SIEMENS EDA

Calibre® MDPview™ User's and Reference Manual

Software Version 2021.2

SIEMENS

Support Center: support.sw.siemens.com
Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction to Calibre MDPview

The Calibre® MDPview™ data viewer is a full-featured application providing quick and easy review of fractured data and the layout data used to generate it.

Calibre MDPview Overview

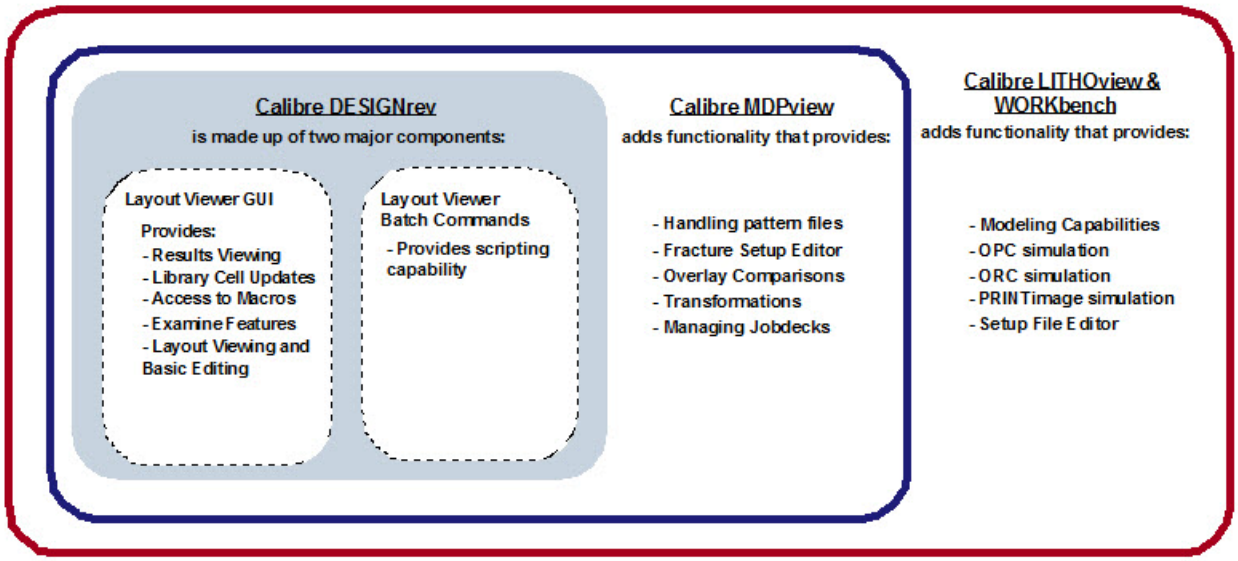
The Calibre MDPview layout viewer is part of a family of Calibre layout viewers that are used to view and manipulate layouts. With it, you can do the following:

- View Nuflare (VB11, VSB12, VSB12i, NUFLARE_MBF), MEBES, Hitachi, Micronic, OASIS®[1] (OASIS.MASK, OASIS.MBW, and OASIS.MAPPER) formats, and JEOL pattern files
- View Nuflare, MEBES, Hitachi, Micronic, JEOL, and MALY job decks
- Overlay layout database and pattern files
- View layout databases

As illustrated in Figure 1-1, Calibre MDPview shares core functionality with Calibre DESIGNrev, but has additional functionality targeted for mask data preparation.

1. OASIS® is a registered trademark of Thomas Grebinski and licensed for use to SEMI®, San Jose. SEMI® is a registered trademark of Semiconductor Equipment and Materials International.

Figure 1-1. Layout Viewer Family of Products



Tip

For More Information: This manual only covers the Calibre MDPview GUI components seen in Figure 1-1. Layout Viewer batch commands can be found in the Calibre DESIGNrev Reference Manual; the basic Calibre layout viewer operations are described in the *Calibre DESIGNrev Layout Viewer User's Manual*.

Table 1-1 provides a guide to locating information in the different layout viewer-related manuals.

Table 1-1. Where to Find Information

Functionality/Task	Where to Find It?
Invoking Calibre MDPview	"Calibre MDPview Modes of Operation" on page 19 *Calibre DESIGNrev Layout Viewer User's Manual*

Table 1-1. Where to Find Information (cont.)

Functionality/Task	Where to Find It?
Core Layout Viewer Tasks • Viewing layouts (view depth, zoom, and so on) • Configuring the layout viewer (rulers, grids, palettes, and so on) • Importing cells • Using Overlay/Merge to visually compare layouts • Sharing data • Flattening a hierarchy • Creating a non-design layer (marker layer) • Invoking RVE from the layout viewer • Making minor edits to the layout	*Calibre DESIGNrev Layout Viewer User's Manual*
Layout Viewer GUI Reference	
Palette Colors	
Concepts and Definitions	
Keyboard Shortcuts	
Configuration Files • keyprefs file • layermap file • layerprops file • preference file • readerprefs file • .signaext file	
Troubleshooting	
Batch Commands and Macros	*Calibre DESIGNrev Reference Manual*
Calibre MDPview-Specific Tasks • Loading pattern files • Viewing job decks • Importing cells	"View Data" on page 27 "Load Data" on page 43
Using the Fracture GUI For Fracture And MDPverify Operations	"Calibre FRACTURE Operations With Calibre MDPview" on page 61

Table 1-1. Where to Find Information (cont.)

Functionality/Task	Where to Find It?
Inspecting Pattern File Data • Pattern file statistics • Trapezoid statistics	"Data Operations" on page 83
Converting Data • Converting fractured data to OASIS format • Optimizing hierarchy of external tool-generated GDS or OASIS files. • Checking OASIS format conformance • Creating a single OASIS file from a job deck and its input pattern files (job deck smashing) • Converting job decks to OASIS files • Converting a MEBES job deck to VSB format • Converting a MEBES job deck to Micronic format • Convert a 2D barcode to OASIS data	
Indexing Files to Speed Loading and Viewing	
Assembling Multiple Layout Databases into a Single Database	"Assemble Multiple Layout Databases into a Single Database" on page 81
Error and Warning Messages	"MEBES Job Deck Warning Messages" on page 141

Calibre MDPview provides the capability for a quick review of the design data in GDSII and OASIS format as well as in all supported mask writer pattern formats and data assemblies through the respective job decks.

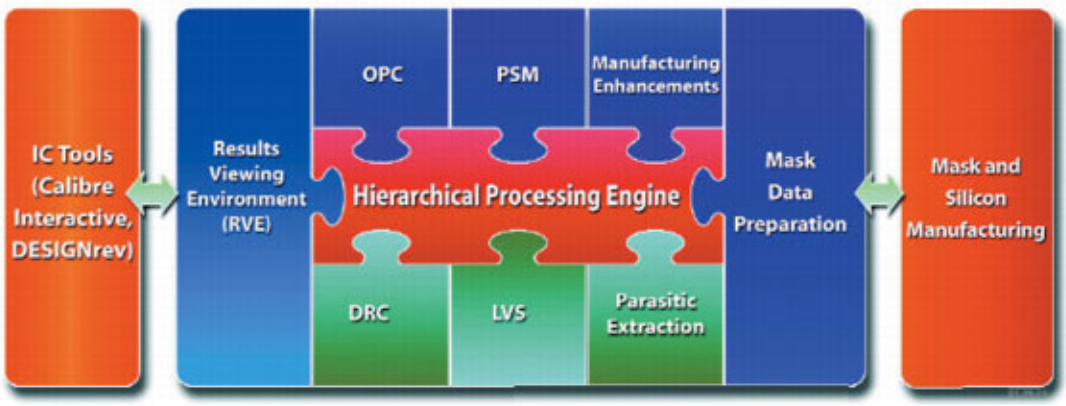
Calibre MDPview can be linked with Calibre RVE to guide you through areas previously identified as problems during the mask rule check executed with Calibre® MDPverify™®. The tool matches the location with the displayed layout, zooms-in, then highlights the affected pattern.

Further analysis is enabled by overlay capabilities that allow you to display and align various files on top of each other. Statistical utilities are available to characterize pattern files. Macros and scripts can be created and executed in batch mode to customize repeated functions such as the assembly of a number of GDSII input files into a single database.

Calibre MDPview Workflow

Calibre MDPview is part of a suite of tools for the Calibre Mask Data Preparation (MDP) flow. Calibre MDP allows for seamless continuation of the data manipulations required for Resolution Enhancement Technology (RET) techniques to the mask data format conversion in one batch run, keeping data hierarchically represented as long as possible.

Figure 1-2. Calibre Design-To-Silicon Flow



The Calibre MDP suite of tools include:

- Calibre® MDP Embedded SVRF™ — Allows you to insert a block of Standard Verification Rule Format (SVRF) commands in a FRACTURE, Calibre MDPverify, MDP EMBED, or DENSITY_CONVOLVE invocation. SVRF commands are documented in the *Standard Verification Rule Format (SVRF) Manual*.
- Calibre® MDP EMBED™— Provides section-based processing for some SVRF commands, essentially providing section-based DRC capability. Section-based processing is an alternate method of layout processing that may improve scalability and turnaround time for some limited cases such as mask rule checks (which are DRC-like checks on mask data).
- Calibre® MDPverify™ — Allows you to evaluate fractured MDP data by performing geometrical comparisons of fractured data output to the original layout data, database layer, or fractured data in a different pattern file.
- Calibre® MDPmerge™ — For VSB11 only, reads a VSB11 job deck and merges individual chips and chip placements into a single chip and placement, respectively, based on certain criteria. The output is a new job deck consisting of the merged chips and copies of chips that did not meet the merge criteria.
- Calibre® MDPstat™ — Analyzes vector beam data such as Nuflare, JEOL, Hitachi, and OASIS formatted data allowing you to assess the quality of results of the fracture.

- Calibre® MPCpro™ — A rule-based MPC application that applies pattern-based mask process corrections specified by SVRF commands to your design data prior to fracturing. This includes correcting linearity and proximity effects as well as longer-range processing effects such as fogging and develop- and etch-loading.
- Calibre® MPCverify — A grid-based mask process simulator and MPC results verification tool designed to predict mask manufacturing processes. It supports the industry-standard MPC usage models and seamlessly integrates with other Calibre tools. Refer to the *Calibre MPCverify User's and Reference Manual* for further information.
- Calibre® MASKOPT™ — Modifies the input geometry to reduce shot count and improve mask write time through rule-based vertex alignment.
- Calibre® nmMPC™ and Calibre® nmCLMPC — Calibre nmMPC is a suite of functions used for modelling, simulating, and correcting distortions of the mask manufacturing process. Calibre nmCLMPC performs the same functions, but is specifically targeted at curvilinear layouts containing skew edge polygons. Models generated by this process are utilized by Calibre nmMPC, Calibre nmCLMPC, and FRACTURE tools to accurately reflect the final output of the mask process. Refer to the *Calibre nmMPC and Calibre nmCLMPC User's and Reference Manual* for further information.
- Calibre® MDPview™ — A Graphical User Interface (GUI) viewer that enables you to visually and interactively inspect the results of fracture and verify operations (pattern files, job decks), as well as large post-tape-out OASIS files. Calibre MDPview utilities are TCL-based commands that allow script-based manipulation of fractured data and job decks, and include conversion of pattern files or job decks to OASIS (job smashing), conversion of MEBES job decks from MEBES to other formats, and pattern file quality, optimization and informational utilities. This tool is documented in the *Calibre MDPview User's and Reference Manual*.
- Calibre® Job Deck Editor™ — A tool invoked from Calibre MDPview that is used to generate an optimized chip placement on the wafer, with the ability to manually edit chips in the job deck. This tool is documented in the *Calibre Job Deck Editor User's Manual*.
- Calibre® MDPDefectAvoidance™ — A tool invoked from Calibre MDPview that is used to find shifts in a layout to prevent extreme ultraviolet lithography (EUVL) blank mask defects from appearing on patterns. This tool is documented in the *Calibre MDPDefectAvoidance User's Manual*.
- Calibre® DefectReview™ — A tool that provides efficient classification, and trend analysis of defects identified by wafer and mask inspection systems. The software includes features for defect navigation, selection, filtering, classification, and clustering. Analysis tools include visual display capability, CD analysis, analysis over multiple inspections, and repeatability and trend analysis. This tool is documented in the *Calibre DefectReview User's Manual*.

- Calibre® MDPAutoClassify™ — A tool that is used for automatic classification of defects observed on a blank substrate before any patterning is performed on the substrate. This tool is documented in the *Calibre MDPAutoClassify User's Manual*.
- Calibre® DefectClassify™— A tool that en ables automatic classification of defects observed on a patterned mask. This tool is documented in the *Calibre DefectClassify User's Manual*.

The following table lists the documents associated with the related Calibre tools.

Table 1-2. Related Products and Their Manuals

Related Products	Documentation
Calibre® FRACTUREc™ Calibre® FRACTUREh™ Calibre® FRACTUREi™ Calibre® FRACTUREj™ Calibre® FRACTUREm™ Calibre® FRACTUREn™ Calibre® FRACTUREp™ Calibre® FRACTUREt™ Calibre® FRACTUREv™ Calibre® MDPmerge™ Calibre® MDPstat™ Calibre® MDPverify™ Calibre® MPCpro™ Calibre® MASKOPT™ Calibre® MDP Embedded SVRF	*Calibre Mask Data Preparation User's and Reference Manual* *Calibre Release Notes*
Calibre® MDPview™	*Calibre MDPview User's and Reference Manual* *Calibre Release Notes*
Calibre® Interactive™ Calibre® RVE™	*Calibre Interactive User's Manual* *Calibre RVE User's Manual*
Calibre® nmDRC™ Calibre® nmDRC-H™	*Calibre Release Notes* *Calibre Verification User's Manual* *Standard Verification Rule Format (SVRF) Manual*

Table 1-2. Related Products and Their Manuals (cont.)

Related Products	Documentation
Calibre® WORKbench™	*Calibre WORKbench User's and Reference Manual*
Tcl/Tk Batch Commands	*Calibre DESIGNrev Reference Manual*
Calibre® Metrology API (MAPI)	*Calibre Metrology API (MAPI) User's and Reference Manual*
Calibre® Job Deck Editor	*Calibre Job Deck Editor User's Manual*
Calibre® MDPDefectAvoidance™	*Calibre MDPDefectAvoidance User's Manual*
Calibre® nmMPC™ Calibre® nmCLMPC	*Calibre nmMPC and Calibre nmCLMPC User's and Reference Manual*
Calibre® MPCverify	*Calibre MPCverify User's and Reference Manual*
Calibre® DefectReview™	*Calibre DefectReview User's Manual*
Calibre® MDPAutoClassify™	*Calibre MDPAutoClassify User's Manual*
Calibre®DefectClassify™	*Calibre DefectClassify User's Manual*

Calibre MDPview Prerequisites

There are a number of prerequisites before using Calibre MDPview.

- **Platform support** — Calibre MDPview is available on all supported platforms found in the *Calibre Administrator's Guide*. Refer to that document for instructions on how to install Calibre software.
- **Licensing** — To run any of the Calibre layout viewers, you must have the license file required by the tool. Calibre DESIGNrev, Calibre WORKbench, Calibre LITHOview, and Calibre MDPview each require a different license. For more information on licensing, refer to the *Calibre Administrator's Guide*.
- **Required files** — The following files are required to perform MDP operations:
  - A GDS or OASIS layout ready for mask data preparation
  - An SVRF file

Calibre MDPview Modes of Operation

You run the Calibre MDPview data viewer from a command line shell using command line invocations.

Interactive GUI Mode

You invoke Calibre MDPview from a command console. The actual command you use controls which of the tools you invoke. The command arguments control the mode of operation. Use the following steps to invoke Calibre MDPview.

Prerequisites

- Before invoking the application, you must set the environment variable CALIBRE_HOME to the location of your Calibre Software tree. Calibre tools require that the CALIBRE_HOME environment variable be set. See "*Setting the CALIBRE_HOME Environment Variable*" in the Calibre Administrator's Guide for details.

Procedure

1. Invoke a command line shell.
2. In the shell, invoke the application by typing the appropriate command:

   ```
   calibremdpv
   ```

   Alternatively:

   ```
   calibrewb -mdp
   ```

   The prompt in the shell changes to "%" and the Calibre MDPview graphical user interface (GUI) appears.
3. You can also specify additional command line arguments, as described in "Batch Mode" on page 20.

Batch Mode

Used alone, the invocation command **calibremdpv** invokes the Calibre MDPview layout viewer. Optional command switches let you open the application in different modes, evaluate scripts, or load data upon invocation.

Tip

Specifying the -h option shows a brief help list of the available command-line invocation options. The layout viewer attempts to run that file as a Calibre layout viewer Tcl script. Use this option when there is no XWindows display connection.

Usage

calibremdpv [-a *command*] | [*script_name*] | [-shell] [-s *script* | -startupscript *script*] [-m *layoutfile* [-levels *level1 ... leveln*]] [-v *drlayoutfile*] [-l *layerpropfile*][-g *cellname*] [-o *overlayfile*] [-colorByDefinition | -colorByPlacement] [-rve [*dbType*] [*rveFile* [*topCellName*]]] [-b *bookmarkname*] [-hidelayers] [-endDepth *depth*] [-dl *default_file*] [-tb classic | origin | ignore | extent] [-wait *time* | -nowait] [-noedit] [-threads *numthreads*] [-index_mask_files {ALL | NONE | *size_in_MBs*}] [-h] [-version] [--]

Arguments

- -a *command*

  An optional switch that evaluates a single batch command and exits.

- *script_name*

  An optional switch that causes the layout viewer to run the script_name file when *script_name* is the first argument. Any text that appears after the “--” (double dash) argument is sent to the script. The layout viewer exits after the script completes.

- -shell

  An optional switch that opens just the terminal shell window (no GUI). To interact with the program you must type commands into the shell. Use this option when there is no XWindows connection.

- -s *script* | -startscript *script*

  An optional switch that evaluates the specified Tcl script on invocation. Can be used with GUI mode or -shell. Use this option when you want to recreate a situation from a previous run.

- -m *layoutfile*

  An optional switch that loads flat data into the viewer using a specified layout (such as GDS and OASIS), fracture, or pattern file(s) (for example, JEOL, VSB, MICRONIC, HITACHI, MEBES) on invocation. If you specify a layout file without the -m option, the layout viewer automatically attempts to determine if it is a layout file. Otherwise, the tool assumes the file is a Tcl macro script and passes it to the Tcl interpreter.

If you are loading multiple files, each file name must be specified with its own -m option. The -m option cannot be used with the -v option.

Caution

The Calibre layout viewers do not support spaces in path names. The command line syntax does not allow spaces.

- -levels *level1... leveln*

  Specifies a list of levels used for normal, extended, and flex MEBES job decks. This is used in conjunction with -m option. Using a level list reduces load times and memory usage for job decks that have assigned unique files to each level, since only those pattern or index files that match a level in the list are loaded.

  Using a level list on the other job deck types (Hitachi, JEOL, VSB, Micronic) has no effect.

- -v [*drlayoutfile*]

  Specifies an optional OASIS layout file to be loaded as a view-only, disk-resident, indexed file. If a disk-resident index file is not present, one is created. You can also specify -v without an OASIS file to load all OASIS files as view-only disk-resident files within the Calibre MDPview session. This puts Calibre MDPview into the disk-resident OASIS mode for any other OASIS file reads.

  If you want to place Calibre MDPview into the disk-resident OASIS mode and still load an OASIS file at startup, specify "calibrempdv -v -m <*layout*>.*oas*". The mode is set before reading the -m specified file.

  For overlays created with disk-resident OASIS files, simply add -v to the startup switches and load the overlay with the -o option or through the GUI.

  Note

  Disk-resident OASIS files have different top cell names from normal "memory-resident" OASIS files. These cell names are saved in the overlay file, and the overlay files are not interchangeable between file read mechanisms. For example, overlays created from "memory-resident" OASIS files can only be loaded in the normal OASIS read mode, and overlays created from "disk-resident" OASIS files can only be loaded in the disk-resident OASIS read mode.

- -l *layerpropfile*

  Can only be used with -m or -v. Loads a single layer properties file. If you load multiple files, this layer properties file is used for all layouts. This option causes the -dl option to be ignored.

- -g *cellname*

  Opens the design file to the specified cell. This option cannot be used with the -v option.

- -o *overlayfile*

  Loads an overlay file (.ovly). This file must have been previously created using the Overlay Setup dialog (described in the *Calibre DESIGNrev Layout Viewer User's Manual*).

- -colorByDefinition

  Causes all layers within each job deck chip definition to be colored the same. The same color is assigned to both the level layers and their contained chip layers in the level and chip views. A single MDP job deck can be loaded with the -colorByDefinition startup switch. View modes can only be changed by closing the layout and reloading with a different mode setting. This option is further described in the section "Display Chip File Names in the Layer Palette" on page 38.

- -colorByPlacement

  Enables the MDP Chip View option to enable display of chip File Names in the layer palette (as described in the section "Display Chip File Names in the Layer Palette" on page 38) for the job deck loaded with the -m option. This affects the underlying data in the database and changing the preference only takes effect when loading the job deck (reloading does not change it). The options are to change the option, close the layout, open the job deck, or exit and restart the tool.

- -rve [*dbType*] [*rveFile* [*topCellName*]]

  Runs Calibre RVE from the command line (this must be run in GUI mode). The syntax is as follows:

  dbType— Specifies one of the following database types: -drc, -lvs, -dfm.

  rveFile — The layout viewer invokes RVE and loads an optional RVE file specified by *rveFile*.

  topCellName— Specifies the top cell name. This cannot be specified unless *rveFile* is also specified.

- -b *bookmarkname*

  Loads a previously-created bookmark.

- -hidelayers

  Causes all the layers to start as set to hidden in order to speed up initial load time when specified with the -m option, Overrides any other related preference settings for this session only.

- -endDepth *depth*

  Starts with the end depth set to the specified value. Overrides other related preference settings for this session only.

- -dl *default_properties_file*

  Specifies a default layer properties file. This option is overridden by the -l option.

- -wait *time*| -nowait

  Specifies how Calibre DESIGNrev, Calibre MDPview, Calibre WORKbench, and Calibre LITHOview handle waiting for licenses.

