x-angle> <y-angle> <z-angle> ABS

x-angle, y-angle and z-angle are the individual rotation angles for the different axes in degree (-360 to 360).

6.1.4. Rotation End Step

This command returns to the original viewing angles of the individual views.

The syntax for this command is:

0 ROTSTEP END

6.1.5. Matrix Calculation Of Rotation Steps

For each rotation step command (except END) the rotation matrix is computed by the following formula:

wx, wy and wz are the angles in radiants

s1 = sin(wx) s2 = sin(wy) s3 = sin(wz)
c1 = cos(wx) c2 = cos(wy) c3 = cos(wz)

a = c2 * c3
b = -c2 * s3
c = s2
d = c1 * s3 + s1 * s2 * c3
e = c1 * c3 - s1 * s2 * s3
f = -s1 * c2
g = s1 * s3 - c1 * s2 * c3
h = s1 * c3 + c1 * s2 * s3
i = c1 * c2

The resulting view-angle to be used by each individual view is computed in the following way, where "newmatrix" is the new matrix to be used by the view, <rotation matrix> is the matrix computed for the command. Multiplication is the standard way for multiplying matrixes.

For relative rotation steps the new view matrix is:
newmatrix = <view default matrix> * <rotation matrix>

For additative rotation steps the new view matrix is:
newmatrix = <current view matrix> * <rotation matrix>

For absolute rotation steps the new view matrix is replaced with the rotation matrix:
newmatrix = <rotation matrix>

6.2 Background Command

The background command is used to display a picture in the background of the view. The picture itself will be always behind the model, so that the model is fully visible to the viewer.

MLCad displays the image after executing the command. Another background command replaces previous background commands.

The syntax for this command is:

0 BACKGROUND <filename>

<filename> is the name of the image file. The filename can include a path name. It is recommended to set the file name in quotations marks (“). Using quotation marks also allows the use of blanks in the filename and thus avoids problems with opening the file.

6.3 Buffer Exchange Command

The buffer exchange command allows the user to create animation like instructions. It more a trick to get building instructions like the real once.

The buffer command either stores an image or retrieves a stored image. The image is taken from the actual view and when restored put in there. Es an example imagine you want to display where a motor has to be put on the chassis of a car. So you first create a building step with the motor exposed, possibly with an arrow to the point where the motor should be placed and in the next picture the motor should be there where it is supposed to be. This effect can be done as follows:
You first create the model completely without the motor. Then you save an image of the current view, and after that you put the motor above the model. When the user continues drawing you retrieve the previously saved image and redraw the motor in it’s final position.

This command is ignored if the model containing this commands is used as a sub-model.

With the help of the buffer command you can save up to eight independent images

The syntax of this command is:

0 BUFEXCHG <A-Z> STORE | RETRIEVE

<A-Z> the letter identifier for the storage to use
STORE .. to save the image or
RETRIVE .. to load the image back to the view

6.4 Ghost Commands

Ghost commands are a special form of other commands. Any MLCad or Ldraw command can be a ghost command. To make a command to a ghost command simply put “0 GHOST” at the beginning of the line.

So what’s the deal with this type of command?
MLCad displays ghost commands only if the model containing the command is the main model. If the model is a sub-model of another or simply a part included into a model the ghost lines are simply ignored. They are useful in connection with the buffer exchange command and allow to show detailed assembly instructions of a part when it’s viewed by its own, but if it’s included as a part these steps will not be shown.
Without this ghost command you would see things between buffer exchange commands uncontrolled since buffer exchange commands are also ignored in sub-models.
You will find some examples for the use of this command on http://www.lm-software.com/mlcad

The general syntax of this command is:

0 GHOST <other ldraw or mlcad command>

6.5 Group Commands

For simpler editing purposes MLCad allows the definition of groups, where two ore more parts are put together in a single virtual part. You can nearly everything with such a group as you can do with a normal part, except single part modifications are not possible. To keep the information about grouping after reloading a saved model, MLCad uses the group command to identify a group.

The syntax of this command is:

0 GROUP <n> <name>

where <n> is the number of commands belonging to this group and <name> is the name for this group.

The commands belonging to the group are identified by the MLCAD BTG command.

You must not use recursive groups!

6.6 Group Mark Commands

Group mark commands are used to identify that a part belongs to a group. The line following this command is selected to be included into the group identified by the BTG command.

Syntax:
0 MLCAD BTG <group name>
x ….

<group name> is the name of the group to which the following line belongs to.