  By default, a Calibre tool waits indefinitely for a license to become available. With the -wait option, a tool waits for any appropriate license for the number of minutes specified by the *time* argument. The tool then exits if no licenses become available before*time* expires.

  Licenses can be acquired from this product or from any of the tool's parent products. The following is the license search order:

  a. Calibre DESIGNrev

  b. Calibre MDPview

  c. Calibre LITHOview

  d. Calibre WORKbench

  License checking occurs in two stages. The first pass checks for a qualified license without a wait. If no license is acquired, a second pass is performed and a wait is done on the first existing qualified license.

  For example, Calibre MDPview can use a Calibre MDPview, Calibre LITHOview, or Calibre WORKbench license, if one is available, but it cannot use a Calibre DESIGNrev license.

- -tb classic | origin | ignore | extent

  Specifies how the extent of a text affects the calculation of a cell bounding box:

  classic — Text origin is used as a lower-left coordinate of the text extent. The height is set to 0.25 microns independent of the actual text attributes or user units of the design. The contribution to the bounding box extent of the width is zero regardless of number of characters in text. The text extent is added to the cell extent (but not the text width).

  origin — The text origin is used as the lower-left and upper-right coordinate of the extent. The text has no extent and only its origin impacts the bounding box extent. The origin of the text is added to the cell extent.

  ignore — The text does not contribute to the cell extent.

  extent — The text extent in the database is determined mainly by the height of text. The text width is calculated by using the character length multiplied by a fixed ratio of the height. The ratio is given by the font geometry of the application. Because of the under stroke of certain characters like, "g" the lower-left coordinate can be slightly lower than the origin of the text. If the text is rotated or aligned, the extent is calculated taking these attributes into account. The text extent is added to the cell extent.

- -noedit

  Prohibits a user from editing the layout. It can be used in GUI mode or with -shell. The -noedit option is commonly used with the option '-m' to view the changes made to a layout following a batch mode run.

- -threads *numthreads*

  Specifies the number of threads to be used for multithreading. The default is to use all available CPUs, one thread each. However, multithreading is always on, even if only one CPU is available. The startup switch is only useful to limit the number of threads.

- -index_mask_files {ALL| NONE | *size_in_MBs*}

  Generates an index file. For several formats, a persistent additional index file associated with the input layout is generated. In general, the index file is helpful in maintaining good performance, but you can control whether or not to generate a file on disk using index_mask_files. This option is supported for MEBES and OASIS files.

  ALL — Generates index files for any pattern file.

  NONE — Prevents the creation of an index file on disk (an "index" module is still created in the program without performance degradation).

  size_in_MBs — Represents the threshold size in MegaBytes (MBs) for the pattern file may also be specified. For any pattern file that is greater in size than *size_in_MBs*, an index file is generated.

  The default setting is ALL.

- -h (or -help)

  Prints the invocation command line help usage.

- -version

  Prints the version number of the application.

- --

  Designates that all remaining arguments are passed directly to a script's argv variable without interpreting them. This option does not work with the -s option. Use this option when a Tcl script requires arguments or when specifying arguments that have hyphens embedded within them

Tip

Layout Viewer displays status, errors, and warning messages in the shell window. The Linux tee command can be used to place log information into a file, which is useful for debugging and diagnosing problems. For example, for Calibre WORKbench, the command would be the following:

% calibrewb | tee <*filename*>

Description

Layer Properties File Processing Order

If you specify the -l, -dl, and -m options in the same invocation, each option specified successively overrides like values of the previous option. The order of processing is as follows:

1. The default file (the standard *.calibrewb_workspace/layerprops* as specified by -dl).

2. Calibre detects the presence of a <*layoutfile*>.*layerprops*file.
3. A file specified by -l.

This order applies without any of those options declared. The default and *layoutfile*.layerprops file are always searched for and used.

Run RVE From the Command Line

You can run Calibre RVE from the command line with the -rve option. The layout viewer also invokes Calibre RVE when it first starts up, if -rve is specified. If a valid Calibre RVE filename and argument list are specified, the application loads that file using the specified arguments. Note that the file and any arguments need to be interpreted as a single character string by Tcl, which requires you to enclose the string in quotes (“”) or braces ([]). If you are supplying a filename with no arguments, the quote marks are optional.

Examples

The following example evaluates the Tcl code in report.tcl, then exits.

```
calibremdpv -startscipt report.tcl
```

The following example starts an interactive Calibre MDPview data viewer session and opens the GDS file called topcell, but does not permit editing it.

```
calibremdpv -m topcell.gds -noedit
```

Caution

Do not start the application as a background process as it is not supported.

Syntax Conventions

The command descriptions use font properties and several metacharacters to document the command syntax.

Table 1-3. Syntax Conventions

Convention	Description
Bold	Bold fonts indicate a required item.
Italic	Italic fonts indicate a user-supplied argument.
Monospace	Monospace fonts indicate a shell command, line of code, or URL. A bold monospace font identifies text you enter.
Underline	Underlining indicates either the default argument or the default value of an argument.

Table 1-3. Syntax Conventions (cont.)

Convention	Description
UPPercase	For certain case-insensitive commands, uppercase indicates the minimum keyword characters. In most cases, you may omit the lowercase letters and abbreviate the keyword.
[ ]	Brackets enclose optional arguments. Do not include the brackets when entering the command unless they are quoted.
{ }	Braces enclose arguments to show grouping. Do not include the braces when entering the command unless they are quoted.
‘ ’	Quotes enclose metacharacters that are to be entered literally. Do not include single quotes when entering braces or brackets in a command.
| or ||	Vertical bars indicate a choice between items. Do not include the bars when entering the command.
…	Three dots (an ellipsis) follows an argument or group of arguments that may appear more than once. Do not include the ellipsis when entering the command.
Example: **DEVice** {*element_name* [‘(’*model_name*‘)’]} ***device_layer*** {***pin_layer*** [‘(’*pin_name*‘)’] …} [‘<’*auxiliary_layer*‘>’ …] [‘(’*swap_list*‘)’ …] [BY NET| BY SHAPE]	

Chapter 2
View Data

There are a number Calibre MDPview-specific operations related to viewing mask data in either database format (typically GDS or OASIS) or fractured format. Once displayed, you can analyze the data.

Note

More general layout viewing operations are covered in the *Calibre DESIGNrev Layout Viewer User's Manual.*

Calibre MDPview Display Overview

Calibre MDPview supports the following fracture pattern files shown in the following table.

Table 2-1. Supported FRACTURE Formats

HITACHI Formats	
HL-800	HL-950M
HL-900D	HL 7000M
JEOL Formats	
JBX-9000MV (v3.0)	JBX-9300FS-50kV (v3.0)
JBX-3030MV (v3.0 and v3.1)	JBX-9300FS-100kV (v3.0)

Table 2-1. Supported FRACTURE Formats (cont.)

JBX-3040MV(v3.0 and v3.1)	JBX-3050MV(v3.0 and v3.1)
JBX-3200MV	
MEBES Formats	
Mode 3[1]	Mode 5
Mode 4	
MICRONIC Formats (MIC)	
v1.6	v1.8
Toshiba/NUFLARE Formats[2]	
VSB11	VSB12 VSB12i (VSB12iEL) NUFLARE_MBF
OASIS Formats	
OASIS.MASK OASIS.MAPPER	OASIS.MBW

1. For MEBES Mode 3, writing and indexing are not supported. Reading is supported for the following products and utilities: Calibre MDPview (including the utilities mebes2oasis, jobToOasis, and mebesinfo) and Calibre MDPverify.
2. This includes support for EBM 3000/3500/4000/6000/7000/8000/900.

VSB12i, VSB12iEL, and NUFLARE_MBF Format and Attribute Information Support

The VSB12i, VSB12iEL, and NUFLARE_MBF formats support attribute information (AI) values for the geometries. Calibre MDPview displays the geometries with different AI values on different datatypes such that the datatype numbers match the AI values. The geometries with no AI values are displayed on datatype 0. VSB12i supports a maximum of 8 different AI values, while VSB12iEL supports a maximum of 1024 different AI values. By default, the viewer displays the complete list of datatypes, whether any geometries contain the corresponding AI value or not. In the following two scenarios, the viewer trims the list of datatypes to display only the datatypes containing geometries:

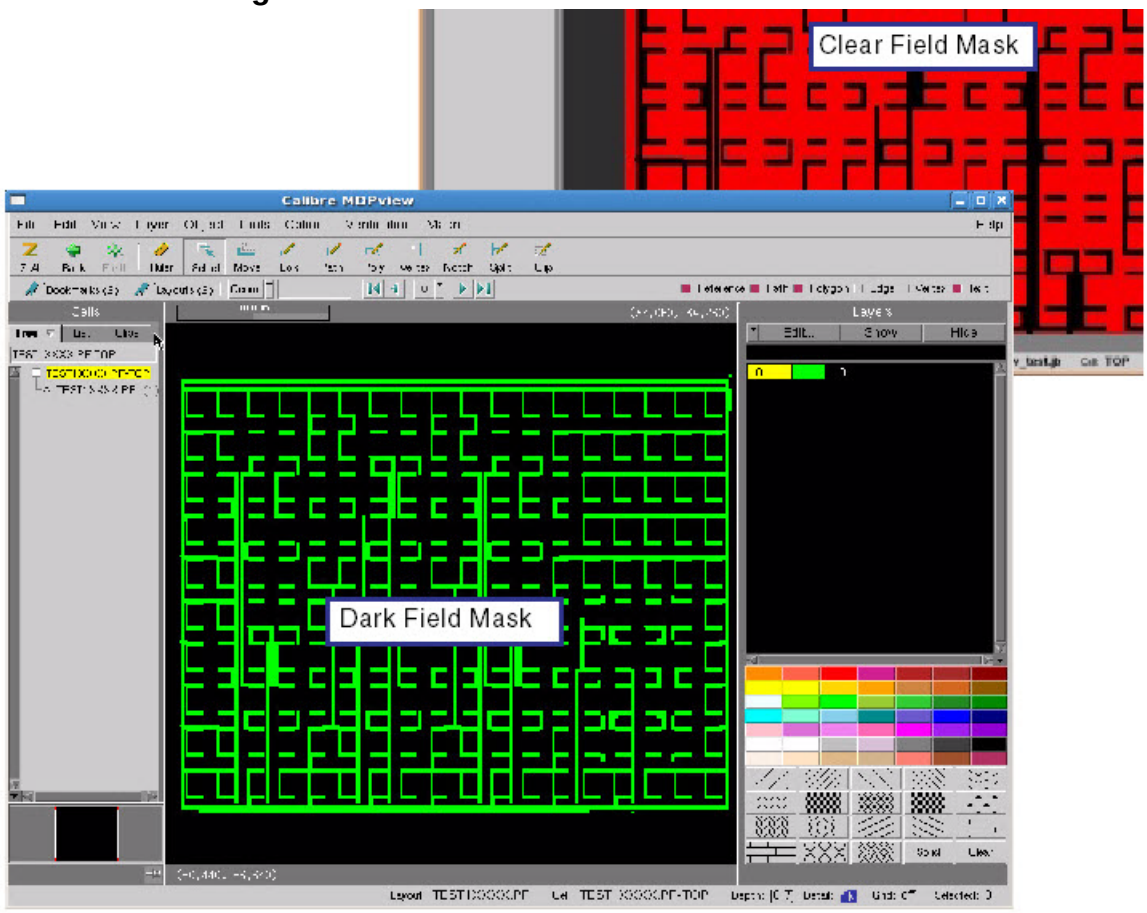
- The index file corresponding to the VSB12i, VSB12iEL, or NUFLARE_MBF pattern file is present. The datatype filtering is faster in this case since the required information is readily available in the index file.
- The following environment variable:

```
CALIBREWB_VSB12I_LAYER_FILTER {true | false}
```

is set to true. This environment variable instructs Calibre MDPview to parse the entire pattern file to obtain the required information, thus slowing the filtering process.

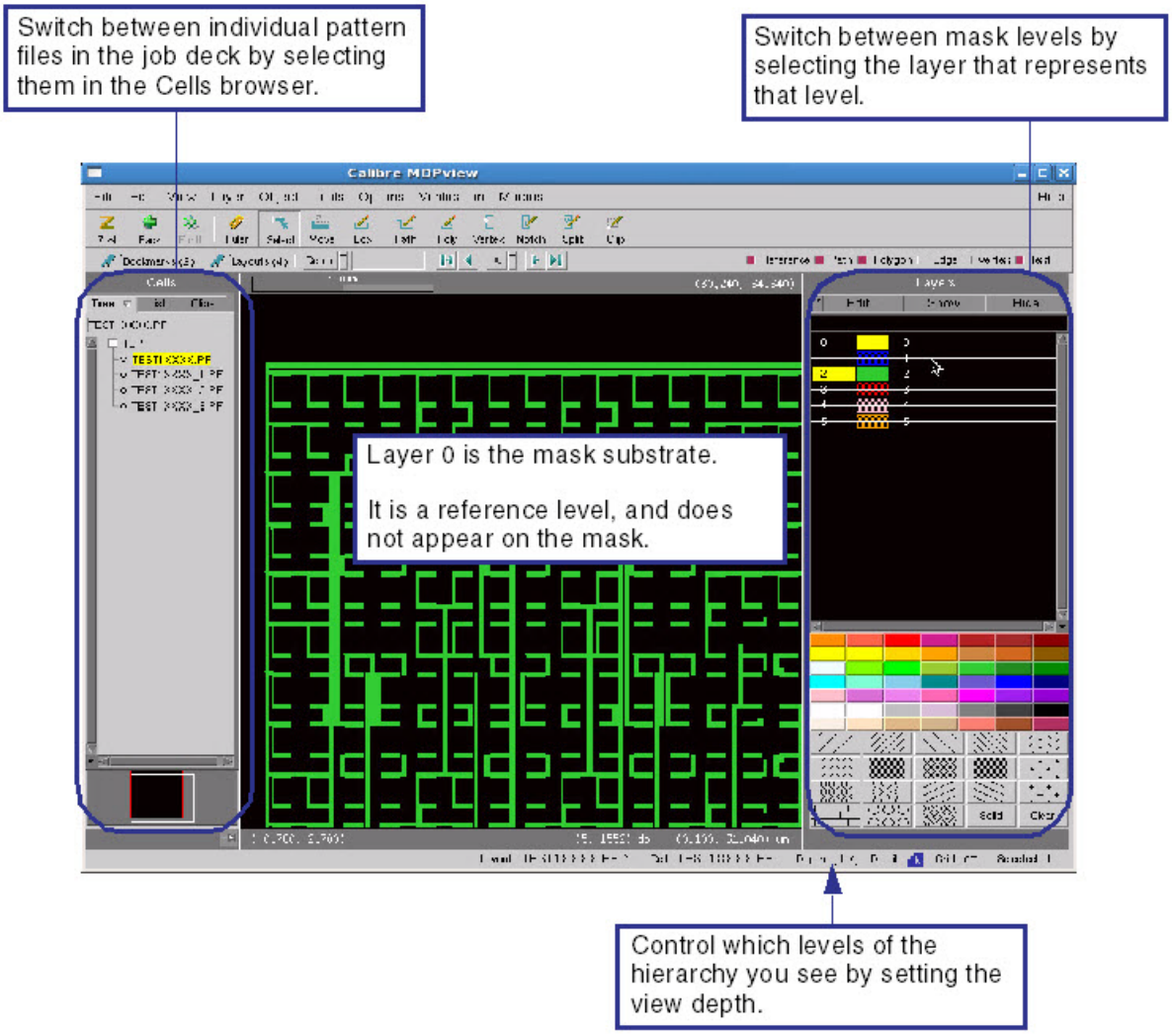
In the Calibre MDPview data viewer, the foreground (colored data) represents the areas the mask writer exposes for positive photoresist. By default, the viewer displays the complete list of 1024 data types (for both VSB12i, VSB12iEL, and NUFLARE_MBF).

Figure 2-1. Mask Data in Calibre MDPview



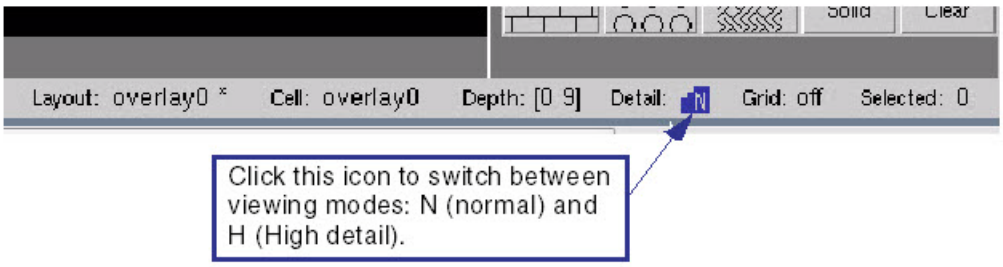
The following figure shows some of the ways you can control which mask data is visible at any time. For more detailed ideas, refer to the sections that follow.

Figure 2-2. Controlling the Data You See



In addition, you can control the level of viewing detail in your display, choosing between two modes: N for normal display, and H for a higher level of detail (see the following figure).

Figure 2-3. Display Detail Button



Control Which Mask Levels You See

Data displayed in Calibre MDPview is comprised of layers. For fractured data, each layer represents a unique mask level. For database data, it may take several layers to make up a mask level.

The Layer pane, normally shown in the right side of the Display area (you can resize the pane to hide it using the grey scroll box at the bottom left corner of the pane), lets you control the properties of all the layers in the current layout.

By default, layers are sorted by number and displayed in ascending numerical order. You can change the order in which layers are displayed using the Layer Sort options, which you access by choosing **Layer > Sort by** from the menu (or the Sort drop down in the Layer pane).

The options allow you to sort layers by number, name, select status, or visibility.

A bright white layer number indicates that the layer has polygons visible in the current view. If the layer number is a dim grey, no objects of that polygon are currently viewable.

When viewing a single pattern file, there is a one-to-one mapping between a mask level and a viewer layer.

When viewing a job deck, you can choose between two modes:

- **Default mode** — Each mask level in the job deck is displayed as a single color. In this mode the entire mask level appears as a single layer in the viewer.
- **Chip mode** — The levels in the job deck are expanded into a set of layers, one for each chip. In this mode, mask level 2 in chip 1 is displayed as a different viewer layer than mask level 2 in chip 2.

You can switch between modes using Ctrl key + comma (,).

Control Which Pattern File or Cell You See

The layout hierarchy defines the structure of the layout data: the building blocks and instances of these building blocks.

All job decks and most design databases are hierarchical objects, though the hierarchy represents different kinds of data, depending on the type of data you are viewing:

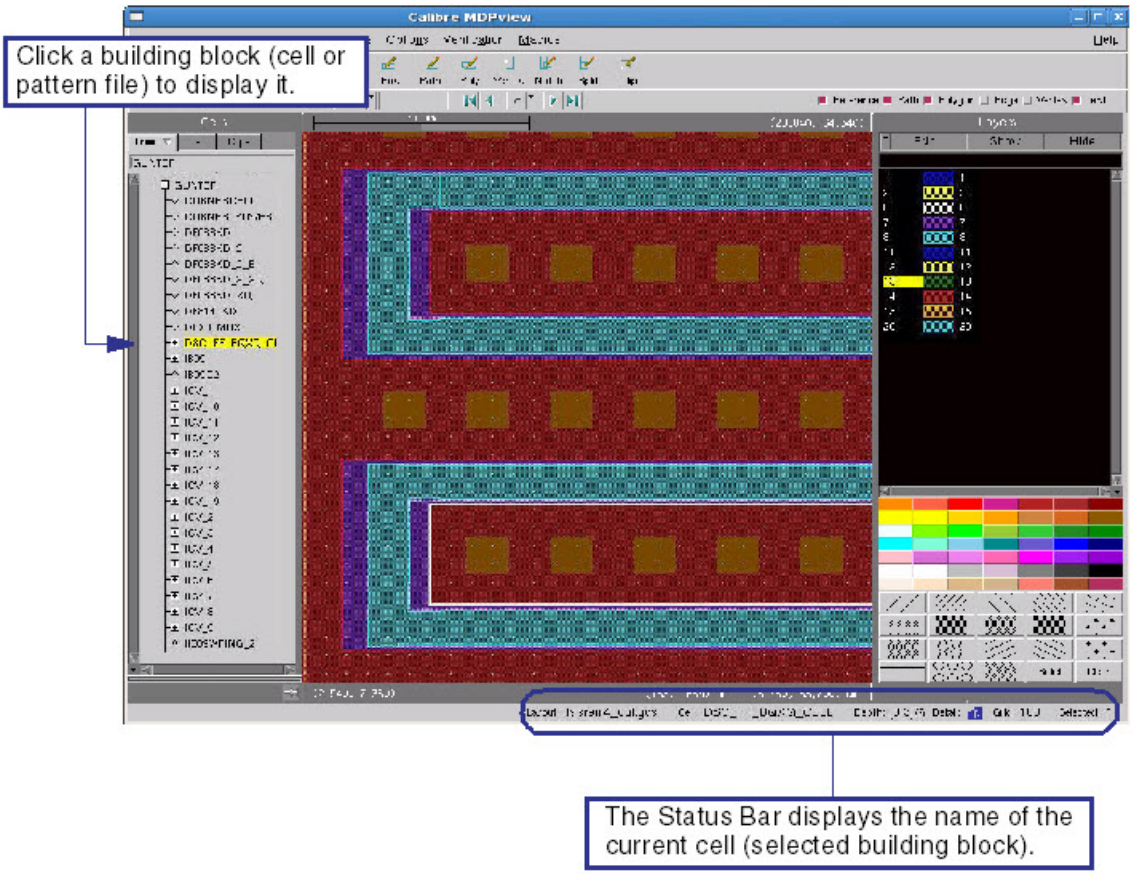
- The building blocks for job decks are the individual pattern files that are printed one or more times on a mask.
- The building blocks for design databases are cells that are instantiated one or more times in a chip.

Table 2-2. How Hierarchy Relates to Different Types of Files

Type of File	# of Levels	Levels
Pattern File	2	**Top Level** — Data extent **Lower Level** — Fractured Data
Job Deck	2	**Top Level** — Whole mask outline **Lower Level** — One for each pattern file
Database (GDS or OASIS)	Unlimited	**Top Level** — Whole chip view **Lower Levels** — Cells that are building blocks for the chip.
Merged Layout	Unlimited	**Top Level** — Top level Next Level —One for each layout that was merged (top level before merging) **All Others** — Data from levels > 0 before merging.

In the Calibre MDPview data viewer, the building blocks of the hierarchy, whether they are pattern files or design cells, are referred to as *cells*. The building block that is visible is called the *current cell*.

Figure 2-4. Hierarchy and What You See



Controlling What Level of Detail You See

In some cases the window size needed to view all of your data can be very large in relation to the size of the geometries. When this occurs, your geometries may not display properly. To ensure that you see the geometries correctly you can change the size of the smallest object that is displayed.

Procedure

1. The first method to adjust the level of details displayed is as follows:
   a. Choose **Edit > Preferences**.
   b. Select **View** tab.
   c. Change **Minimum Object size to draw** text entry box to **0**.
   d. Apply changes.

      To improve drawing performance, Calibre MDPview applies a resolution process known as “culling.” With this process, if something in the view is smaller than **x**

(specified by **Minimum Object size to draw**), then a box of fixed size **x** by **x** is drawn in its place. For example, if you specify that 10 pixels is the smallest size limit, anything with a side less than 10 pixels is drawn as a 10x10 box. This can lead to some drawings becoming coarser, but culling greatly improves performance.

2. Another method that prevents the dropping of shapes (at a cost of increased draw time) is as follows:

   a. Select the **View** menu. The Choose Viewing Detail dialog appears.

   b. In the Chose Viewing Detail dialog box, select **Custom**, then set the minimum object pixel size and detailed reference pixel size to 1.

   a. Click **Apply**.

Results

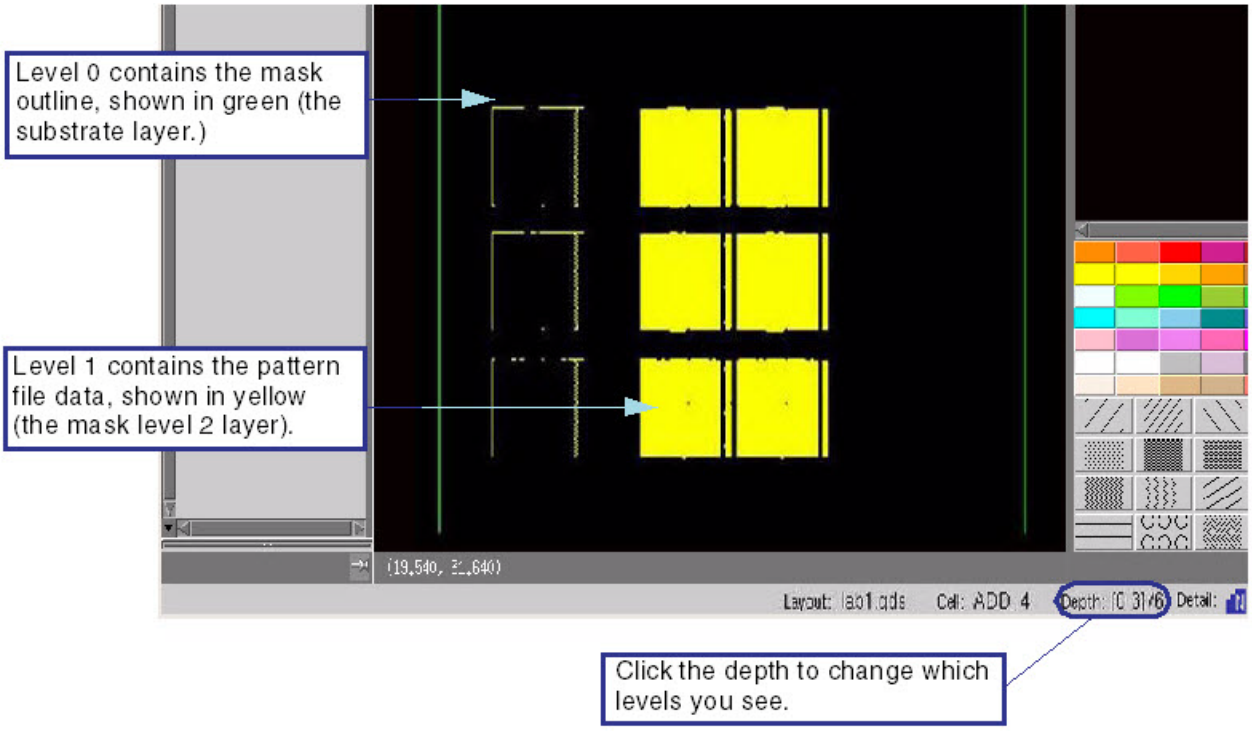
Calibre MDPview updates its display to the detail level set.

Control Which Levels of the Hierarchy You See

When the current cell (the data that you can see in the Calibre MDPview data viewer) is hierarchical, you can view more than one hierarchical level at a time.

This is illustrated in Figure , which displays the job deck’s mask outline plus pattern files.

Figure 2-5. Viewing Two Levels of Hierarchy at the Same Time



The Calibre MDPview data viewer displays data on all levels between From depth and To depth, inclusive. You set the From and To depths through the View Depth dialog box (click depth values in the lower right corner of the window) or with keyboard shortcuts.

Table 2-3. View Depth Keyboard Shortcuts

	From Depth	To Depth
set to value	ctrl-<number>	<number>
increment	ctrl-shift-'>'	>
decrement	ctrl-shift-'<'	<
maximum	—	double-9

Viewing Cell Orientation

At the top level of the job deck view depth, you can show the relative orientation of all cells to the viewer window (indicated by an triangle with an "F"). This allows you to see how the top cell is oriented after transformation (such as with a rotation or mirror) when interactive commands are given, without having to re-render the entire cell.

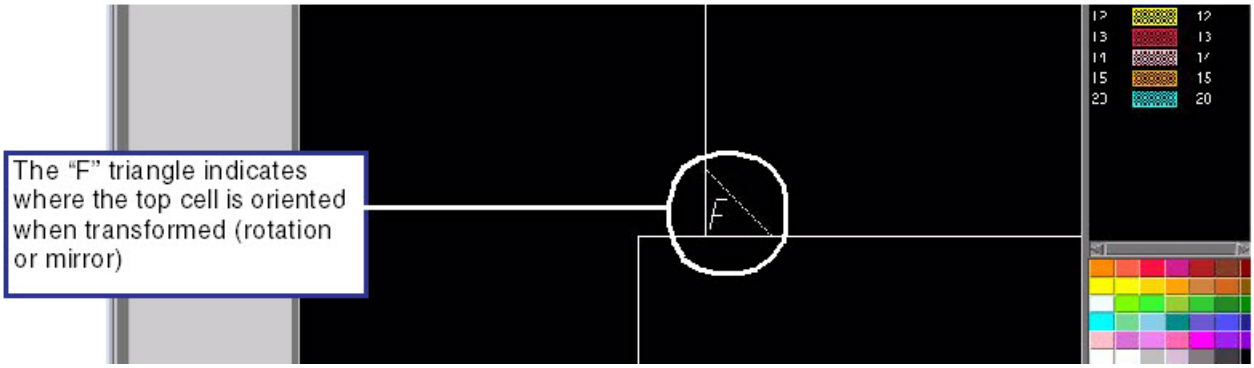
Procedure

1. To enable orientation markers:
2. In Calibre MDPview, select **Options > Layout View**. The Preferences dialog appears.
3. In the Preferences dialog, click the **View** tab (if it is not currently displayed).
4. In the **View** tab, click **Draw orientation markers of references**.

Results

The top cell level (level 0) is redrawn with the “F” triangles.

Figure 2-6. Cell Orientation Markers



Control Which Groups of Fractured Data You See

For fracture formats like MEBES that represent fractured data using a proprietary organizational structure, you can control which groups of the pattern data you see by setting the MDP mode.

Note

Flat data such as pattern files and data produced from fracturing has limited support in the Calibre MDPview display. You cannot edit the data, and any modifications made are not saved.

The Calibre MPDview application supports the modes shown in the following table. Set the MDP mode either through the menus by using **View > MDP mode**, or by using the Alt key plus a number key (1-6).

Table 2-4. Controls for Pattern Data Visibility

Visible Data	Controls
Extent Layout extent only.	Alt+0

Table 2-4. Controls for Pattern Data Visibility (cont.)

Visible Data	Controls
All geometry data (default)	Alt+1
All geometry + boundary This is all geometry and hierarchy elements. The hierarchy elements vary depending on format: • MEBES: Segment and stripe boundaries • JEOL: Field boundaries • Hitachi: SSF boundaries • OASIS, Micronic: layout hierarchy; depth can be controlled with the standard viewer depth controls.	Alt+2
Boundaries Hierarchy elements only.	Alt+3
Style4 Style5 These styles depend on the format: • MEBES: View individual geometry only (Style4) or array geometry (Style5). • JEOL: View LB (Style4) or non-LB geometry (Style5).	Alt+4 Alt+5
Style5: Arrayed Data (MEBES only)	Alt+5
All + extent	Alt+6
Color Extent By Ref Color Extent By Chip These options cause either Extent or All+Extent to display extents in layer colors or chip colors, respectively. Note that when displaying the extents by chip color, they are rendered with the layers and can, therefore, be covered by objects rendered on layers above any layer with extents. Displaying by reference color renders the extents on top of all layers as before.	—
Cross This draws a cross at the center of polygons that are otherwise too small to see at high levels.	—

Table 2-4. Controls for Pattern Data Visibility (cont.)

Visible Data	Controls
Missing Chip Markers This draws markers for any missing chip definition files.	—

Note

Effects resulting from changing the MDP mode are not visible unless the view depth is set so that the pattern file data is visible (the depth is set > 3).

Note that the settings from **View**> **MDP Mode** are saved in the Preferences file any time an option changes.

Control Which Cells Are Displayed

You can perform visual cell pruning (hiding certain cells from display in order to speed up redraw times) using two different methods.

- **View > Selective Cell Display** menu option.

  Selecting this option brings up a dialog box showing the list of all the cells in the currently displayed layout. Select a cell name and click the right arrow button to move the cell to the Hidden side of the dialog box. Click **Update Display** to apply the change.

- Right-click a cell name in the Cells browser and select **Hide Cell Instances** from the popup menu to hide all instances of the selected cell. You can later select Show Cell Instances in the same way to re-display the hidden cells.

Display Chip File Names in the Layer Palette

If you select View > MDP Chip View, you can display chip file names displayed as colored layers in the Layer Palette.

There are two different modes available, which you can pre-select from **Edit > Preferences > Session tab > MDP chip view**:

- Color By Chip — Displays chip file names in the Layer Palette for better identification.
- Color By Placement — Treats unique job deck placement definitions as unique chips and assigns them separate colors to add a visual distinction between placements, the Color By Placement mode. This mode also causes the **jobToOasis** command to generate unique chips in the OASIS file, thus producing large files. To avoid this, load the job deck using Color By Chip instead before running **jobToOasis**.

- **Color By Definition** — Causes all layers within each job deck chip definition to be colored the same. The same color is assigned to both the level layers and their contained chip layers in the level and chip views. A single MDP job deck can be loaded with this option. View modes can only be changed by closing the layout and reloading with a different mode setting.

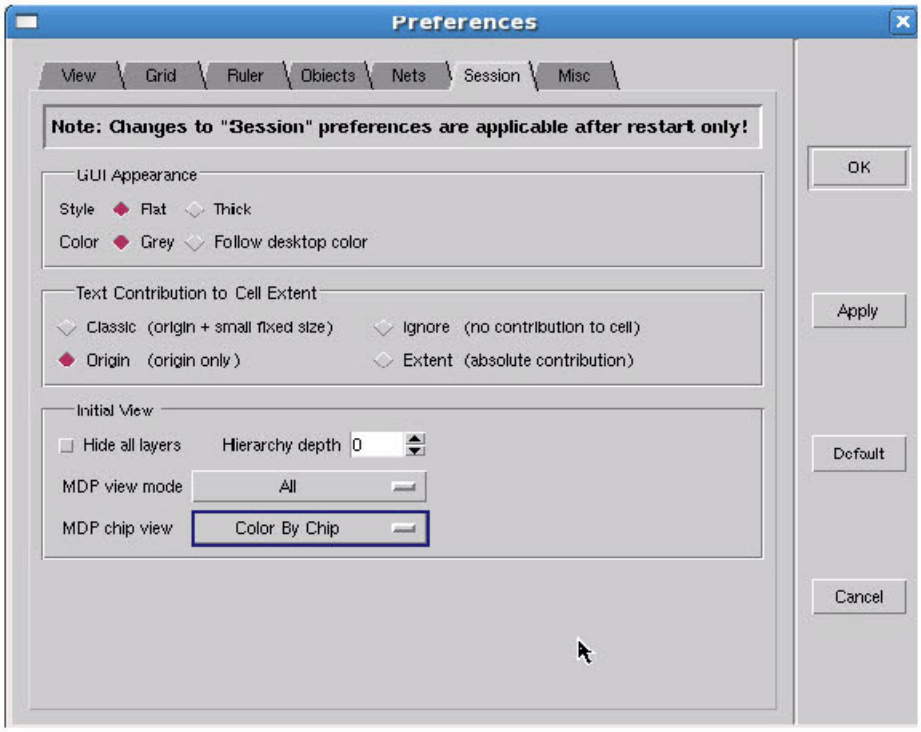
  Note

  This option modifies the internal layout database in the same manner as Color By Placement. Therefore, writing layouts from a database formatted for this type of viewing do not result in an optimized layout. Databases should always be loaded in the Color By Chip mode when writing the data in layout format.

  Running the **Layer > Reassign Colors > By MDP Chip Definition** restores the Color By Definition colors to all layers.

  All layers in the Color By Definition mode use data type assignments to ensure the layers are unique within each chip definition. This guarantees the colors are unique across all defined layers within the definitions, regardless of whether the definitions share level layer numbers. The datatypes are assigned by the order of the definitions listed in the job decks. If the ordering changes, the assigned datatype changes.

Figure 2-7. MDP Chip View Mode Setting



For job decks, layer filters are automatically generated for the chip view mode that corresponds to each level layer. The filter name is the level layer number. Each filter holds the chip layers contained within the level layer.

Note

Layers that do not receive chip file names are true layers containing non-chip information (such as MEBES labels). These layers correspond to levels. Hiding the levels causes the associated chips to be removed from the display.

Chip layers cannot be unhidden unless the level layer that contains them is also unhidden.

Display the Layout Origin

You can mark the layout origin point (0,0) by toggling the Marker at Origin setting (found in the **Options > Layout View** dialog box).

- None (default) does not print an origin marker.
- Cross prints a cross-shaped marker at the origin.
- Square prints a square box at the origin.

Note

If your layout does not lie near the origin (0,0) coordinates, the origin marker may not be visible when you execute a Zoom All command.

Draw Crosses for MDP Geometry Objects

When drawing crosses on MDP geometry objects, Calibre MDPview allows you can add, track, and remove selected layers from a set of layers for cross-drawing.

Select **MDP Mode > Cross** to activate or deactivate cross drawing for all layers. If you want to activate or deactivate cross drawing on individual layers, place the cursor over the layer palette and select the right-mouse-button popup menu option **Toggle Cross**. Cross-enabled layers are indicated by drawing a plus sign (+) in the layer palette next to the layers color/stipple pattern.

Select **View > MDP Mode > Cross** to activate (Show Cross) or deactivate (Hide Cross) redraws with all crosses when no Show Cross settings exist. This causes a redraw with only the chosen cross layers when they do exist.

Disabling Cross mode causes a redraw with no crosses drawn, but maintains individual layer cross settings. In the absence of any Show Cross settings (activated or deactivated), enabling Show Cross causes a redraw of crosses on the specified layer.

Each subsequent Show Cross redraws the specified layer, and adds it to the set of cross layers. A Hide Cross causes a redraw with the specified layer removed from the set of cross layers. A Hide Cross action that removes the last layer from the cross layers set deactivates the **MDP Mode > Cross** and causes a redraw of all layers.

Cross Drawing Using the Layer Properties File

You can also control cross drawing in the layer properties (.layerprops) file using the crossLayers command.

The basic syntax is as follows:

crossLayers [-show *layerlist*] [-enable]

where at least one of the following must be present:

- -show *layerlist*: Turns on cross drawing for the specified layers.
- -enable: Turns on crosses for either -show *layerlist* or all layers if -show *layerlist*is not specified.

It is recommended that this specification be used in layout-file-specific layer properties files. However, *layerlist* layers that do not exist in the layout are ignored by the layer properties file method. Saving a layerprops file with cross layers previously defined or enabled in Calibre MDPview generates an appropriate crossLayers command.

Refer to the *Calibre DESIGNrev Layout Viewer User's Manual* for information on the layerprops file.

The following examples show four different use scenarios:

```
crossLayers -enable
```

This example draws crosses on all layers.

```
crossLayers -show 1 2 6
```

This example sets cross drawing for layers 1, 2, and 6, but not enable cross drawing. These layers draw crosses when enabled from the GUI.

```
crossLayers -show 1 2 6 -enable
crossLayers -enable -show 1 2 6
```

These examples set cross drawing for layers 1, 2, and 6, and enable and draw crosses on those layers.

Find Missing Patterns in Shared Reticles

If you have a shared reticle (where several different chips are in the same reticle), if patterns are missing, it may be difficult to find out the specific patterns. You can search for these missing

patterns using the **View > MDP Mode > Missing Chip Markers**. This option draws markers for any missing chip definition files.

Calibre MDPview missing chip markers typically appear as X-crosses, two times larger than MDP crosses (invoked from **View > MDP Mode > Cross**).

The **Missing Chip Markers** option shows missing patterns only in the visible and enabled layers.

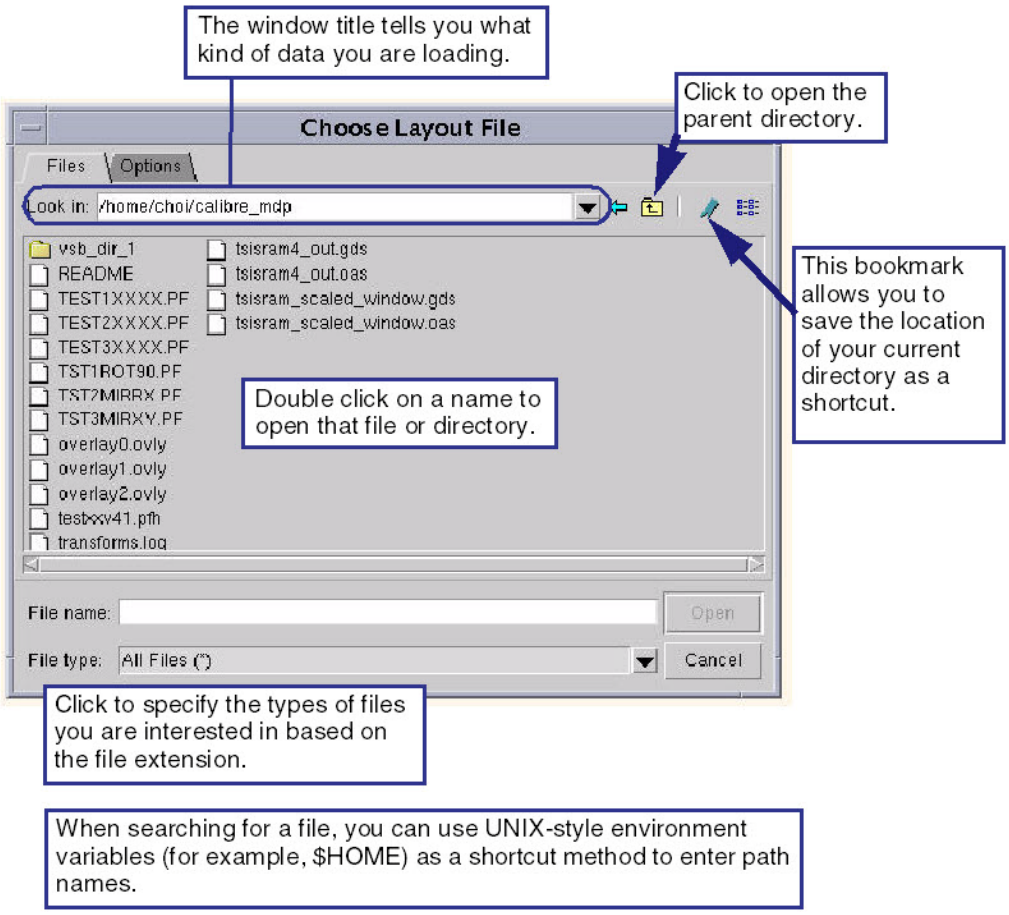
If there are missing patterns in a layer that is hidden or switched off in the Layers pane, the corresponding missing chip markers are not shown.

Chapter 3
Load Data

Loading pattern file or database data causes that data to be displayed, making it the *current layout*[1]. Any previously loaded layouts remain available, though they are not visible, and therefore not current.

The following figure shows the basic file navigation scheme.

Figure 3-1. File Navigation Explained for New Users



1. In the Calibre MDPview data viewer, each data file, pattern file, or database, is referred to as a layout.

Load Pattern Files

The Calibre MDPview data viewer supports loading and viewing pattern files in the following industry-standard fracture formats: MEBES, Hitachi, JEOL, Micronic, Nuflare, and OASIS.

Because pattern files are typically large, the Calibre MDPview data viewer does not load the data into memory when it displays pattern files for viewing. Instead the viewer leaves the data on disk, sampling it as necessary to change views. Also, the viewer cannot load compressed pattern files. (Database data, on the other hand, can be loaded directly into memory, even if it is compressed.)