The line following this statement is put into the specified named group.

6.7 Hide Commands

Minimum MLCad Version required: 3.10

Hide commands are used to temporarily hide parts during editing. The command has no effect to the viewing mode and will be ignored in this mode.

Parts can be hidden or made visible from certain menu options within MLCad.

Syntax:
0 MLCAD HIDE <other ldraw or mlcad command>

6.8 Skip Section Commands

For future use MLCad already includes the functionality to skip a certain section of the file in a controlled manner. Therefore two commands are defined which identify the begin and the end of the section to skip:

0 MLCAD SKIP_BEGIN
…
0 MLCAD SKIP_END

Everything between the two lines including the SKIP commands themselves are not loaded into the memory when reading the file.

In the future several commands will be added which will be processed in MLCad in a special way. When the model is then saved replacement commands for none MLCad compatible viewers are written inside the skip section.

6.9 Rotation Point Commands

MLCad supports rotation points. A rotation point is the virtual zero point for a rotation of a part or primitive. The following commands define such rotation points, and store actual rotation point settings for a model.

6.9.1. Rotation Point Definition

0 ROTATION CENTER <x> <y> <z> "<Name>"

<x>, <y> and <z> are the coordinates of the rotation point in model space.
<Name> is a unique name for the rotation point which is enclosed by two quotion marks.

A file may contain multiple rotation points.

6.9.2. Rotation Point Setting Information

Everytime MLCad saves a model file, it also stores rotation point information with it. Since the user may define multiple rotation points and can select between them and some predefined points, MLCad stores the current user selection with the following command:

0 ROTATION CONFIG <Rotation point ID> <Visible Flag>

<Rotation point ID> values smaller or equal than 0 are predefined rotation points:
0 .... Part origin
-1 .... Part center
-2 .... Part rotation point
-3 .... World origin

Values higher than 0 are custom defined rotation points where 1 means the first defined point, 2 the second ... Rotation points themself are stored in order. That means if <Rotation point ID> is 2 than the second 0 ROTATION CENTER command is used.

<Visible Flag> set to 0 if the rotation point is not displayed in an editor application, or 1 if it should be displayed.

6.9.3. Reservered Rotation Related Commands

The following commands have been reserved for future use:
0 ROTATION AXLE

This command is reserved for future definition of a rotation axle.

6.10 Spring Pseudo Part

Minimum MLCad Version required: 3.00

Syntax: 0 MLCAD SPRING <color> <pos> <rot> <H> <R> <D> <SD> <TR>

<color> is the Ldraw color code
<pos> is the position in x, y and z coordinates
<rot> is the rotation matrix - <pos> and <rot> are equal to the standard part command of Ldraw
<H> is the total height of the spring
<R> the number of revolutions without the top and bottom revolutions
<D> is the diameter of the spring
<SD> is the thickness of the springs material
<TR> number of revolutions on top of the spring

 number of revolutions on bottom of the spring

following this command there is a SKIP section, containing standard Ldraw command for viewing in a none MLCad compatible viewer.

The algorithm for generating the spring is based on the spring tool from Marc Klein, who kindly provided the source for it.

6.11 Flexible Hose Pseudo Part

Minimum MLCad Version required: 3.00

Syntax: 0 MLCAD FLEXHOSE <c> <pos> <rot> <b> <bcp> <e> <ecp> <n> <fpn> <ipn> <dl>

<c> is the Ldraw color code
<pos> is the position in x, y and z coordinates
<rot> is the rotation matrix - <pos> and <rot> are equal to the standard part command of Ldraw

<b> is the begin point of the flexible hose in x, y and z coordinates
<bcp> is the begin control point in x, y and z coordinates
<e> is the end point of the flexible hose in x, y and z coordinates
<ecp> is the end control point in x, y and z coordinates

<n> is the number of intermediate parts to generate inside the hose
<fpn> is the name of the fist part
<ipn> the name of intermediate parts
<dl> a flag indicating if a line should be drawn following the flexible hose center

! File names are not enclosed in any quotation marks and must not contain any blank characters.

following this command there is a SKIP section, containing standard Ldraw command for viewing in a none MLCad compatible viewer.

The algorithm for generating the flexible hose is based on a algorythm from Chris Daelman, who kindly provided the source for it.