Table 3-1. Level Of Support for Pattern File Data

File Type	Options	Menu Path
MEBES	single file	**File > View Mask Pattern > MEBES**
	job deck	**File > View Mask Job Deck > MEBES**
	flexible job deck	**File > View Mask Job Deck > Flex. MEBES**
JEOL	single file	**File > View Mask Pattern > JEOL**
	job deck	**File > View Mask Job Deck > JEOL**
Hitachi	single file	**File > View Mask Pattern > HITACHI > View**
	job deck	**File > View Mask Job Deck > HITACHI**
Nuflare (VSB)	single file	**File > View Mask Pattern > VSB**
	job deck	**File > View Mask Job Deck > VSB**
Micronic	single file	**File > View Mask Pattern > MICRONIC**
	job deck	**File > View Mask Job Deck > MICRONIC**
OASIS	single file	**File > View OASIS Layout**

Note

Calibre MDPview does not support the following Micronic data types: circles, arcs, inverted geometry types, and open-ended figures.

Load OASIS Indexed Files

OASIS files in extended MEBES job decks can be “pre-indexed” to speed job deck loading. Typically, you would use the -v option when invoking Calibre MDPview to load an OASIS layout file as a view-only, disk-resident, indexed file

For more information on indexing, refer to “OASIS File Indexing” on page 129.

However, you can also specify -v without an OASIS file to load multiple OASIS files as view-only disk-resident files within the Calibre MDPview session. This places Calibre MDPview into the disk-resident OASIS mode for any other OASIS file reads.

If you want to place Calibre MDPview into the disk-resident OASIS mode and still load an OASIS file at startup, specify “calibrempdv -v -m <*layout*>.*oas*”. The mode is set before reading the -m specified file. For example:

```
calibremdpv -v –m file1.oas –m file2.oas
```

Note

If you are using **View OASIS Layout** to view small-shaped, sparse layers, always create index files using the -noView option, or load the small-shaped, sparse layers into memory. This helps to prevent small shapes on sparse designs from disappearing at higher hierarchical levels. If small shapes disappear at higher hierarchical levels in other modes, you can use RVE to scan through the geometries.

Create Overlays With OASIS Indexed Files

For overlays created with disk-resident OASIS indexed files, simply add -v to the startup switches and load the overlay with the -o option or through the GUI.

For complete information on overlays, refer to the *Calibre DESIGNrev Layout Viewer User’s Manual*.

The following example creates an overlay file using indexed OASIS files and a standard GDSII file as input, and it then opens the resulting overlay file.

1. Load the GDS and OASIS files.

   ```
   calibremdpv -v –m a.gds –m b.oas –m c.oas –v
   ```

2. To save the overlay, in Calibre MPDview, select **File** > **Save As** > gds_oas.ovly.
3. To load the overlay file:

   ```
   calibremdpv -o gds_oas.ovly -v
   ```

Note

Disk-resident OASIS files have different top cell names from normal memory-resident OASIS files. These cell names are saved in the overlay file, and the overlay files are not interchangeable between file read mechanisms. For example, overlays created from memory-resident OASIS files can only be loaded in the normal OASIS read mode, and overlays created from disk-resident OASIS files can only be loaded in the disk-resident OASIS read mode.

View Job Decks

Once loaded into the Calibre MDPview application, job deck data behaves as if it were a single file. As with viewing a single pattern file, you typically see the design as white bounding boxes; however, in Job View the lines show the individual pattern files that were loaded in.

You can view the layers in a job deck in either of two modes:

- **Default mode** — Each layer in the job deck is displayed as a single color. In this mode the entire mask layer appears as a single layer in the viewer.
- **Chip mode** — The layers in the job deck are expanded into a set of layers, one for each chip. In this mode layer 2 in chip 1 is displayed as a different viewer layer than layer 2 in chip 2.

Note

Disk-based viewing including View OASIS Layout, job deck viewing, and pattern file viewing do not support new layer generation inside the file, as these are “view only” features. Patterns from job decks are not loaded into memory.

You can switch between modes using Ctrl key + comma (,). When you use this methods, Calibre MDPview also switches the color mapping between level-color mapping and chip-color mapping for a single job deck layout view.

The tables in this section list supported job deck commands (refer to the manufacturer’s documentation for a full description of these commands). Commands not listed in these tables are NOT supported. If you attempt to load a job deck containing unsupported job deck command, the application prints a warning message to the terminal shell window. If you attempt to load a job deck pointing to nonexistent pattern files, the application prints a warning message to the terminal shell window and displays the existing pattern files.

Note

An error message is printed to standard output if a job deck defines a nonexistent pattern file, but ignores all references to the file later. This behavior may be changed for MEBES job decks only by setting the variable EJOBDECK_NOCHIP_SEVERITY to a value of ERROR in the *.signaext* file; the default value is WARNING. In this case, the job deck is not successfully loaded, and the calling function fails. Refer to “Calibre MDPview Tcl Extensions” on page 143 for a complete description of the *.signaext* file.

Table 3-2. MEBES Supported Job Deck Commands

SLICE (Type One only) IDstring Slice Code — slice 22 supported for 7.25 rounded substrate WIDTH REFLECT OPTION A/AD C/CL D/SF M O/AO X/XC Y/YC R Z/VA N P T F OFID/FID	SCALE address size demagnify factor CHIP I M U X AD AO CL XC YC AD R O Y VA=1 A (all) UX UY	<eoj> and <EOJ> <boj> and <BOJ> MTITLE ITITLE DTITLE STITLE ORIENT ROWS GROUP END
Additional MEBES Commands Supported in Extended MEBES Job Decks Only		
TC SF BX BY	UX UY AG LY	DT

Table 3-3. Hitachi Supported Job Deck Commands

Lot Data File	Array File
mask	macro_chip_management
y_mirror	matrix_define
array_file	macro_chip_define
version	chip_set
version_no	macro_position
revision_no	macro_matrix
plate_size	macro_array
	comment_define
	date, position, type
	mark, position, type
	serial_no, position, type
	indication
	layer
	title, name, position
	select, date, mark, serial_no

Table 3-4. JEOL Supported Job Deck Commands

Common Block Commands	Layer Block Commands	Termination Commands
JOB	LAYER	END
jobdeck name	MIRROR	
workpiece size	SCALE (sx = sy)	
OPTION	P(chip definition)	
mirror chip position	SKIP	
SF	DEVINF	
PATH	SPPRM	
proximity correction	CIRCLE	
smash writing	C	
e_beam writing	CHAR	
path name		
absolute path		
PATH /P/S		
ARRAY		
ASSIGN		
chip or array assignment		
to points of array		
AEND		
PEND		
CHAR		
C		

Table 3-5. VSB Supported Job Deck Commands

layout.ini	chip.ini	Draw Script
layout_name script	chip_name chip_dir chip_data_format	layout_name mask_size (vsb-11) mask_size_x (vsb-11) mask_size_y(vsb-11) units_length mask_mirror CHIP_INI_SEARCH_PATH CS_DEVICE_NAME CS_ID_MIRROR CS_ID_INVERT CS_ID put multiput matrixput draw xdraw

Table 3-6. Micronic Supported Job Deck Commands

Header Section	Pattern Section	End Section

Table 3-6. Micronic Supported Job Deck Commands (cont.)

#MASKSETF1	ARRAY ARRAYSTAMP PREPAREDCELL AX AY DM DP EC EF EP FO MP PF R SF SH X RTITLE MX MY R ROT TX PAUSE	END
Options Section		

Table 3-6. Micronic Supported Job Deck Commands (cont.)

ALIGNMENT	DISTORTIONMAP	PLATECORNERREF
ALPHA	FACEDOWN	PREPLOGID
BRIDGEALIGN	FLYMEAS	PROCESS
CDCDENSITY	JOBLIB	SORTEDLIB
CDCGLOBAL	LAYERS	UNIT
CDCSITE	MAGAZINE	
CDMAP	NEGRESIST	
CORNERENHANCEMENT	OPTION	
	ZX	
	ZY	

Table 3-7. MALY Job Deck Commands

Command	
BEGIN MALY	ROOT
END MALY	MASKMIRROR
BEGIN MASKSET	BEGIN TITLE
END MASKSET	END TITLE
BEGIN CMASK	DATE
END CMASK	SERIAL
BEGIN MASK	STRING
END MASK	BEGIN STRGROUP
BEGIN PARAMETER	END STRGROUP
END PARAMETER	SREF
MASKSIZE	AREF
FONT	PROPERTY (the DNAME, MNAME, ENAME is displayed along with the TOP cell and the location)
BASE	
ARYBASE	
The MALY job deck currently only supports OASIS-type chip placements.	
The following sub-commands are currently not supported: ROOT NATIVE (you can specify OASIS.MASK or NATIVE with the ROOT record), FONT NATIVE (currently, only the STANDARD font is supported), and REFERENCE TOOL.	
The size of SERIAL and DATE strings for the MALY job deck can be adjusted using .signaext variables. See “Control Size of MALY SERIAL and DATE Strings” on page 150 for information.	

VSB Job Decks

When viewing VSB job decks (Nuflare formats VSB11, VSB12, VSB12i, and NUFLARE_MBF), you must use an absolute path for accessing pattern files in other locations for the job view. By default, a VSB job deck is represented by one directory. Under this directory are the job deck script files *layout.ini*, *chip.ini*, *layout* directory, and the chip directories. The draw script files are under layout directory.

The chip data files are under their corresponding chip directories; the script in *layout.ini*specifies the draw script file name without any other path, and the chip_dir in *chip.ini*specifies chip directory name without any other path. The Calibre MDPview job deck viewer automatically finds the correct path for its data. If job deck files are not organized accordingly, absolute full path names must be specified for the script entry in *layout.ini* and the chip_dir entry in *chip.ini* to let the VSB job deck viewer locate the desired data.

For VSB12i job decks, as the VSB12i format is not always a single-layer format; each trapezoid may have Attribute Information associated with it. The Attribute Information is assigned at the time of fracture. When displaying VSB12i format data that contains attribute information, datatypes are added to the layer palette and geometries with different attributes display on those separate layer or datatypes. This allows you to see if the Attribute Information was assigned correctly, displaying trapezoids with different Attributes on different layers in the layer palette.

Flexible Job Decks

The MEBES flexible job deck is a job deck written using the MEBES commands that can reference pattern files in MEBES, JEOL, Hitachi, or VSB format, as well as GDSII and OASIS formatted data. Flexible job deck functionality allows you to create a MEBES type job deck that describes a mask composed of a mixture of any of these pattern files.

A MEBES flexible job deck supports all MEBES job deck commands extending the pattern file in CHIP commands to be any readable MDP format file (MEBES, JEOL, VSB, and OASIS).

Note

OASIS files used in the flex job view must comply with the format requirements for extended job decks, as described in the section “MEBES Job Decks Referencing GDSII and OASIS Data” on page 56.

Example

The following is an example flexible job deck:

```
SLICE FLEXTEST,17
DTITLE  1,FLEXIBLE JOB TEST
OPTION R1,R3,M
CHIP <eoj>,(1R,TEST1XX-XX-PF,AD=.5) *NO REVTONE
$           (2R,TEST1XX-XX-PF,AD=.5) *REVTONE
$           (3R,TEST1XX-XX-PF,AD=.5) *NO REVTONE
ROWS 38100/25400,3,25400
*
CHIP REVI, (1,testn_s0.j309000mv) *REVTONE * JEOL
$           (2,d03,AD=.002) *NO REVTONE * VSB
$           (3,TEST1XX-XX-PF,AD=.5) *REVTONE
ROWS 25400,3,25400/63500
*
   END
```

MEBES Job Decks Referencing GDSII and OASIS Data

Calibre MDPview allows you to load extended job decks, which are MEBES job decks that reference MEBES data and GDSII or OASIS formatted data.

Extended job decks allow you to view how your original data would appear after transformations (mirroring, rotation, magnification) without having to fracture the data.

Extended job decks are subject to the following requirements within the MEBES syntax:

- All data in the GDSII or OASIS file must be within the window extent as printed.
- Specify only layers and datatypes that are to be printed on the mask. The maximum number of layers and datatypes is 65,536.
- The GDSII or OASIS filename must be 12 characters, where the 10th character is a period (.). All other characters must be uppercase alphas (A-Z) or numbers (0-9).
- Specify the TC job deck command if the GDSII or OASIS file contains more than one top-cell name, otherwise all cells are printed.
- Boolean or sizing functions are not supported.
- Mirroring is supported about the y-axis only.
- The GDS or OASIS file specified in the job deck uses the same syntax as a MEBES file name.

The supported job deck options (level 2 CHIP) are listed in the following table.

Table 3-8. Supported Type 2 CHIP Options MEBES Job Decks

Type	Description	Required?
AG	Specifies angle of rotation. The angle of rotation can be 0, 90, 180, or 270. If omitted, the angle is 0.	No
BX	Specifies bottom-left X coordinate of pattern (in microns) after scale factor is applied.	Yes
BY	Specifies bottom-left Y coordinate of pattern (in microns) after scale factor is applied.	Yes

DT	Specifies data types set to be printed on the mask. If omitted, all data types in the GDS or OASIS file are printed on the mask. For example:
	`DT={12, 15}`
	`// Only print data types 12 and 15 on the mask`
	Note: The set of layer and datatype pairs is the Cartesian product of the sets of layers and datatypes. For example, LY = {1,3,5} and DT= {2, 0} produces the set of layer-datatype pairs {(1, 2), (1, 0), (3, 2), (3, 0), (5, 2), (5, 0)}. You cannot specify the DT option without specifying the LY option. You can optionally specify BX, BY, UX, UY for OASIS placements inside a MEBES job deck to place a partial (clipped) region. Calibre issues either a condition of IGNORE, WARN, or ERROR depending on the EJOBDECK_CLIPPING_SEVERITY setting.
		No

Table 3-8. Supported Type 2 CHIP Options MEBES Job Decks (cont.)

Type	Description	Required?

LY	Specifies layer set to be printed on the mask. For example:
	`LY = {1-3, 5}`
	`// Only print layers 1-3 and 5 on the mask`
	You can also specify a datatype associated with a layer using the LY syntax, separated by a semicolon (**;**). For example:
	`LY={1, 2-4, 5;10}`
	This example shows layer L1 printed with all datatypes, the range of layers 2 to 4 printed with all datatypes, and layer 5 printed with datatype 10.
	You can also specify an explicit mapping from a layer to a datatype using LY. In this case, you cannot specify data types using DT. For example:
	`LY={1;1 1;2 2;2 2;3}`
	This example prints only layer-datatype pairs 1.1, 1.2, 2.2, and 2.3 on the mask.
	Calibre also supports both the space and comma as separators between multiple layers (or pairs). If you specify datatypes using LY, you cannot use DT.
		Yes

TC	Specifies the top cell name. Only the specified top cell data is printed on the mask. For OASIS placements, if no TC is provided, the TOP cell from the corresponding index file in picked and it is printed on the mask. For GDS placements, the TC option is not supported and all TOP cells are printed on the mask. The following is an example of TC:
	`TC=PUB26`
	`// only print top cell PUB26 on the mask`
		No

UX	Specifies upper-right X coordinate of pattern (in microns) after scale factor is applied.	Yes
UY	Specifies upper-right Y coordinate of pattern (in microns) after scale factor is applied.	Yes

Note

Calibre MDPview cannot load MEBES job decks with level numbers exceeding 1024. If you attempt to load a MEBES job deck exceeding this number, Calibre MDPview issues an error message “Level number can not be more than 1024 at line *x*” and exits.

The following example demonstrates the CHIP command for a GDS or OASIS file:

```
CHIP OASIS,(1,TESTXXX-XX-XX,TC=TOP,LY={1,10},BX=0.0,BY=0.0, \
   $ UX=5000.0,UY=5000.0,AG=90)
```

Load MEBES, VSB, or JEOL Job Decks With Remote Pattern Files

Normally, loading a job deck through either **File** > **MEBES** > **Job View**, **File** > **View Mask Job Deck** > **VSB**, or **File** > **JEOL** > **Job View** requires the pattern files referenced in a job deck to exist in the directory containing the job deck file.

You can load job decks with reference pattern files in other directories by defining a search path containing the paths to the locations where a pattern file may be found. You define this search path in the *.signaext* file, using the application variable MDPDataSearchPath:

```
set ::MDPDataSearchPath /full/path/to/pattern/files
```

The paths specified must be full pathnames or paths relative to the current working directory where Calibre MDPview is invoked.

You can define multiple file locations by creating a colon (:) separated list of locations, as in the following example:

```
set ::MDPDataSearchPath pathname_1:pathname_2...:pathname_n
```

Additionally, if the tool encounters multiple occurrences of the *identical* file when searching, then Calibre MDPview takes the first occurrence of the file.

You can alternatively set the search path in your *.signaext* file by entering the MDPDataSearchPath command in the same TCL window used to invoke Calibre MDPview. The search path is then used the next time you open a job deck for viewing in that Calibre MDPview session.

Pattern File Name Recognition in MEBES Job Decks

Calibre MDPview follows a specific process and rules for a loaded MEBES job deck in order to recognize corresponding pattern file names.

Calibre MDPview uses the following process to identify chip names and matching them to pattern file names:

1. Calibre MDPview matches the case of pattern files in the job deck.
2. If the pattern file names are in upper case in the job deck, it matches the names to upper case pattern files on the disk.

3. If the pattern file names are in lower case in the job deck, Calibre MDPview first matches lower case pattern file names on the disk.
4. Calibre MDPview searches for upper case pattern file names on the disk.
5. If the pattern files are of mixed case, the job deck will only match the mixed case names in the job deck. If the name does not match exactly, it issues a warning and does not load the pattern file to the viewer for the job deck displayed. Failure to match mixed-case pattern names between the disk and job deck can also cause the jobToOasis utility treat themas missing patterns and drop them entirely.

Extended MEBES Job Deck Pattern File Search Priority

Extended MEBES job decks support GDS or OASIS data along with MEBES pattern files. MEBES chip names in a job deck are typically a 7-2-2 character format (for example, *TESTXXX-XX- XX*). When searching for a pattern file with a particular MEBES chip name, the priority order is followed where the MEBES name is in 7-2-2 format:

1. Search for a file name with eleven characters followed by the suffix, 11.<*suffix*> (for example, *TESTXXXXXXX.oas*).
2. If the file is not found, search for a file name using a 9_2.<*suffix*> format (for example, *TESTXXXXX_XX.oas*).
3. If the file is not found, search for a file name using a 9.2.<*suffix*> format (for example*TESTXXXXX.XX.oas*).
4. If the file is not found, search for a file name using a 9.2 format (for example *TESTXXXXX.XX*).
5. If the file is not found, search for a file name using a 9_2 format (for example, *TESTXXXXX_XX*).
6. If the file is not found, search for a file name using a 7-2-2 format (for example, *TESTXXX-XX-XX*).
7. If no file is found, the name referenced is considered as a missing chip

Steps 1, 2, and 3 are ignored if EJOBDECK_PATTERN_SUFFIX is not specified. If the MEBES name is an arbitrary name (anything other than 7-2-2 format), Calibre searches for the arbitrary name as it appears on the disk. The tool either issues message of IGNORE, WARN, or ERROR depending on the EJOBDECK_NOCHIP_SEVERITY setting.

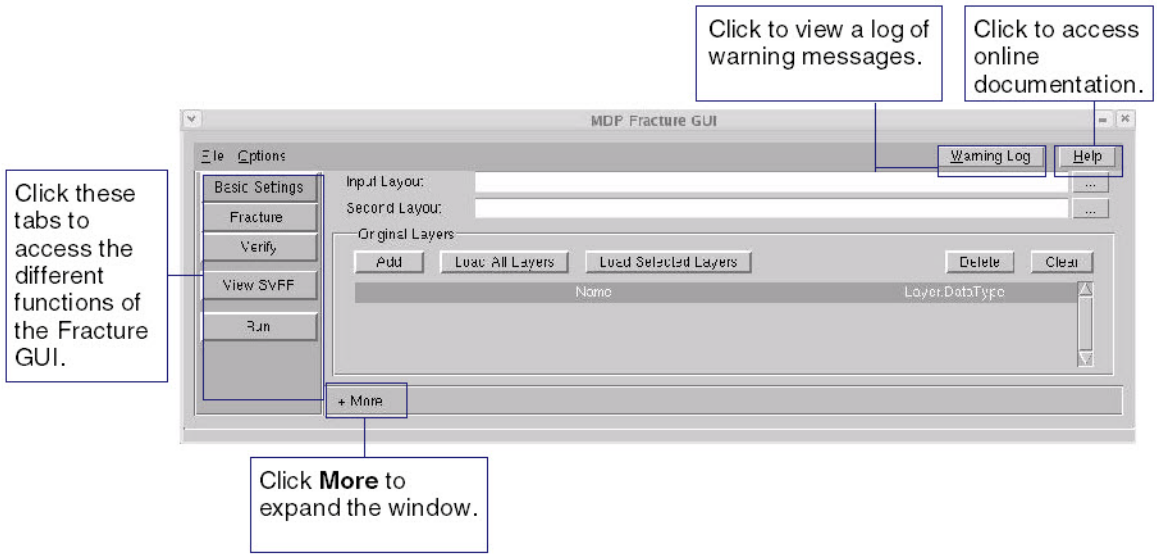
Chapter 4
Calibre FRACTURE Operations With Calibre MDPview

The Calibre FRACTURE operation converts layer data into MEBES, JEOL, Hitachi, Micronic, Nuflare, and OASIS formatted data.

You define a rectangular section of a Calibre polygon layer as input to the Calibre FRACTURE Standard Verification Rule Format (SVRF) operation. The polygons on the layer are optionally modified and converted to trapezoids. The trapezoids are then transformed to new coordinate and unit systems, and rearranged into the hierarchy allowed by the requested format. These trapezoids are finally output in the fracture format.

You can perform the FRACTURE operations using Calibre MDPview by accessing **Tools > MDP Fracture GUI**. The MDP Fracture GUI appears as shown in Figure 4-1:

Figure 4-1. MDP Fracture GUI



Fracture Operations in Calibre MDPview

With the MDP Fracture GUI, you can a number of FRACTURE-related tasks.

These include the following:

- Create an SVRF rule file and run MDP FRACTURE for your particular format (MEBES, JEOL Nuflare, Micronic, Hitachi, or OASIS and automatically run Calibre MDPverify afterwards.
- Run Calibre MDPverify on two previously fractured layouts.

Fracturing a Database and Verifying the Results

Both Fracture and Verification can be performed through the MDP Fracture GUI.

Procedure

1. Choose **Tools > MDP Fracture GUI**.
2. In the MDP Fracture GUI window, select one of the following from the **Options** menu:
   - To run both Fracture and Verify, select **Options > Run Mode > Run Fracture and Verify**.
   - To run Fracture only, select **Options > Run > Run Fracture Only** (this greys-out the **Verify** tab).
3. On the Basic Settings Tab specify the following at a minimum:
   - Input layout database (GDS or OASIS).
   - Layers from the input layout database.

   If you intend to load selected layers or all layers from the input layout, you must load a layout database into Calibre MDPview.

   You can optionally specify derived layers, layout parameters, output parameters, and other flags. Click **More** to reveal the additional options and **Less** to hide them.
4. Click the Fracture Tab and specify the following at a minimum:
   - Fracture Format (such as for Hitachi, JEOL, MEBES, Nuflare, Micronic, or OASIS) and click **Properties**.

Note

The MDP Fracture GUI does not support the VSB12i, VSB12_V2, or NUFLARE_MBF formats.

In the format-specific dialog box, fill in the appropriate format-specific options (such as for Hitachi, JEOL, MEBES, Nuflare, Micronic, or OASIS) in the format dialog box, then click **OK**. These options correspond to Calibre FRACTURE keywords and are described in the Calibre FRACTURE chapter of the *Calibre Mask Data Preparation User's and Reference Manual*.

- Input layer (from the original input layout file).
- Fracture Region. Specify one of the following:
  - **Layer**: The entire layer is processed.
  - **From Layout**: A set of coordinates defines the region to be fractured. You can also click the **Area** button to automatically select the coordinates of the region currently displayed in Calibre MDPview. However, to do this, you must have a layout already loaded into Calibre MDPview.
  - **Extent**: Only the layer extent is processed.

5. You can automatically verify the fracture file against the original layout database by clicking the **Post Fracture Verify** radio button. This is equivalent to running a Calibre MDPverify fracture file comparison to a database (such as MEBES2DB or HITACHI2DB). If you do not select this option, you can still manually define a Calibre MDPverify run in the **Verify** tab, or simply not execute a verification pass.
6. Click View SVRF to invoke a text viewer with the generated SVRF code displayed. This creates an SVRF file with the same name as the layout's name appended with .svrf. The file is saved by default, in the directory from which Calibre MDPview was invoked (unless otherwise specified).
7. Click **Run** to execute the fracture run.
8. You can save your settings at any time by selecting **File > Save Session**, then reload it later by selecting **File > Load Session**.

Results

A fracture file is produced that has been verified.

Compare Regions for Two Fractured Files

You can compare verify regions on two fracture files in the MDP Fracture GUI.

Procedure

1. Load the input layout file into Calibre MDPview.

2. Select **Tools > MDP Fracture GUI**.
3. In the MDP Fracture GUI window, select **Options > Run > Run Fracture Only** (this greys-out the **Fracture** tab).
4. Click **Basic Settings** specify the following at a minimum:
   - Input layout database (GDS or OASIS).
   - Layers from the input layout database.

   If you intend to load selected layers or all layers from the input layout, you must load an input file into Calibre MDPview.

   You can optionally specify derived layers, layout parameters, output parameters, and other flags. Click **More** to reveal the additional options and **Less** to hide them.
5. Click the Verify Tab. Specify the following at a minimum:
   - Verify Type: The type of comparison to be performed, such as MEBES2MEBES, VBOASIS2JEOL, JEOL2HITACHI, and so on. These are described in the Calibre MDPverify chapter of the *Calibre Mask Data Preparation User's and Reference Manual*.
   - An input layer from the input layout database.
   - File1: The first pattern file to be compared.
   - Verification Region: Specify one of the following:
     - **Layer**: The entire layer is compared.
     - **From Layout**: A set of coordinates defines the region to be compared. You can also click the **Area** button to automatically select the coordinates of the region currently displayed in Calibre MDPview. However, to do this, you must have a layout already loaded into Calibre MDPview.
     - **Extent**: Only the layer extent is compared.
     - **None**: No region is defined.
   - Click the **More** button to optionally specify transform options such as Mirror, Rotate, Magnify, Shift, reverse tone, as well as output parameters. These are described in the Calibre MDPverify chapter of the *Calibre Mask Data Preparation User's and Reference Manual*.
6. In the **Verify** tab, click the **Fracture File 2** button and specify the second fracture file you wish to compare. You can also optionally specify transform options.
7. Click View SVRF to invoke a text viewer with the generated SVRF code displayed. This creates an SVRF file with the same name as the layout's name appended with .svrf. The file is saved by default, in the directory from which Calibre MDPview was invoked (unless otherwise specified).

8. Click **Run** to execute the Calibre MDPverify run.
9. You can save your settings at any time by selecting **File > Save Session**, then reload it later by selecting **File > Load Session**.

Results

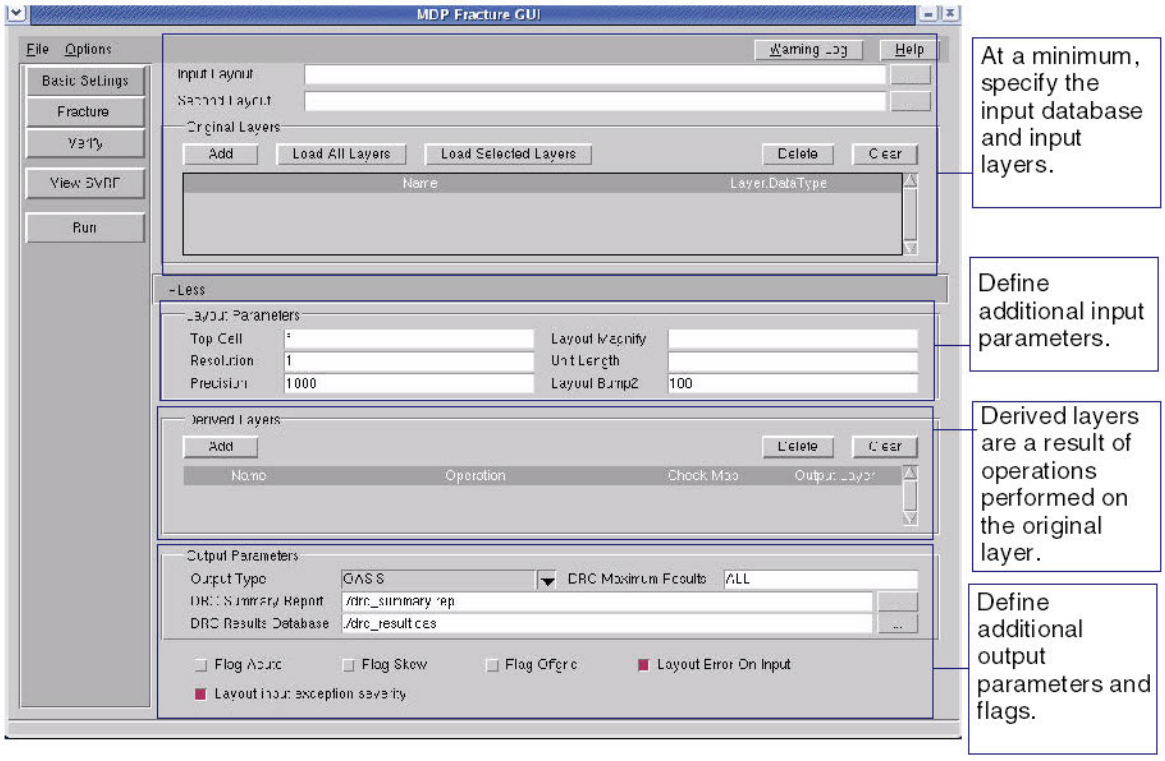
The Calibre MDPverify tool verifies the two specified areas in the two fracture files.

Basic Settings Tab

To Access: **MDP Fracture GUI** > **Basic Settings** button

The Basic Settings tab allows you to define the original and derived layers with the output parameters as well as required user inputs.

Figure 4-2. Basic Settings Tab



Objects

Table 4-1. Key Features of the Basic Settings Tab

Option		Description
Required		
Original Layers		
	Add	Defines an input layer.
	Load All Layers	Loads all the layers defined in the layout loaded in Calibre MDPview.
	Load Selected Layers	Loads selected layers only (layers highlighted in the Layers Palette of the Calibre MDPview window).
	Delete	Deletes a specified layer.
	Clear	Clears all layers.

Table 4-1. Key Features of the Basic Settings Tab (cont.)

Option		Description
Optional		
Layout Parameters		
	Top Cell	Specifies the top cell name (* is default).
	Resolution	Defines the layout grid step size. When only one value is supplied, the grid step size is the same in both the x and y direction. The default value is 1.
	Precision	Defines the ratio of database units to user units. Precision can either be specified as a single value or by two integers that define a ratio. The default is 1000 and is updated automatically by the precision of the input layout loaded into the Fracture GUI.
	Layout Magnify	Specifies that the input layout database is to be magnified by the given value as it is being read into the Calibre application. When specified, all coordinate data, including placement base points and array pitches, is multiplied by the given value, regardless of its hierarchical position.
	Unit Length	Specifies the user unit for length. When a number is supplied, it specifies the user unit of length in terms of meters. When a factor is supplied, it specifies the user unit in terms of another unit, such as mil, mm, cm, or inch. The default is 1E-6, or 1 micron.
	Layout Bump2	Specifies the layer number increment for a second optional input layout.
Derived Layers		
	Add	Defines a derived layer.
	Delete	Deletes a specified layer.
	Clear	Clears all layers.
Output Parameters		

Table 4-1. Key Features of the Basic Settings Tab (cont.)

Option		Description
	Output Type	Specifies OASIS, ASCII, or GDS as output.
	DRC Maximum Results	Specifies the number of results to save to the results database. The default value is 1000. For saving fracture input, this statement must set DRC Maximum Results to ALL to ensure that the application saves all corrected geometries. This statement is not related to saving fractured data.
	DRC Summary Report	Specifies the full pathname for the summary report file. Additional optional arguments let you control how this file is saved. The default file name is drc_summary.rep, and is saved in the same location Calibre is running from.
	DRC Results Database	Specifies the full pathname to the primary results database, and its format. The optional keyword PSEUDO specifies that the results database should include the additional levels of hierarchy that the application creates.The default file name drc_results.*format*(where *format* is oas for OASIS and gds for GDSII) is and is saved in the current directory.
Flags		
	Flag Acute	Instructs the application to issue a warning when it encounters an acute angle.
	Flag Skew	Instructs the application to issue a warning when it encounters unmerged, original, polygons that are not orientable or non-simple (containing portions that overlap other parts of themselves).
	Flag Offgrid	Instructs the application to issue a warning when it encounters an off-grid vertex.
	Layout Error On Input	Specifies whether errors or warnings encountered while reading in a layout database result in fatal errors.
	Layout Input Exception Severity	Specifies the PRECISION_RULE_FILE exception type and how it impacts the reading of a layout database.

Fracture Tab

To Access: **MDP Fracture GUI** > **Fracture** button

The Fracture tab allows you to define a Calibre FRACTURE operation to convert a GDS or OASIS database to a particular fracture format such as MEBES, JEOL, Hitachi, Micronic, Nuflare, or OASIS (listed as “VBOASIS”). This tab is available when **Options > Run Mode** is set to either **Fracture and Verify** or **Fracture only**.

Note

Select the specific Calibre FRACTURE parameters by pressing the **Properties** button. This invokes a format-specific dialog box (see Figure 4-4 for an example for the JEOL format).

You can also automatically perform an Calibre MDPverify comparison between the original layout and the resulting fractured database by selecting the **Post Fracture Verify** radio button on the Fracture tab (this greys out the **Verify** tab).

Figure 4-3. Fracture Tab

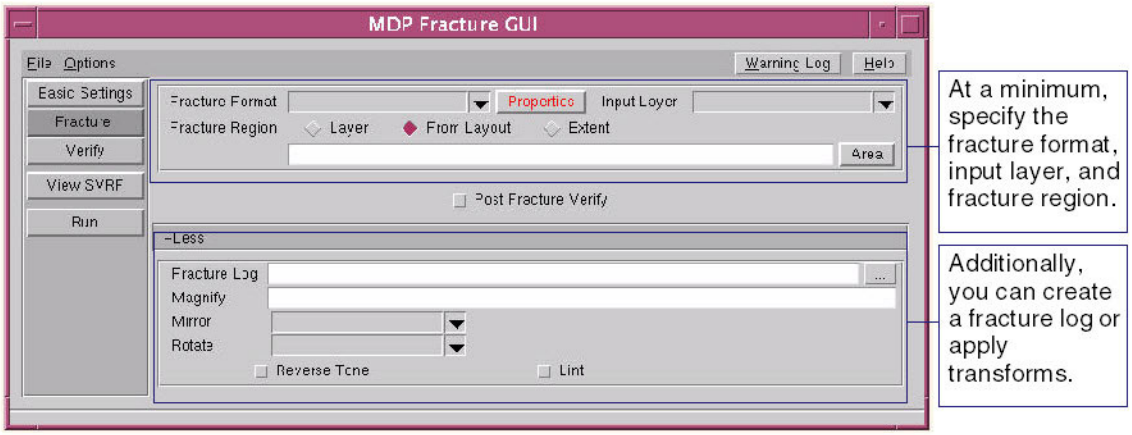
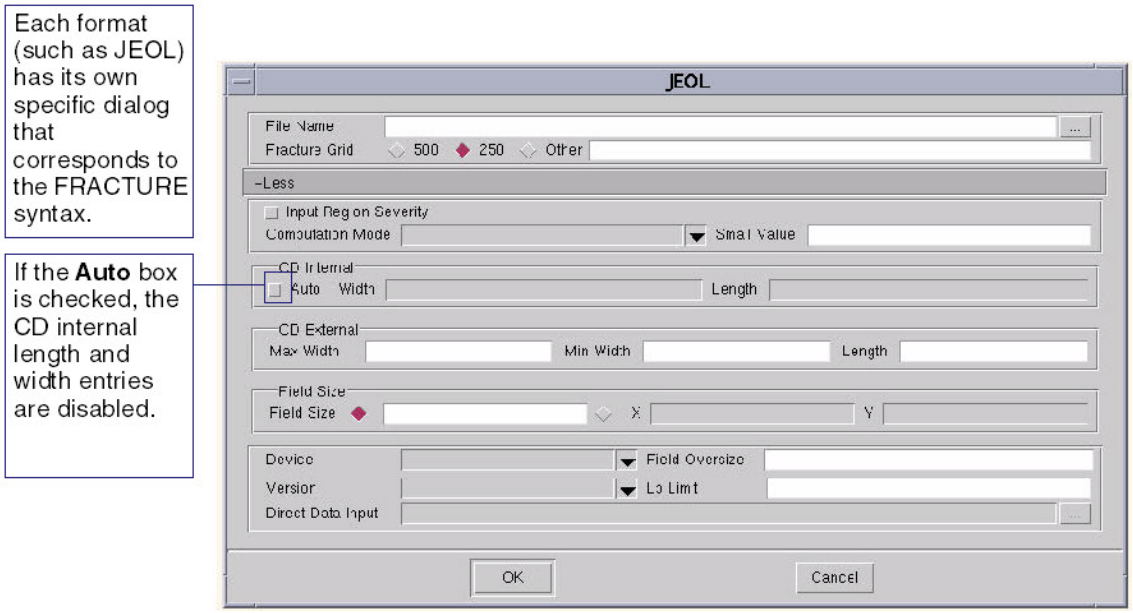


Figure 4-4. Format Property Dialog (JEOL)



Objects

Table 4-2. Key Features of the Fracture Tab

Option		Description
Required		
Fracture Format		Specifies the format (JEOL, Hitachi, MEBES, Micronic, Nuflare, OASIS), then click **Properties** to fill out the format-specific options.
Input Layer		Specifies a layer from the input database.
Fracture Region		
	Layer	Specifies a layer for fracture.
	From Layout	Specifies coordinates for fracture (or click **Area** to use the coordinates of the view displayed in Calibre MDPview).
	Extent	Only the extent is fractured.
Post Fracture Verify		Run MDP verify between the original layout and the results of the Fracture operation.
Optional		

Table 4-2. Key Features of the Fracture Tab (cont.)

Option	Description
Fracture Log	Specifies the path- and filename of the Calibre MDP summary report. The default file name is *filename_extension*.log where *filename_extension* is taken from the name and extension of the input pattern file (for example, a *TestXXXX2.jl* JEOL pattern file results in a *TestXXX2_jl.log* file).
Mirror	Defines the mirroring behavior to be applied to the layer.
Rotate	Defines the amount of rotation to be applied to the layer (0, 90, 180, 270).
Magnify	Defines the level the magnification to be applied to the layer.
Reverse Tone	Instructs the tool to perform a tone reversal of the layer.
Lint	Performs a set of checks on the input and parameters of the FRACTURE command and output the results. This lint screen warns of potential performance problems.

The specific formats are documented in the *Calibre Mask Data Preparation User's and Reference Manual* in the following sections:

FRACTURE HITACHI (FRACTUREh)

FRACTURE JEOL (FRACTUREj)

FRACTURE MEBES (FRACTUREm)

FRACTURE MICRONIC (FRACTUREc)

FRACTURE NUFLARE (FRACTUREt)

FRACTURE NUFLARE_MBF (FRACTUREn)

FRACTURE OASIS_MBW (FRACTUREi)

FRACTURE OASIS_MAPPER (FRACTUREp)

FRACTURE OASIS_MASK (FRACTUREv)

Verify Tab

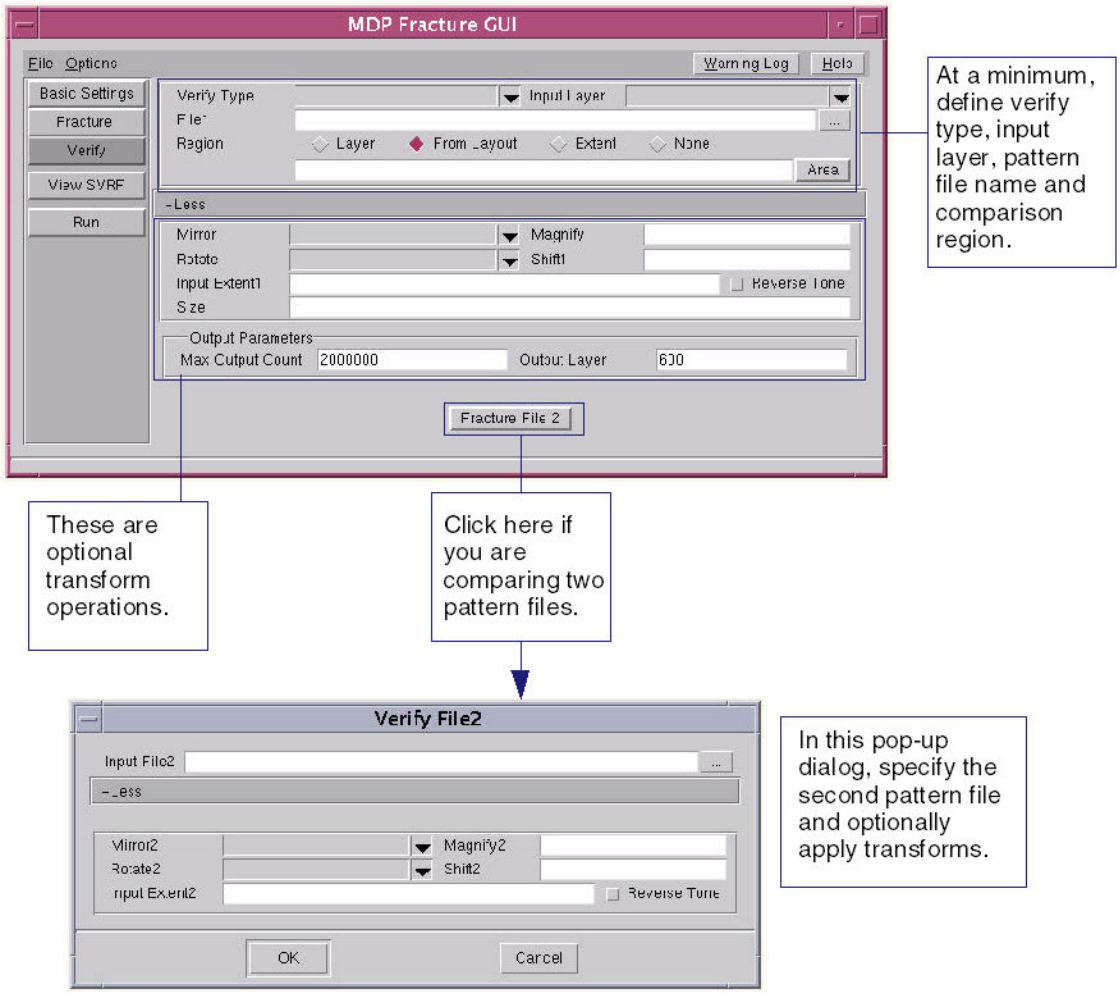
To Access: **MDP Fracture GUI** > **Verify** button

The **Verify** tab allows you to define settings for Calibre MDPverify to perform comparisons between an original database to a pattern file, or between two pattern files. This tab is available when **Options > Run Mode** is set to either **Fracture and Verify** or **Verify only**.

Note

You can also automatically perform a Calibre MDPverify comparison between the original layout and the fractured database by selecting the **Post Fracture Verify** radio button on the Fracture tab (this grays out the **Verify** tab). Figure 4-5 shows the contents of the **Verify** tab.

Figure 4-5. Verify Tab



Note

The options on the **Verify** tab correspond to keywords in the "Calibre MDPverify" chapter of the *Calibre Mask Data Preparation User's and Reference Manual*.

Objects

Table 4-3. Key Features of the Verify Tab

Option	Description
Required	
Verify Type	Specifies the type of comparison: pattern file to pattern file, or pattern file to database, which can be one of the following: HITACHI2DB HITACHI2HITACHI JEOL2DB JEOL2HITACHI JEOL2JEOL JEOL2VSB MEBES2DB MEBES2HITACHI MEBES2JEOL MEBES2MEBES MEBES2MICRONIC MEBES2VSB MICRONIC2DB MICRONIC2MICRONIC VBOASIS2VSB VBOASIS2JEOL VBOASIS2MEBES VBOASIS2MICRONIC VBOASIS2VBOASIS VSB2DB VSB2HITACHI VSB2VSB VSBJOB2VSBJOB
Input Layer	Specifies the input layer.
File 1	Specifies the name of the pattern file to be compared. If you are comparing two pattern files, you must specify the second name by clicking **Fracture File 2**.
Region	

Table 4-3. Key Features of the Verify Tab (cont.)