6.12 Rubber Belt Pseudo Part

Minimum MLCad Version required: 3.00

Syntax: 0 MLCAD RUBBER_BELT <c> <pos> <rot> <XD> <YD> <R1> <R2> <D> <P> <CY>

<c> is the Ldraw color code
<pos> is the position in x, y and z coordinates
<rot> is the rotation matrix - <pos> and <rot> are equal to the standard part command of Ldraw

<XD>, <YD> is the distance of the second point from the origin (pos).
<R1> is the radius of the virtual circle around the origin (pos)
<R2> is the radius of the virtual circle around the second point
<D> is the thickness of the rubber belt
<P> the precision
<CY> is a flag indicating weather to use lines and quads or cylinder parts. Cylinders are used if this flag is set to 1.

following this command there is a SKIP section, containing standard Ldraw command for viewing in a none MLCad compatible viewer.

The algorithm for generating the spring is based on the spring tool from Philo who ported it from Marc Klein's spring generator , who kindly provided the source for it.

6.13 Arrow Pseudo Part

Minimum MLCad Version required: 3.10

Syntax: 0 MLCAD ARROW <c> <pos> <rot> <w1> <w2> <L1> <L2> <D> <R> <T> <S> <CP> <CI>

<c> is the Ldraw color code
<pos> is the position in x, y and z coordinates
<rot> is the rotation matrix - <pos> and <rot> are equal to the standard part command of Ldraw

<w1> Pointer width
<w2> Indicator width
<L1> Total pointer length
<L2> Indicator length for streight pointers
<D> Distance back from the end of the pointer to the start of the indicator
<R> Radius for round arrows
<T> Type of arrow
1 ... Streight arrow, 2 ... round arrow
<S> Number of sections (1 – 3)
<CP> Color code of pointer
<CI> Color code of indicator

following this command there is a SKIP section, containing standard Ldraw command for viewing in a none MLCad compatible viewer.

6.14 Plate Pseudo Part

Minimum MLCad Version required: 3.10

Syntax: 0 MLCAD PLATE <c> <pos> <rot> <W> <L> <H>

<c> is the Ldraw color code
<pos> is the position in x, y and z coordinates
<rot> is the rotation matrix - <pos> and <rot> are equal to the standard part command of Ldraw

<W> Width of the plate or brick
<L> Length of the plate or brick
<H> Height of the plate or brick

following this command there is a SKIP section, containing standard Ldraw command for viewing in a none MLCad compatible viewer.

The algorithm was kindly provided by Tore Ericcson.

7. MLCAD INI-FILE

Minimum MLCad Version required: 3.00 – previous versions will not read the ini file

General rules:
File name: “MLCAD.INI”
Lines beginning with ; are comments ...
And empty lines are ignored

The ini file is read only which means MLCad does not modify it. It is used to be able to enhance some features of MLCad when new libraries of Ldraw are available without the need to modify MLCad itself. The latest version of this file can be downloaded from http://www.lm-software.com/mlcad/ always.

Each section begins with a section name in brackets: [SECTION-NAME]
There must be no space between the brackets and the section name itself.
A sections ends when another section begins or at the end of the file.

Feel free to modify/extend the contents of the file, but please send a copy of the file to michael.lachmann@lm-software.com if you would like to distribute it to the public. I will check the contents and provide it on my web server for distribution.
Please understand that I cannot except an ini-file containing references to unoffical or private parts!
Also please make sure that you are using the lastest part libraries available from www.ldraw.org

7.1 Minifig Generator Section

All Minifig Sections are containing lines of the following format:

; The following sections define the elements which are available in the
; minifigure generator. There is one section per minifig element
; The section names are mandatory - if they are missing or wrong MLCad uses
; a internal default instead, which is different from the content here!
; Each line has the following format:
"<Display name>" "<DAT/LDR file name>" <Flags> <Matrix> <Offset>

<Display name> is the name of the element as it is displayed in the element list
<DAT/LDR file name> The file name of the element or "" for a hidden element e.g. NONE
<Flags> always 0 reserved for future use
<Matrix> a rotation matrix a11 a12 a13 ... a33 for optimal appearance at 0 degree rotation angle
<Offset> The offset of the part to be in place

Example:
"Plain Leg" "971.DAT" 0 1 0 0 0 1 0 0 0 1 0 0 0

Following now a list of sections to be used for the minifig generator:
[HATS], [HEAD], [BODY], [BODY2], [NECK], [LARM], [RARM],
[LHAND], [RHAND], [LHANDA], [RHANDA], [LLEG], [RLEG],
[LLEGA], [RLEGA]

7.2 Path Scan Order Section

Minimum MLCad Version required: 3.10

The feature has been thought to include unofficial and custom parts into MLCad, without the need to mix them with the official Ldraw parts in the \PARTS and \P folders. DAT files found in the scan will be included in the parts library and - depending on the flag - show up in the parts tree library and parts preview window.
Take notice that this feature is NOT equal to the "File > Scan Parts" command and therefore will NOT rewrite the Parts.lst file. Other LDraw tools depending on the Parts.lst file might not recognise LDR or MPD files containing such "external" parts!
The start-up of MLCad will be slowed down depending on the number of folders and subfolders to be scanned.
The syntax supports extended ASCII-code.