Option		Description
	Layer	Specifies a layer for comparison.
	From Layout	Specifies coordinates for comparison (or click **Area** to use the coordinates of the view displayed in Calibre MDPview).
	Extent	Specifies that only the extent is compared.
	None	No region specified.
Fracture File 2		Specifies a second pattern file name (and optionally transforms).
Optional		
Mirror		Applies an inverse mirror to the file before performing the comparison. Mirroring is across the axis running through the center of the region.
Rotate		Apply an inverse rotation to the file(s) before performing the comparison. Allowed values are 90, 180, and 270.
Magnify		Apply an inverse magnification to the file(s) before performing the comparison
Shift		Specifies the lower-left coordinates of the desired data extent in the Calibre coordinate system (the coordinate values to shift). The shift values used internally by MDPverify are derived based on this point. So for each file, you must determine the desired data extent for that file in the HDB coordinate system after all transform inversion is completed and specify its lower left corner.
Input Extent		Specify an alternative region that overrides the extent for an input data source. Typically, the input data sources provide an explicit definition of the extent of the corresponding layout. For instance, each fracture format contains this information in the file header. However, files of some formats may not contain such information, in which case the default layout extent is the extent of geometry in the file.
Reverse Tone		Instruct the tool to perform a tone reversal of the layer.

Table 4-3. Key Features of the Verify Tab (cont.)

Option		Description
Size		Apply a “size underover” operation to the output polygons as a final step in the processing. The size_value must be a non-negative value, expressed in user units in the Calibre coordinate system.
Output Parameters		
	Max Output Count	An optional keyword/value pair defining the maximum number of output polygons to be generated. This is useful for early detection of errors that cause a large number of outputs to be generated, such as misalignment. Specify “all” for reporting errors over 2000000.
	Output Layer	Specifies the output layer number. The default is 600.

View SVRF

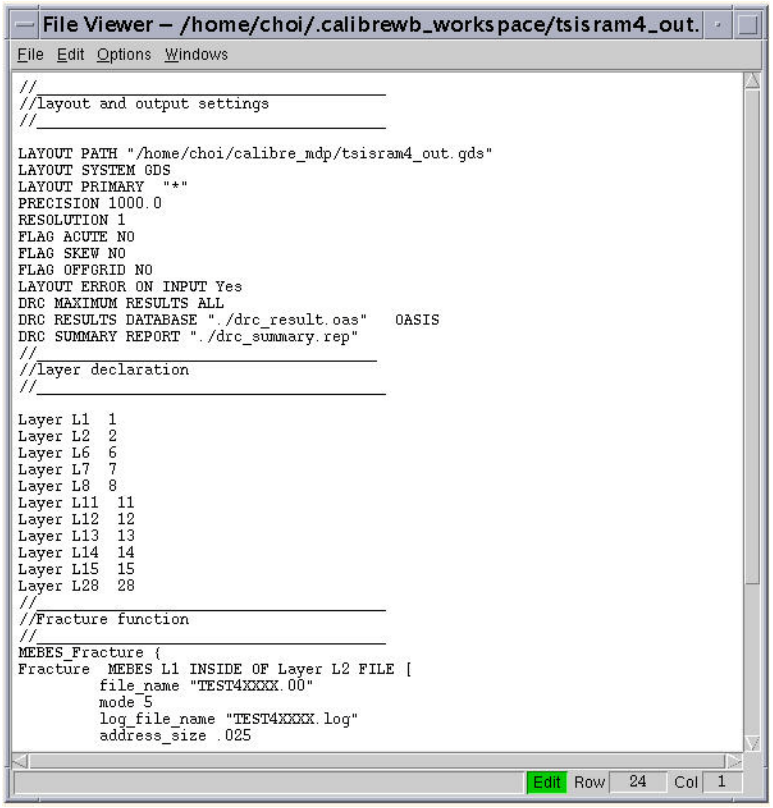
To Access: **MDP Fracture GUI** > **View SVRF** button

If you have completed filling in the information for Basic Settings, Fracture and/or Verify, click the **View SVRF** button to see the SVRF code in a text viewer. Clicking **View SVRF** creates an SVRF file with the same name as the layout's name appended with *.svrf*. The file is saved in the working directory. The file is saved by default, in the directory from which Calibre MDPview was invoked (unless otherwise specified).

Note

Modifications to the output SVRF file are not reflected to the next run since the new execution regenerates the SVRF file from the data available in the GUI. To run the updated file, you must use the Calibre batch command.

Figure 4-6. View SVRF



Objects

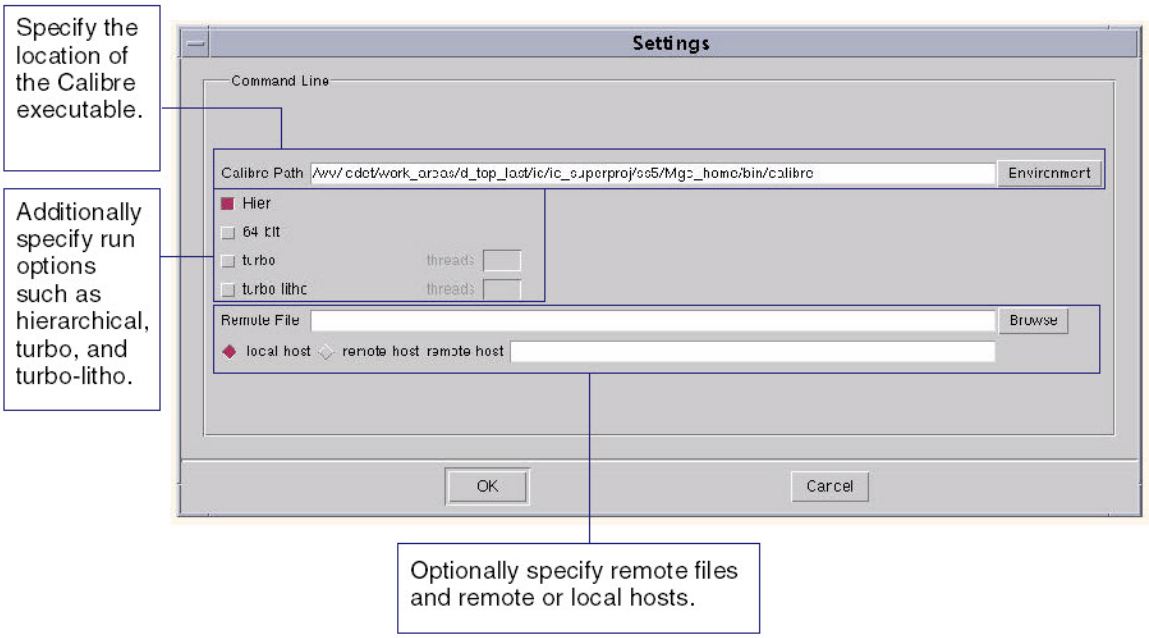
- Main Display Area: The SVRF code is shown in the main display area of this window.

MDP Fracture GUI Menu Bar

To Access: Select **Tools > MDP Fracture GUI**

The menu bar contains several operations for FRACTURE and MDPverify runs.

Figure 4-7. Run Settings



Objects

Table 4-4. MDP Fracture GUI Menus

Menu	Description
File Menu	
New	Creates a new project.
Save Session	Saves the current project.
Load Session	Loads a saved project.
Save Fracture Template	Saves fracture settings to a template file (.ft).
Load Fracture Template	Loads a previously-saved fracture template file (.ft).
Exit	Quits the Fracture GUI.
Options Menu	
Run Mode	

Table 4-4. MDP Fracture GUI Menus (cont.)

Menu		Description
	Fracture and Verify	Executes both Calibre FRACTURE and Calibre MDPverify functions on the same run. The fracture function data must be completed to permit the run, while the verify function can be deactivated by choosing Null to the Verify Type.
	Fracture Only	Executes Calibre FRACTURE only.
	Verify Only	Executes Calibre MDPverify only.
Run Settings		Specifies Calibre execution settings (see Figure 4-7). The options in this dialog correspond to command-line execution options described in the chapter "Calibre FRACTURE Operations" of the *Calibre Mask Data Preparation User's and Reference Manual*.
Working Directory		Specifies the current working directory for the Fracture GUI. You can select the directory where the output SVRF file and transcript of files are to be saved after the run. The output SVRF file receives the name of the input layout with a *.svrf* extension. The transcript also receives the same name, but with the .log extension. The default location is *.calibrewb_workspace*.
Enable Defaults		Choose to use all the fracture format parameters with their default values, or just use the required fracture parameters.

Chapter 5
Assemble Multiple Layout Databases into a Single Database

The scripting capabilities provided by the Calibre MDPview data viewer allow you to write an external script that manipulates one or more layouts databases. You write these scripts using the Tcl programming language, referencing the Viewer layout object.

You run the scripts through the Calibre MDPview data viewer, using the **calibremdpv** *script_name* invocation option described in "Batch Mode" on page 20.

Note

Writing scripts for use with the Calibre MDPview data viewer requires Tcl/Tk knowledge.

The use of this scripting capability is fully documented in the *Calibre DESIGNrev Reference Manual*.

Layout Assembly

Layout assembly takes existing cells or layout files and combines them. Assembly is used in a variety of situations, such as combining multiple layouts into a single hierarchical database, similar to a MEBES job deck.

This example creates a new GDS layout file containing four instances of an existing GDS layout file.

```
## assembly scheme for "jobdeck-processing"

## Define the input files
set L1 [layout create  "./jobdeck_example.gds"]

## Create an empty layout - which holds the assembly
set L2 [layout create]

## Cell generation and placement

## create a new level of hierarchy - a new topcell
$L2 create cell SAMPLE_MERGED

## Load the layout(s) to be placed into the assembly layout
$L2 import layout $L1 FALSE overwrite

## Place references to SAMPLE (the topcell from the original
## layout) in the SAMPLE_MERGED (the topcell for the assembled
## layout)
$L2 create ref SAMPLE_MERGED SAMPLE 0 0 0 0 1
$L2 create ref SAMPLE_MERGED SAMPLE 600000 0 0 0 1
$L2 create ref SAMPLE_MERGED SAMPLE 0 600000 0 0 1
$L2 create ref SAMPLE_MERGED SAMPLE 600000 600000 0 0 1

## Create labels/etc. on the mask level
## First create a layer for the labels
$L2 create layer 0

## Then create the text objects (specify origin, height/width
## of the box, character width, content, and orientation)
StringFeature test_text 0 -570 80 15 10 "TESTMASK" "center"
StringFeature chip1 -525 0 60 15 10 "CHIP 1" "center"
StringFeature chip2 75 0 60 15 10 "CHIP 2" "center"
StringFeature chip3 -525 605 60 15 10 "CHIP 3" "center"
StringFeature chip4 75 605 60 15 10 "CHIP 4" "center"

## Insert the text objects into the layout
# (specify layout, cell, grid to snap, layer number)
test_text addToLayout $L2 "SAMPLE_MERGED" 1 0
chip1 addToLayout $L2 "SAMPLE_MERGED" 1 0
chip2 addToLayout $L2 "SAMPLE_MERGED" 1 0
chip3 addToLayout $L2 "SAMPLE_MERGED" 1 0
chip4 addToLayout $L2 "SAMPLE_MERGED" 1 0

## File output
$L2 gdsout   ./jobdeck_example_merged.gds

## File reload
set L3 [layout create jobdeck_example_merged.gds]
```

Chapter 6
Data Operations

Because pattern files are typically quite large, the Calibre MDPview data viewer does not load the data into memory when it displays it for viewing. Instead, it leaves the data on disk, sampling it as necessary to change views. Database data, on the other hand, is loaded directly into memory. As a result, pattern file data is not editable or selectable, in the ways that database data is.

Pattern File Data Inspection

There are several different methods supported by Calibre MDPview to inspect pattern file data.

The following is a list of methods available:

- Inspect individual features in a pattern file or review a job deck by zooming in and measuring features with the ruler tool (rulers are described in detail in the *Calibre DESIGNrev Layout Viewer User's Manual*).
- Visually compare two or more layouts
  - Overlay a pattern file over an original database or overlay Calibre MDPverify results on top of a pattern file. This produces a view-only layer that combines the pattern and database layers for visual inspection. The MDP-specific overlay process is summarized in "Overlaying Results Using the Overlay Tool" on page 85 and the general overlay features are described in the *Calibre DESIGNrev Layout Viewer User's Manual*.
  - Merge two layouts after they have both been loaded into the application. This creates a new layout layer that combines the pattern file and database file. See "Extracting Transformation Data" on page 87 followed by "Aligning Data Manually" on page 89.

  In either case, to compare fractured data to the original layout database, you must compensate for any transformation that occurred during fracturing so the two align properly.
- Generate statistical reports using batch commands. These can include:
  - "MDP Summary Reports" on page 94 (produce summary reports of pattern file statistics or index file information)
  - "Statistical Analysis" on page 101 (a report of figure counts and shot counts)

Align Layouts in Calibre MDPview

A typical method of inspecting the results of a Calibre FRACTURE or Calibre MDPverify operation is to visually compare the output results with the original layout, using one of the following methods:

- Overlay a pattern file over an original database or overlay Calibre MDPverify results on top of a pattern file. See “Overlaying Results Using the Overlay Tool” on page 85.
- Merge two layouts after they have both been loaded into the application. This creates a new layout layer that combines the pattern file and database file. Note that you cannot save merged layouts created by overlaying a pattern file on a database file. These are for visual inspection only. See “Extracting Transformation Data” on page 87 followed by “Aligning Data Manually” on page 89.

When viewed using Calibre MDPview, initial results of a Calibre FRACTURE or Calibre MDPverify operation are often vastly different than the original database when overlaid. This is due to the transforms applied during the fracture and viewer operations, such as origin shifting, scaling, and other activities.

Overlaying Results Using the Overlay Tool

The primary method of visually comparing results is through the Overlay tool. Use this tool to overlay a pattern file over an original database or overlay Calibre MDPverify results on top of a pattern file.

Prerequisites

- The fracture operation performed on a layout database
- Fracture log file

Procedure

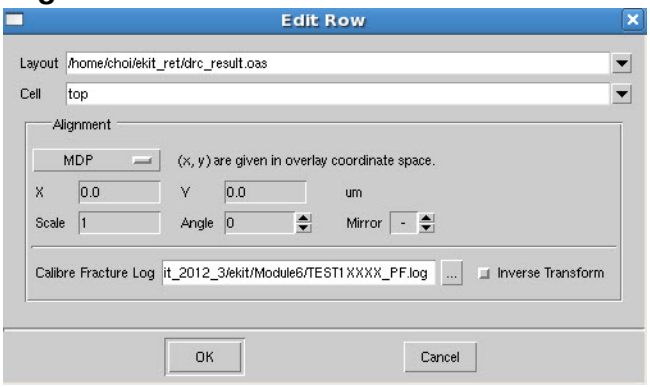
1. Load layouts and construct an overlay with no transformation adjustments.
   - From the command line, while invoking a layout viewer such as Calibre DESIGNrev or Calibre WORKbench:

     ```
     calibremdpv -m <file1> -m <file2> ... -m <fileN> -overlay
     ```

- From the layout viewer GUI, select **File > Open Layout Files**... (or Ctrl-o) and load your layout, pattern files, or Calibre MDPverify results, using the **Open Overlay** option from the drop-down menu. The files are automatically loaded into Calibre MDPview as an overlay.

2. Right click the overlay in the Cells Browser, and then select **Edit Overlay...**. The Edit Overlay dialog box is displayed.
3. Use the Edit Overlay dialog box's **Add Files** and **Add Loaded Layouts** buttons to add layout databases to the overlay. The **Add Files** button can be used to add the same layout(s) twice, and the **Add Loaded Layouts** button only adds in-memory layouts not already in the list.
4. Select a layout from the Edit Overlay dialog box's ID column and click the **Remove** button to remove a layout file from the overlay.
5. Select layout fields in the Edit Overlay dialog box's layout table to change the topcell name and also make layout translations. This can also be accomplished clicking the **Edit...** button. This invokes the Edit Row dialog box.
   a. In the Edit Row dialog box, select a row with a fracture output layer that you want to align.
   b. Select **MDP** from the drop-down menu in the **Alignment** section.
   c. Compensate for any transformation that occurred during fracturing by specifying the path to your fracture log in the Calibre Fracture Log text entry field. Calibre MDPview will take the MDP transform information from the log file and use it to align the selected overlays.
   d. If you wish to apply an inverse transformation (for example, if you want to match the pattern file to the original layout), select **Inverse Transform**.

Figure 6-1. Edit Row With MDP Transform



   e. Click **OK** to apply the transforms.

6. In the Edit Overlay dialog box, click **Apply** to save the changes.
7. Click **Close** when you are done with all changes.

Results

The overlay layers are displayed in Calibre MDPview and are ready for visual comparison.

The Overlay tool is described further detail in the *Calibre DESIGNrev Layout Viewer User's Manual*).

Extracting Transformation Data

To compare fractured data to the original layout database, you must compensate for any transformation that occurred during fracturing so the two align properly.

Note

When you merge or overlay two layouts, where one of them is a Hitachi pattern file, you must manually realign the data so that they overlay properly. Refer to the section "Aligning Data Manually" on page 89 for detailed information on how to do this. This occurs because Hitachi data files allow you to specify any location as the origin.

Do this by extracting the transform from the fracture log file and applying it appropriately.

Prerequisites

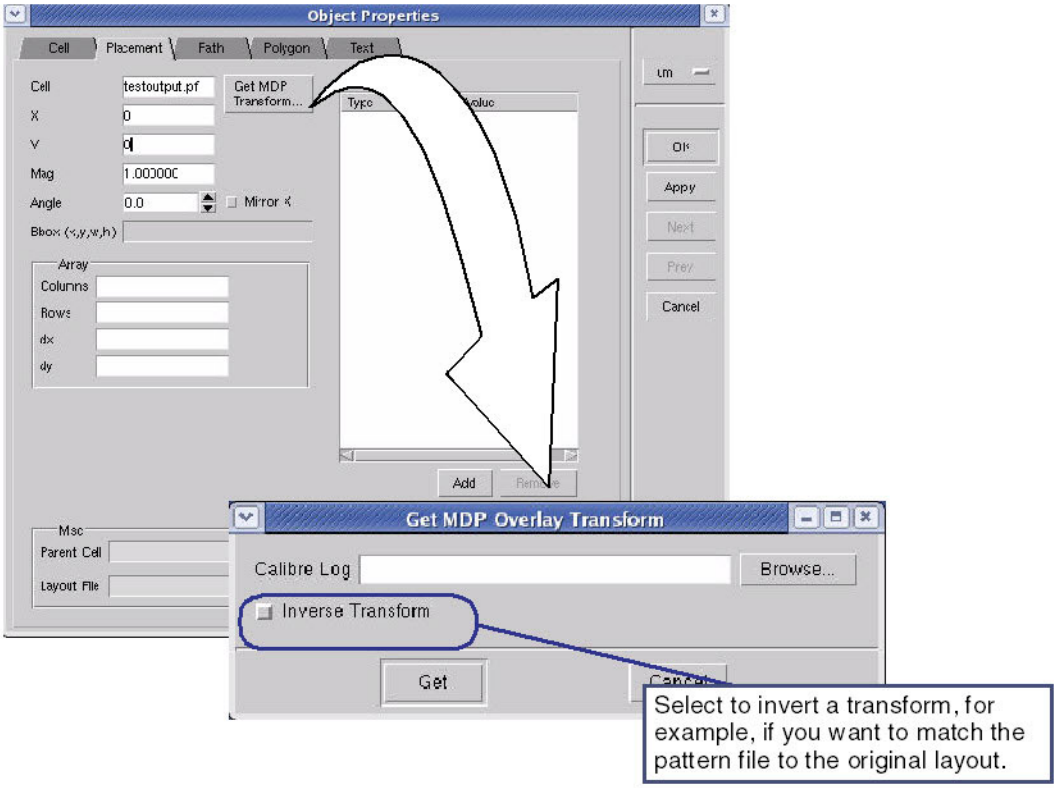
- The fracture operation performed on a layout database
- Fracture log file
- The layout loaded into Calibre MDPview

Procedure

1. Make sure the top level cell is displayed
2. With the viewer depth set to 2 or higher so you can see the actual data, select the cell containing the data to be transformed.
3. Open the Object Properties dialog box: **Object > Properties** > **Placement** tab.[1]

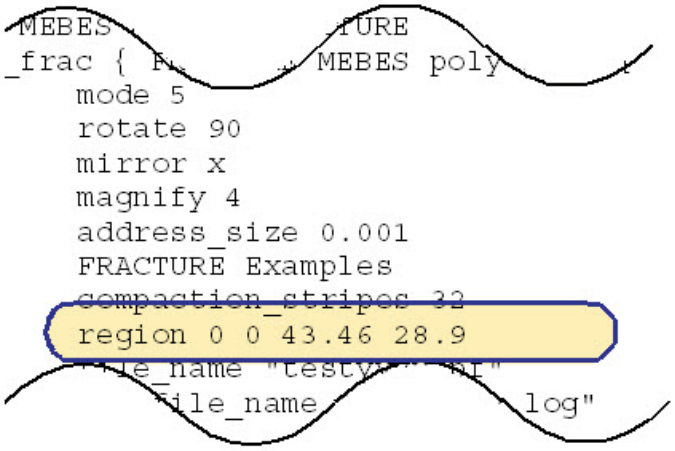
1. If the layout was shifted during the merge, you must shift the layout back into its original position prior to proceeding to the next step.

4. Click **Get MDP Transform** to open the Get MDP Overlay Transform dialog box.



5. Browse to select the fracture log file name, or copy the pathname directly from the Fracture operation in the rule file.
6. Select the correct type of transform:
   - To transform the layout database to match the pattern file generated from it, make sure Inverse Transform is not selected.
   - To transform the pattern file to match the layout database, select**Inverse Transform**.
7. Click **Get** to load the transform into the Object Properties dialog box.
8. Click **Apply** to transform the data.
9. If you have transformed the mask layer data before using it as input to the Fracture command, you must make further adjustments, as described in “Aligning Data Manually” on page 89.

Figure 6-2. Browsing the Fracture Operation



Results

With the transformation data applied to the mask layer data, you can now compare the fracture results to the original layout.

Aligning Data Manually

The fracture log file only records the transformation applied by the fracture operation. If you have transformed the mask layer data before using it as input to the Calibre FRACTURE command, you must make further adjustments.

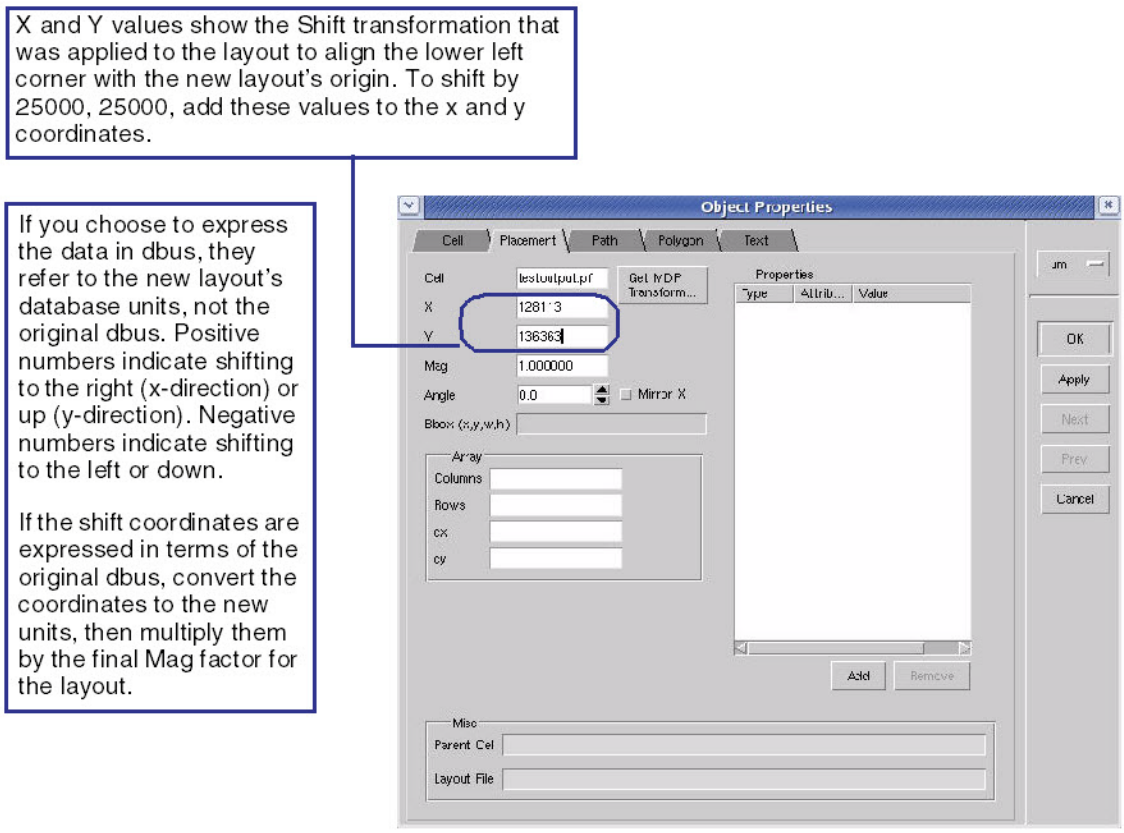
Prerequisites

- The fracture operation performed on a layout database
- The layout loaded into Calibre MDPview

Procedure

1. For alignment to work properly, make sure the top-level cell is displayed.
2. With the viewer depth set to 2 or higher so you can see the actual data, check how well the layouts are aligned to begin with.
3. If the two layouts are not aligned exactly, set the viewer depth to 0 so only the cell outlines are visible, then select the outline of the layout you want to adjust.
4. Display the Properties dialog box by choosing **Object > Properties**.

Figure 6-3. Modifying the Transformations



5. Align the data by editing the values in the shift (x and y) and Mag fields. Keep in mind that if you are manually aligning data after applying transforms extracted from the fracture log file, the final values in these fields must reflect the dual transformation. If you have calculated the additional shifting needed expressed in original layout database units, calculating the dual transformation involves three steps:

   a. Convert user units of the original database to the database units of the merged database.

   b. Multiply the shift transformation values by any other magnification factor, as required.

   c. Add these values to the shift values.

Note

FRACTURE HITACHI has the default location for the origin (0,0) in the center of the mask. In overlaying to other file formats, the HITACHI data is offset by one-half the window size. To align to other file formats, you can use the log file generated during the fracture run, or put in an offset equivalent to half the window size.

Results

The transform data allows you to align the mask layer to the layout database and examine the fracture results.

Merge and Transform

Assume layouts A and B where dbuA = 2 nm and dbuB = 1.25 nm. To find a DBU usable in the transform operation, find the least common multiplier of the dbus, as follows:

```
LCM(1/2,1/1.25) = LCM(0.5,0.8) = 4
1/4 = 0.25
```

The DBU for the merged layout is 0.25 nm.

If the application cannot calculate a new DBU for the merged layout using the previous method (because of overflow, for instance), it uses the lesser of the two input dbus. In the previous example, if the application failed to calculate a DBU for the merged layout using the LCM method, it would have used a DBU of 1.25.

The layout viewer then applies the necessary magnification automatically, as part of the merge process. Layout A is magnified by a factor of 8 (2 /0.25) and layout B is magnified by a factor of 5 (1.25 /0.25).

Now assume that during this merge operation, the Calibre MDPview data viewer shifted layout B by (-22000, -97000) to align its lower left corner with the lower left corner of layout A,

which is at (0,0). Looking at the original data, layout B must be shifted by (500, 500) dbuB to align it properly. You also see that layout B is still half the size it needs to be:

Figure 6-4. Adjusted Values for Merged Example

dbu_B = 1.25 nm
dbu_{merged} = 0.25 nm
1 dbu_B = 5 dbu_{merged}

Convert to dbu_{merged}:

(500, 500) dbu_B = (2500, 2500) dbu_{merged}

Calculate final Mag value:

merge Mag = 5
additional Mag = 2
final Mag = (5 * 2) = 10

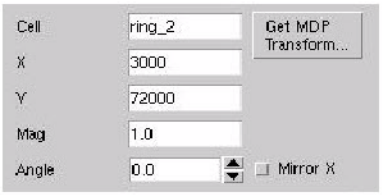
Multiply shift values by final Mag value:

10*(2500, 2500) dbu_{merged} = (25000, 25000) dbu_{merged}

Add to the shift values used during the merge:

(-22000,-97000) + (25000, 25000) = (3000,72000)

Enter these values in the x,dbu and y,dbu fields in the Object Properties dialog box.

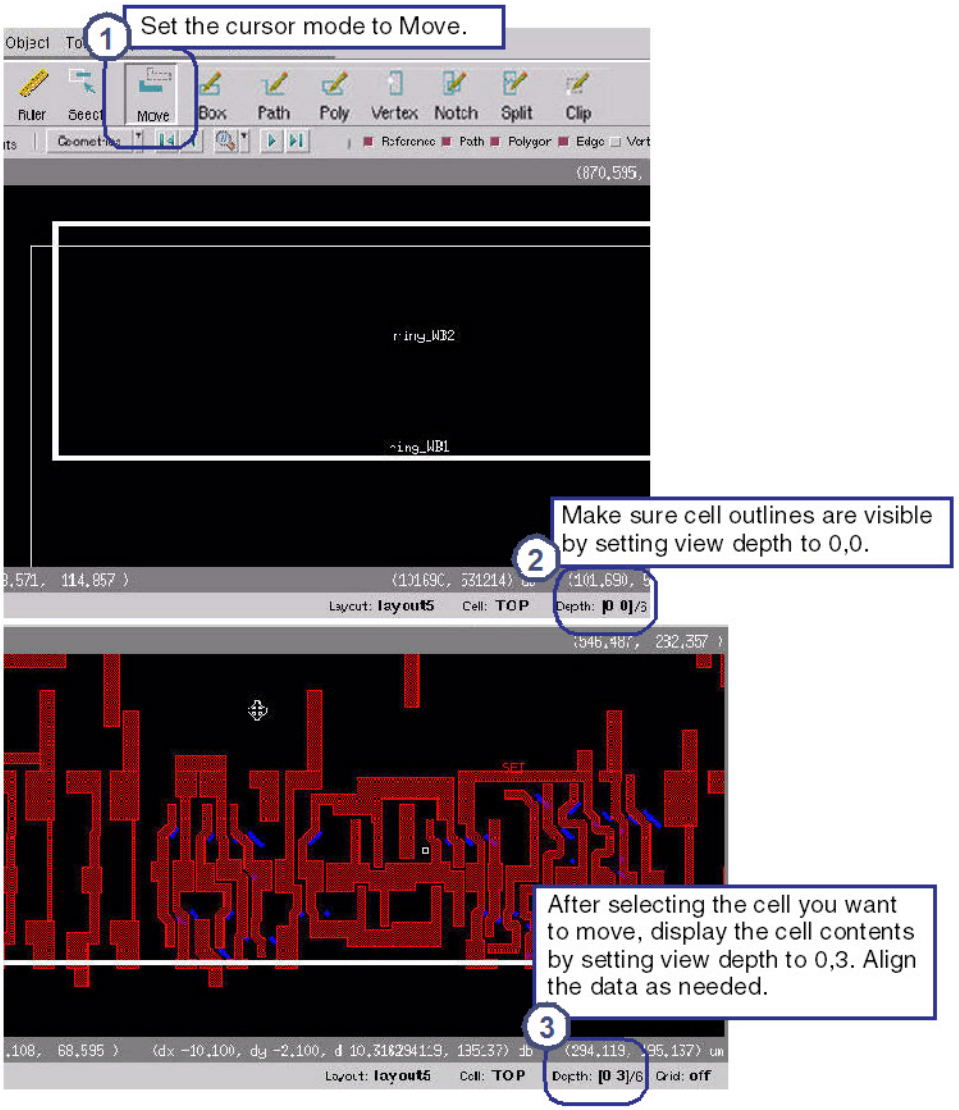


A Shortcut for Aligning Data

If the two layouts area magnified, mirrored, and rotated properly, but still not aligned right, you can use the Move function to reposition the data. The challenge here is that to select a “cell” in a merged layout[2], you must have only the cell outline visible, but to align it properly, you must see the contents. You can get around this by changing the view depth after putting the cursor in Move mode and selecting the cell.

2. The original layout is represented as a cell in the new layout.

Figure 6-5. Aligning Data Shortcut



MDP Summary Reports

A set of Calibre MDPview batch commands that provide a summary reports such as pattern file statistics and index file information.

Usage

mebesIndexInfo ***fractured_file_name***

mebesinfo ***fractured_file_name*** [-header]

jeolinfo ***fractured_file_name***

jeolIndexInfo ***fractured_file_name***

vsbinfo ***fractured_dir_name*** [-detail] [-checksum] [-checksum_per_frame]

hitachiinfo ***fractured_file_name***

micronicinfo ***fractured_dir_name*** [-detail] [-hier]

oasisIndexInfo ***oasis_file_path*** [-v]

oasisinfo ***oasis_file_path*** [-layoutInfo *options*]

layoutType ***type_path***

Arguments

- ***fractured_file_name***

  A required parameter specifying the name of the fracture file.

- ***fractured_dir_name***

  A required parameter specifying the directory containing fracture information.

- ***index_file_name***

  A required parameter specifying the name of the index file.

- ***oasis_file_path***

  A required parameter specifying the OASIS file path.

- -header

  An optional parameter that allows you to limit the mebesinfo output to the MEBES header record only.

- -detail

  An optional parameter that produces a total trapezoid count VSB12 format files.

  Note

  Usage of the **-detail** option requires further processing of VSB files and takes longer to complete. This option does not produce information for VSB11 files.

- -checksum

  An optional parameter that adds a **Checksum:** line to the output report. In addition, this same number is generated during VSB12 fracturing and is output in the run transcript file. The -checksum output can be compared to the checksum reported by the Calibre FRACTURE operation in order to detect storage system errors.

- -checksum_per_frame

  An optional parameter that generates the checksum for each frame. It generates information in the following format:

  Checksum for frame <framenum>: CELL: <cellchecksum> LINK <linkchecksum> REF: <refchecksum>

- -hier

  An optional parameter that produces a hierarchical count of the pattern file statistics.

- -v

  An optional parameter that outputs the size of the file in the transcript and includes the Remote size and Flat Vertex count.

- -layoutinfo *options*

  An optional parameter that specifies the following options for oasisinfo:

  -bbox *cellname*... — An optional keyword that prints information about the bbox (bounding box) of the specified cells in the following format: bbox {*cellname*{*lx ly width height*}}...

  For example:

  ```
  bbox {top {0 0 1000 1000}} {cell1 {-10 -10 100 100}}
  ```

  -cellcount — An optional keyword that prints the cell count in the following format: cellcount *value*

  For example:

  ```
  cellcount 12589
  ```

  -cells — An optional keyword that prints the names of all cells in the following format: cells {*cell1 cell2*...}

  For example:

  ```
  cells {cell1 cell2 and1x2}
  ```

  -layers — An optional keyword that prints the layer number, datatype, and layer name in the following format: layers {{*layernumber datatype layername*}...}

  For example:

  ```
  layers {{1 0 layer1} {2 0 layer2} {3 33 m1} {3 99 m1fill}}
  ```

  If a layer name is not present, a layer number is specified as the layer name.

-precision — An optional keyword that prints the precision information in following format: precision *value*

For example:

```
precision 1000.0
```

-topcell — An optional keyword that prints the name of the topcell in the following format: topcell *value*

For example:

```
topcell top_chip
```

-units — An optional keyword that prints the unit information in the following format: units *value*

For example:

```
units 0.001
```

More than one option can be specified. Outputs of all options are written out as a white space-separated string. When the outputs contain more than one value (for instance, -cells and -layers), the output string is modeled closely on lists in TCL.

The command generates an error if an index file is not present.

The output of this -layoutInfo only reflects the cells used in indexing and not on the input OASIS file. This is relevant when multiple top cells are present in the input OASIS file, as only one of the top cells will be selected and indexing only considers cells referred in that cell.

The following is an example invocation of oasisinfo with the -layoutInfo option:

```
oasisinfo unit.oas -layoutInfo -precision -units -topcell -cellcount
   -cells -layers -bbox L2_1 TOPCELL L1_1 junk
```

For this invocation, the following output is generated:

```
units 0.001 precision 1000.0 topcell TOPCELL cellcount 4 cells
{ TOPCELL L1_1 L2_1 L1_2 } layers {{0 0 new0} {1 0 new1}}
bbox {L2_1 {-272 92 5896 4680}} {TOPCELL {-3336 -2036 12456 12868}}
{L1_1 {584 3632 5896 4680}}
```

If the -layoutInfo option is not specified, oasisinfo generates information about the cells and count objects present in the input layout that reflects information generated during the read of the data.

- ***type_path***

  A required argument that is interpreted as a Linux file path. The layoutType command attempts to identify the format type of the file or directory. A symbolic string is returned by the command indicating the determined type. This includes:

  - UNKNOWN: The type could not be determined.

- GDS: GDS file.
- OASIS: OASIS file.
- MEBES: MEBES file.
- MEBES_JOB: MEBES job deck.
- JEOL: JEOL file.
- JEOL_JOB: JEOL job deck.
- VSB: NUFLARE chip directory.
- VSB_JOB: NUFLARE job deck.
- HITACHI: Hitachi file.
- MALY_JOB: MALY job deck.

Description

These commands take the path name for the pattern file or directory as an argument.

- If the Calibre MDPview data viewer is running:

  Type the batch commands at the Tcl prompt in the shell window from which you invoked the application.

  ```
  mebesinfo trxxxxx.pf
  ```

- If the Calibre MDPview data viewer is not running:

  Type the command at the Linux prompt, as shown in the following example

  ```
  calibremdpv -a mebesinfo trxxxxx.pf
  ```

These commands write the pattern file statistics to**stdout**, which in most cases is the terminal window. You can redirect the data using any of the standard Linux method. If you are using these commands at the command line, while the application is running or in a Tcl file, you can store the information in a variable, then write it to a text file.

```
:% set a [mebesinfo sample.pf]
:
: <output>
: %echo $a MEBES_data.txt
```

The number of shapes reported by the information commands may be simple counts or reported in terms of U and L, where total count = U * 2^32 + L. For example, in the following example jeolinfo report:

```
total number of rectangles (L, compacted): 459076
total number of trapezoids (L, compacted): 350
total number of rectangles (U, compacted): 1
total number of trapezoids (U, compacted): 0
total number of rectangles (L, decompacted): 2050066
total number of trapezoids (L, decompacted): 512
total number of rectangles (U, decompacted): 3
total number of trapezoids (U, decompacted): 0
```

- The total number of compacted rectangles in the pattern file is 1*2**32 + 459076 = 4295426372.
- The total number of compacted trapezoids in the pattern file is 0*2**32 + 350 = 350.
- The total number of decompacted rectangles in the pattern file is 3*2**32 + 459076 = 8590393668.
- The total number of decompacted trapezoids in the pattern file is 0*2**32 + 512 = 512.

The mebesIndexInfo and jeolIndexInfo commands provide information including the path of the associated index file, its version number, and availability of viewer components. The utility searches for the index file in the location specified by the Linux environment variable MDPIndexSearchPath. When the variable is undefined, the location defaults to the location of ***fractured_file_name***. The mebesIndexInfo utility is now aligned with a similar utility for OASIS, oasisIndexInfo.

The oasisIndexInfo command checks the validity of an index for an OASIS file. The command argument is the OASIS file, not the index file. This command states if an index file exists, the version of any such file, and the version used by the current executable. Finally, the command issues a statement if the index is not valid according to the previously-listed criteria. The following are examples of using oasisIndex (to generate the OASIS index) and oasisIndexInfo, both the command line invocation and the output:

- Create an OASIS index with Calibre 2017.4, OASIS index version 3.0:

```
calibremdpv -a oasisIndex file.oas | tee oasisIndex.file.log
```

```
Creating an index, version: 3.0.
SEG TIME: CPU TIME = 0  REAL TIME = 0
MEMORY: LVHEAP = 3/3/4.
Sequential parsing of OASIS file.
PARSE TIME: CPU TIME = 0  REAL TIME = 0
MEMORY: LVHEAP = 3/59/59.
NOTE: Cell GUNTER was automatically selected as the top cell.
NOTE: REMOVED 89 cells, 143 single, 49 array placements.
Note: Viewer components are present.
POST PROCESSING TIME: CPU TIME = 0  REAL TIME = 0
MEMORY: LVHEAP = 3/59/59.
Opening an index file, file.oas.fvi.
INDEX OUTPUT TIME: CPU TIME = 0  REAL TIME = 0
MEMORY: LVHEAP = 3/59/59.
```

- Report OASIS index version:

```
calibremdpv -a oasisIndexInfo file.oas | tee oasisIndexInfo.file.log

Opening an index file, file.oas.fvi.

Note: Existing index file version: 3.0.  Current version: 3.0.
Note: Viewer components are present.
```

- Report OASIS index version and more verbose information:

```
calibremdpv -a oasisIndexInfo file.oas -v | tee \
       oasisIndexInfo-v.file.log

Opening an index file, file.oas.fvi.

Note: Existing index file version: 3.0.  Current version: 3.0.
Note: Viewer components are present.
```

MEBES Header Adjustment

A set of Calibre MDPview batch commands that allows you to adjust the MEBES header in a report.

Usage

mebesHeaderAdjust ***mebes_disk_file_name*** **-name** ***new_file_name***

Arguments

- ***mebes_disk_file_name***

  Specifies the existing MEBES disk file name.

- ***new_file_name***

  Specifies a new internal MEBES pattern file name. The new name must still meet the MEBES file name requirements. This command allows you to input pattern files up to 20 characters for internal renaming. However, if you violate the MEBES file naming convention, an error is generated and the process stops.

Description

One statistic that is output from a MEBES report is the header name. You may opt to rename the internal name of the pattern file to a new name that complies with the MEBES pattern file naming convention (12 characters inclusive with a period separating the last two). This can be accomplished using the mebesHeaderAdjust command. This command changes the pattern file name field in the MEBES header record to a new user specified pattern file name.

Examples

```
calibremdpv -a mebesHeaderAdjust testxxxxx_10s3.pf -name test10s3x.pf
                    //You can change the internal name of a file.
```

If you rename your pattern files (such as with the MEBES FRACTURE disk_file_name option), you can use mebesHeaderAdjust to restore the names back to MEBES-compliant names. Typically, if these files don't have the same internal/external name, the photolithography exposure tool settings causes it either to issue a warning or an error.

Statistical Analysis

A set of Calibre MDPview batch commands that generates a report of figure counts and shot counts.

Usage

jeolTrapStats ***input_file small_figure_size*** [*shot_size*]

hitachiTrapStats ***input_file small_figure_size*** [*shot_size*]

micronicTrapStats ***input_file small_figure_size*** [*shot_size*]

vboasisTrapStats ***input_file*** {***topcellname*** | *} ***small_figure_size*** [*shot_size*]

vsbTrapStats ***chip_directory small_figure_size*** [*shot_size*]

Note

The mixed-capitalizing of the command name, as shown in the syntax, is required.

Arguments

- ***input_file chip_directory***

  A required literal or relative path name that specifies the location of the JEOL, Hitachi, and OASIS (OASIS.MASK, OASIS.MAPPER, OASIS.MBW) file, or VSB11, VSB12, VSB12i, or NUFLARE_MBF directory.

- ***topcellname*** | *******

  Either specify the top cell name, or an asterisk (*) for the current top cell. It is recommended that you only use the asterisk if there is only one top cell in the input file.