NOTE: This is a temporary solution until LDraw.org has agreed on a overall solution, which will properly define these things in different ini file.

Starts with [SCAN_ORDER].
One line per directory to scan.
The format of a line is: n = <SHOW|HIDE> ["]Pathname["]

SHOW	Will display the parts found inside this directory in the part-library and/or part-preview window
HIDE	Will NOT show the parts found inside this directory in the part-library and/or part-preview window
Pathname	Subdirectory inside ldraw base path or absolute path including drive name \P, \PARTS and \MODELS are mandatory paths and are always present even if not specified. If not declared those will be scaned at last.

Example:

[SCAN_ORDER]
1 = HIDE P
2 = SHOW Unofficial
3 = HIDE Unofficial\s
4 = SHOW My_Parts
5 = SHOW MODELS\Test

7.3 Lsynth Command Section

Minimum MLCad Version required: 3.10

This section is used to define the path to Lsynth and the commands beeing displayed when adding new Lsynth commands to the project.

Regardless of the order of the statements in the .ini file the combo box will sort and list them in alphabetical order.

Starts with [LSYNTH].

%PATH = <path name>	Defines the absolute path to LSynth **NOTE!** (NOT the bin directory!)

The format of the selectable command lines is: <TITLE> = <COMMAND STRING>
TITLE is the name displayed in the combo box of the "Add LSynth Command" dialog.
COMMAND STRING is the command to be added to the project. Please refer to LSynths documentation for a deeper insight.

Example:
[LSYNTH]
%PATH = "C:\Programs\LSynth"
RIBBED_HOSE = SYNTH BEGIN RIBBED_HOSE 16
FLEX_SYSTEM_HOSE = SYNTH BEGIN FLEX_SYSTEM_HOSE 16
PNEUMATIC_HOSE = SYNTH BEGIN PNEUMATIC_HOSE 16

FLEXIBLE_AXLE = SYNTH BEGIN FLEXIBLE_AXLE 16

8. EXTERNAL INTERFACES

8.1 Language support

8.1.1. Language Dlls

To support a new language download the language template files from http://www.lm-software.com/mlcad and translate all english text into your desired language.
The template files represent a dll project which can be compiled using Microsoft VC++ 6.0.
Another approage might be to translate one of the existing language dlls directly using a certain tool which allows you to modify an existing dll, but this one has not been tested yet.
The final dll must follow a predifined naming convention:
MLCadLang<Language>.dll as an example MLCadLangGerman.dll
During startup of the program MLCad checks it's home directory for available dlls which follow this naming convention, performs some basic tests with it and finally presents it in the setup dialog for available languages using the name of the dll with the 'MLCadLang' prefix and without extension (in the example above this would be 'German'.
A dll failing the basic test will not be presented to the user.

The basic tests include:

- Tests if all resources are available
- Test if the language dll version is higher or equal the current MLCad language version. This MLCad language version is independed from the actual MLCad release, but will be increased every time new user interface items are added.

8.1.2. Help Files

As of MLCad version 3.00 I can no longer write the help files for MLCad myself, since I do not have the time for it, and since I would like to spend my spare time in developing MLCad rather than writing the documentation, which anyway is not of my best fields.

This section describes how to write/expand the help files of MLCad for a certain language.

Before you can write help files to be supported my MLCad there must already exist a language dll for that language.
All help files are located in a sub-directory called "HELP" which is in the same path as MLCad.exe itself. Below this sub-directory there is one directory for each language having the same name as the language itself (e.g. German, English, French …). So if you write help files for a new language you will have to create a new sub-directory. As base you can use the english help files for translation.

Since MLCad supports online help it will call the help files from within the program using a predefined url. The path will be the language help directory followed by a certain html file. Inside the help file there should be a ancore for direct access of the topic.
You can download a listing of actual help call mappings from the developer section on http://www.lm-software.com/mlcad