- ***small_figure_size***

  A required floating point number that specifies, in microns, a minimum size for figures.

- *shot_size*

  An optional floating point number, in microns. The default value corresponds to the different mask writers. The value you specify should correspond to the settings for your mask writer. The default value is 2.55.

Description

Calibre MDPview provides utilities for the statistical assessment of JEOL, Hitachi, Micronic, OASIS (OASIS.MASK, OASIS.MAPPER, OASIS.MBW), and VSB11, VSB12, VSB12i, or NUFLARE_MBF formatted data. Small figure counts and shot counts are performance metrics of mask writer data and influence the mask writing time. They also have an impact on the control of critical dimension trapezoids.

These commands are case-sensitive, therefore you must enter them as in the following example, specified using the **-a** argument to **calibremdpv**:

```
calibremdpv -a jeolTrapStats ./jeol_files/test_1.jeol 0.1
```

Note

The shot count estimation algorithm is only approximate; it will likely be close to, but not in exact agreement with, the proprietary shot decomposition algorithms of the mask writing machines.

Calibre MDPview presents the statistical information in the following format:

```
Shot counts: rects: <rectangular_trapezoids_shot_number>
traps: <non_rectangular_trapezoids_shot_number>
total: <shot_number> Small count: <small_figure_size>
```

where:

- rects — Number of shots from rectangular trapezoids generated for the specified shot_size.
- traps — Number of shots from non-rectangular trapezoids generated for the specified shot_size.
- total — Sum of rects and traps values.
- Small count — Number of rectangles or trapezoids that are smaller than the *small_figure_size* value.

The values reported for “rects”, “traps”, and “total” refer to estimated shot counts. The “Small count” refers to trapezoids, not shots. Trapezoids are generally composed of multiple shots.

A small figure is defined as a small trapezoid, where a trapezoid is “small” if any of the lengths of the parallel sides or the distance between the parallel sides is less than the *small_figure_size* parameter. A triangle is small if either of its orthogonal sides have length less than the *small_figure_size* parameter.

The shot_size optional parameter in the Calibre FRACTURE syntax attempts to set the optimum maximum size of a square shot (refer to the *Calibre Mask Data Preparation User’s and Reference Manual* for a description of Calibre FRACTURE syntax). In general, this should

match the mask writer shot_size parameter. The following is a syntax example of shot size implemented in FRACTURE (in this case, VSB12):

```
LAYER ExtentData 98
LAYER FractureExtent 99
POLYGON -770 -580 770 580 ExtentData
POLYGON -770 -580 770 580 FractureExtent

toFracture {FRACTURE NUFLARE ExtentData INSIDE OF LAYER FractureExtent
FILE [
   version 12
   chip_directory "./output_vsb12"
   computation_mode section
   vboasis_path "./input_oas/tsisram.oas"
   magnify 4
   conversion_address_unit 0.0005
   max_skew_approximation_error 0.01
]
```

The shot count is then reported when running TrapStats. The following is an example of vsbTrapStats using *shot_size*:

```
calibremdpv -a vsbTrapStats output_vsb12/ 0.001 0.5
```

In this example:

- output_vsb12 is the ***chip_directory***.
- 0.001 is the *small_figure_size*.
- 0.5 is the value for the optional *shot_size* parameter.

The output would appear much like the following:

```
Shot counts: rects: 68893426  traps: 587045 total: 69480471
Small count: 25
```

For an accurate trapezoid count, always use a TrapStats [*shot_size*] value that matches your mask writer shot size.

For the lowest shot count, always use a shot_size FRACTURE parameter value that matches your mask writer shot size, otherwise, it is set to 2.0 by default.

Data Conversion

Calibre MDPview provides utilities that allow you to convert data from one format to another. Conversion may be needed due to a number of possibilities. For example:

- Convert the fracture file to OASIS format to view the OASIS representation in Calibre MDPview (see "Fractured Data to OASIS Format Conversion" on page 105).
- Optimize the hierarchy of external tool-generated GDSII or OASIS files for the efficiency and speed of downstream processing (see "Hierarchy Optimization for Processing" on page 108).
- You can check whether an OASIS.MASK-style layout is valid (see "Format Conformance Checks" on page 109).
- Create a single OASIS file from a job deck and its input pattern files, also known as "job deck smashing" (see "Job Decks to OASIS Files Conversion (Job Deck Smashing)" on page 111). This allows the files to be read and processed by the hierarchical engine used by Calibre tools.
- Use a MEBES job deck as a "template" and converting it to another format without having to rewrite the job deck for each format. The following conversions are available:
  - "MEBES Job Deck to a VSB Job Deck Conversion" on page 116.
  - "MEBES Job Deck to a Micronic Job Deck Conversion" on page 119.
- Convert a 2D barcode (QR code Model 2) to OASIS data to encode reticle information for EUV steppers (see "Text to QR Code Conversion" on page 126).

Fractured Data to OASIS Format Conversion

A set of Calibre MDPview batch commands that convert a fractured file into OASIS format.

Usage

mebes2oasis [***fractured_file_name*** | stdin] [*output_file_name* | stdout] [-topcell *name*] [-layer *number*] [-grid FINE] [strict {0 | 1}] [cblock {0 | 1}]

jeol2oasis ***fractured_file_name*** [*output_file_name* | stdout] [-topcell *name*] [-layer *number*] [strict {0 | 1}] [cblock {0 | 1}]

vsb2oasis ***fractured_directory_name*** [*output_file_name* | stdout] [-topcell *name*] [-layer *number*] [strict {0 | 1}] [cblock {0 | 1}]

hitachi2oasis ***fractured_file_name*** [*output_file_name* | stdout] [-topcell *name*] [-layer *number*]

micronic2oasis ***fractured_file_name*** [*output_file_name* | stdout] [-topcell *name*] [-layer *number*] [-mem *threshold*] [-size *threshold*] [strict 0 | 1]

Arguments

- ***fractured_file_name*** | *fractured_directory_name* | stdin

  Specifies a relative or explicit pathname for the data to be converted, or specifies standard input (stdin).

- *output_file_name* | stdout

  Specifies a relative or explicit pathname for the file to be created, or specifies that output be piped to standard output (stdout). The converter overwrites any existing file of the same name. You can use the LAYOUT PATH option to specify standard input (stdin) as follows:

  ```
  LAYOUT PATH stdin
  ```

- -topcell *name*

  Specifies the name of the top-level cell to be used in the output file. The default *name* is TOP.

- -layer *number*

  Specifies the layer number to be used in the output file. The default *number* is 0 (zero).

  By default, the micronic2oasis conversion generates multilayer output file with OASIS layer numbers corresponding to Micronic layer numbers. If, however, -layer is specified, all geometry is written onto the given layer; note that Micronic layer operations are not performed in this case.

- -grid FINE

  Applies to mebes2oasis only: Converts the MEBES file such that the OASIS data supports a grid of 1/16 the input MEBES address units. FINE is currently the only legal value for the -grid option.

- -mem *threshold*

  Applies to micronic2oasis only: Limits the size (in bytes) of buffer memory used in the process. It is a good practice to allow the tool to use as much memory as possible, as that lessens, or completely excludes, side-effects due to limited resources, in particular, the distortion of cell hierarchy. The following suffixes can be used in *threshold*: K, KB, M, MB, G, GB (for example, -mem 1G).

- -size *threshold*

  Applies to micronic2oasis only: Limits the size (in bytes) of OASIS cells created in the process. Using this option is recommended if the input pattern file contains massive flat data, in which case limiting the output cell size improves the effectiveness of handling the OASIS file. The recommended option value is 10-20 MBytes for spatially coherent data. If data does not exhibit good spatial coherence, the converter attempts to re-arrange geometry to improve data locality in the output file; in this case, smaller threshold values (1-5 MBytes) are likely to produce better result. Similarly to "-mem", suffixes can be used in threshold value (e.g. -size 2M).

- strict {0 | 1}

  An optional argument that activates (1) or deactivates (0) generation a strict mode OASIS file. By default, all converters generate strict mode OASIS files. This mode is not currently supported for Hitachi conversion.

- cblock {0 | 1}

  An optional argument that activates (1) or deactivates (0) generation of OASIS file using CBLOCKs. By default, all converters generate OASIS files containing CBLOCKs. This option is currently not supported for Hitachi and Micronic converters.

Description

One way to check the validity of your fractured data is to convert it into OASIS format, and compare this new OASIS file with the original file.

Invocation of these commands is as follows:

- If the Calibre MDPview data viewer is running:

  Type the batch command at the Tcl prompt in the shell window from which you invoked the application.

  ```
  mebes2oasis trxxxxx.pf trxxxxx.oas
  ```

- If the Calibre MDPview data viewer is not running:

  Type the command at the Linux prompt. Use the calibremdpv invocation using the -a argument, followed by the batch command.

  ```
  calibremdpv -a mebes2oasis trxxxxx.pf trxxxxx.oas
  ```

Note that because the way different layout formats store layout database data, particularly rectangle and trapezoid data, the OASIS file size is the same as or smaller than the pattern file, because OASIS supports shape arrays and has more efficient data descriptors.

By default, for both vsb2oasis and jeol2oasis, dose levels (achieved through the FRACTURE keyword trapezoid_groups) are automatically converted to unique datatypes so they can be more easily differentiated.

Hierarchy Optimization for Processing

A set of Calibre MDPview batch commands that optimize the hierarchy of external tool-generated GDSII or OASIS files.

Usage

gds2oasis *input_file_name output_file_name* [-opt]

oasis2oasis *input_file_name output_file_name*

Arguments

- ***input_file_name***

  Specifies an input file name. For gds2oasis, the input file is a GDSII file. For oasis2oasis, the input file is an OASIS file.

- ***output_file_name***

  Specifies the name of the OASIS output file.

- -opt

  Specifies that hierarchy optimization be performed on the input GDSII file.

Description

These commands optimize the hierarchy of external tool-generated GDSII or OASIS files for the efficiency and speed of downstream processing.

Format Conformance Checks

A Calibre MDPview batch command that checks the validity of OASIS.MASK layouts

Usage

vboasisCheck *input_file* [*top*] [-extentLimit *elimit*] [-depthLimit *dLimit*]
[-P44 [-relaxedGeomOffsetCheck] [-relaxedBBOXCheck]]

Arguments

- ***input_file***

  A required argument that specifies the path to the OASIS.MASK file.

- *top*

  An optional argument that specifies the top cell name.

- -extentLimit *elimit*

  An optional argument that specifies the maximum cell size.

- -depthLimit *dlimit*

  An optional argument that specifies the hierarchy depth.

- -P44 [-relaxedGeomOffsetCheck] [-relaxedBBOXCheck]]

  An optional argument that specifies that the OASIS.MASK format is checked.

  The -P44 option checks that the bounding box value present in S_BOUNDING_BOX property of the cell is same as the bounding box of the contents of the cell and all hierarchy in the cell. If -relaxedBBOXCheck is specified, a check occurs to see if the bounding box specified in S_BOUNDING_BOX is larger than the cell bounding box.

  With the -P44 option, the P44_GEOMETRY_OFFSET value is checked to be 0 for cells that do not have geometries (including the top cell). A violation of this value is reported as a warning by the checker. However, the -relaxedGeomOffsetCheck option allows you to specify the violation be accepted without issuing a warning.

Description

By default, the OASIS.MASK format is checked. With the -P44 option, the OASIS.MASK format is checked specifically. The maximum cell size (*elimit*) and hierarchy depth (*dlimit*) are checked when the option is specified. The top cell (*top*) is detected automatically if it is not specified.

If vboasisCheck encounters violations of the OASIS.MASK format, it reports them using several common error message formats. For instance:

- ERROR: Record ID *num* is not allowed in *OASIS_ref*
- ERROR: Standard Property, *property* is expected in *OASIS_ref*
- ERROR: Standard Property, *property* with value *x*is expected in *OASIS_ref*

- ERROR: Validation Scheme is expected in *OASIS_ref*
- ERROR: The unit value must be [*x*, *y*] in *OASIS_ref*

The OASIS_ref is the specific versions of the OASIS.MASK format followed by the chapter number of the functional specification where the violation was referenced.

For example:

```
Error: Record ID num is not allowed in OASIS.MASK P44-0708, 6.3.3.2.1.
```

In this example, OASIS.MASK P44-0708 refers to the version of the SEMI OASIS.MASK specification. The number following the specification (6.3.3.2.1) is the chapter number of the functional specification.

Job Decks to OASIS Files Conversion (Job Deck Smashing)

A Calibre MDPview batch command that converts one or more levels of a job deck into OASIS layers for input to a single Calibre hierarchical database. The OASIS file is output in STRICT mode format.

Usage

jobToOasis {*joblevelList* | all} {***jobtype*** [*malyjobfile*]} ***jobpath outpath*** [-unit *unitsize*] [-out_layer *outlayername*] [-exception_severity {ERROR | WARNING}] [-embeddedSVRF] [-noCblock]

Arguments

- ***joblevelList***

  Specifies the levels of the job deck to be converted. You can specify multiple levels, separated by commas. For a MALY job deck, you can specify a comma-separated list of mask names. You can also use a shorter notation to represent inclusive ranges of consecutive level numbers as shown in the following example:

  ```
  jobToOasis 1-10,15,17,21-26 MEBES abc123.jb abc123.jb.oas
  ```

- **all**

  Specifies that all levels of a job deck be converted. If MALY is specified, then all masks in a MALY job deck are converted.

- ***jobtype***

  Specifies the type of job deck and must be specified in all uppercase characters. The supported arguments are MEBES, MEBES_FLEX, JEOL, HITACHI, VSB, MICRONIC, and MALY.

  If you specify MALY, then you must also specify the MALY job file in the *malyjobfile* argument.

- ***jobpath***

  Specifies the location of the job deck and data. This can be either a relative or explicit path name.

- ***outpath***

  Specifies the path for the OASIS output file. The jobToOasis utility outputs a single OASIS file.

- -unit *unitsize*

  Specifies, in microns, the units with which the output data is created. It is suggested that you limit this value to less than one micron.

- -out_layer *outlayername*

  Specifies the number of output OASIS layers or layer-datatype pairs corresponding to input job deck levels. The output layer count must match the input level count. This parameter has the same syntax as *jobLevelList* (except for keyword **all**, which is not applicable). By default, the number of output OASIS layers match the number of input job deck levels.

- -exception_severity {ERROR| WARNING}

  Specifies the exception type. By default, exception_severity is set to ERROR. If any part of the input is missing, the program aborts with an error message. If set to WARNING, the program ignores the missing component of the input and continues to execute and generate the resulting OASIS output, but with a WARNING message.

- -embeddedSVRF

  Produces a rule deck that uses MDP EMBED with OASIS direct input rather than the default hierarchical-mode HDB-input rule deck.

  When specifying -embeddedSVRF with jobToOasis, an intermediate rule file with embedded SVRF is generated. When the data is reverse tone, that rule file must be run in order to get the final output. However, before the rule file can run, a dummy input file must be first generated with the appropriate PRECISION. Use a LAYOUT PLACE CELL statement to automatically generate a dummy input file. For example:

  ```
  LAYOUT SYSTEM OASIS
  LAYOUT PATH "placeholder_1000.oas"
  LAYOUT PRIMARY "*"
  ```

  by adding the statement:

  ```
  LAYOUT PLACE CELL "placeholder_1000.oas" OASIS <topcell> EMPTY
  ```

  The *placeholder_1000.oas*layout is generated automatically. The <*topcell*> is the name of the top cell in the output layout.

- -noCblock

  By default, cblock compression of OASIS for the output file is enabled by default. The -noCblock option deactivates the compression.

Description

Calibre MDPview provides you with functionality to convert one or more levels of the following job decks into OASIS layers for input into a single Calibre hierarchical database:

- MEBES
- MEBES_FLEX
- JEOL
- HITACHI
- VSB

- MICRONIC

This process is also known as “job deck smashing,” which means creating a single OASIS file from a job deck and its input pattern files. The sole purpose of this conversion is to allow reprocessing of a singe positive-tone job deck level representing the mask target through the section mode Calibre FRACTURE, MDPVERIFY or MDP EMBED commands.

Note

Overlapping chip placements are not supported; using reverse-tone markers in such a way may lead to incorrect output.

By default, a common database precision is computed for the resulting OASIS file. This number is the count of OASIS units per micron. The default value is computed as the smallest precision such that all job deck chips can be represented in that precision without error, or, if this precision is so large that the job deck extent overflows the Calibre MDPview coordinate limit, as the largest power of 10 that does not cause such overflow. If you provide the optional *unitSize* argument, specified in microns, the OASIS file precision is the inverse of this number (for example, 1.0 / *unitSize*).

For job decks with reverse-tone directives, the output OASIS file contains the reversed chips in normal tone as well as their corresponding extents. A Calibre nmDRC rule deck fragment that specifies boolean operations to combine the various OASIS layers into the normal-tone mask target is also output; the file name is the specified output file name with a “boolean.drc” suffix. This rule deck fragment should be used in the beginning of an embedded rule deck during section-mode processing in conjunction with OASIS direct input. It produces the mask target in each section, which may then be processed and re-fractured.

The jobToOasis transcript also lists the layer numbers of an output OASIS file. The following is an example taken from a transcript.

```
Input file: MEBES_files/E4XRSCALL.
Output layer numbers in OASIS file: 5 12 23 9 66 101
Finished conversion.
CONVERSION TIME: CPU TIME = 125  REAL TIME = 48
MEMORY: LVHEAP = 2/22/38.
```

The jobToOasis utility can convert extended MEBES job decks with OASIS zipped files. However, to be detected as a zip file, the file must have an extension *.gz* or *.Z*. Since the file name violates the MEBES namespace convention, set EJOBDECK_EXPANDED_NAMESPACE to YES to allow the files to be detected (see “Unrestricted Linux Pattern File Name” on page 144 for more information).

Note

The jobToOasis utility will not clip the OASIS file if the BX, BY, UX, UY specified in SREF or AREF is less than the original chip extent. It will also not write the Titles (similar to other job decks).

Examples

```
jobToOasis 1 MEBES ./mebes_jd_test ./oasis
```

In this example, Calibre MDPview places the output file in the “oasis” directory under the current run directory.

```
jobToOasis all MEBES test.jb test.jb.oas -out_layer 10:1,11:2,12:3,14:1
```

In this example, a number of output layer-datatype pairs (10:1, 11:2, 12:3, and 14:1) corresponding to the input job deck levels are specified.

Extended Formats

There are two mixed-format job decks supported by Calibre: MEBES_FLEX and the extended MEBES job decks.

- The MEBES_FLEX job deck allows a mix of pattern files of any format (MEBES, JEOL, HITACHI, VSB (NuFlare), MICRONIC, and OASIS) to be called within the job deck. To invoke this format for a “job deck smash”, use the MEBES_FLEX keyword with jobToOasis.
- The extended MEBES job deck allows a mix of OASIS and MEBES files to be called within the job deck, following the “MEBES extended job deck” syntax. To invoke this format, use either the MEBES or MEBES_FLEX keywords with jobToOasis. For example:

  ```
  jobToOasis 3 MEBES mebes_test.jb smashed_deck.oas
  ```

  You can also accomplish the same example from a command line:

  ```
  % calibremdpv -a jobToOasis 3 MEBES mebes_test.jb smashed_deck.oas
  ```

  Note

  OASIS.MASK files used in the Flex job view must comply with the format requirements for extended job decks.

Database Precision

By default, a common database precision is computed for the resulting OASIS file. This number is the count of OASIS units per micron. The default value is computed as:

- The smallest precision such that all job deck chips can be represented in that precision without error.

or

- If this precision is large enough that the job deck extent overflows the Calibre MDPview coordinate limit, the unit is represented as the largest power of 10 that does not cause an overflow.

If the *unitsize* argument is provided (specified in microns), the OASIS file precision is the inverse of this number (1.0 / *unitsize*).

For reverse tone job deck directives, an intermediate OASIS file is output with the referenced chips in normal tone and their corresponding chip extents. A DRC script, which performs the NOT Boolean operations between the extents and the chips, is also output. The final chip image can be obtained after executing the script.

The transcript log file reports the final file precision and conversion time at the end of the report (as shown in the following example):

```
Read MEBES job file: "./jobtooasis.jb"
...
...

MEBES infomation---------
input file name: ./DSGNTESTX.PF
file name in header: DSGNPOLYX.PF
mebes mode: 4
address size: 0.040000
stripe height:1024
chip size in x direction: 20102
chip size in y direction: 15806
number of segments: 1
number of records: 13
mask shop information: Poly layer
pattern data file creation date: 02/01/00
Successfully loaded MEBES job file.
OASIS file precision is 25 units per micron.

...
```

MEBES Job Deck to a VSB Job Deck Conversion

A Calibre MDPview command that converts a MEBES job deck to a VSB job deck. This process only changes the ASCII formatting, not the actual data.

Usage

jobconvert mebes2vsb *jobName* [-ln *layoutname*] [level *x*] [-vsb11 | -vsb12 | vsb12i] [-flip_mask_mirror]

Arguments

- ***jobName***

  A required argument that specifies the name of the job deck to be converted.

- -ln *layoutname*

  An optional character string without quotations that specifies the layout name for the output job deck.

  Note

  When converting MEBES job decks into VSB job decks, the MEBES job deck is treated as an exact mirror of a VSB job deck. If your MEBES job deck contains an MA (mask mirror) option, it is stripped from the job deck prior to conversion. If your MEBES job deck does not have the MA option, it is added in before the conversion. VSB data should pre-exist in the conversion directory and contain the same pattern file names as the patterns in the MEBES job deck.

- level *x*

  Optionally specifies the layer of the job deck you are converting. You can specify only one level of the job deck. The *x* parameter is an integer value. If level is omitted, all levels in the MEBES job deck are converted to corresponding MDS files.

- -vsb11 | -vsb12 | -vsb12i

  An optional keyword that specifies the VSB chip type to be placed in output VSB job deck. By default the tool considers VSB12 type chip placements in output VSB job deck.

- -flip_mask_mirror

  An optional keyword that enables flipping of the GLOBAL mask_mirror setting in the output VSB job deck.

  Note

  With this keyword, the input MEBES job deck may create an imperfect overlay with an output VSB job deck.

Description

This command converts a MEBES job deck to a VSB job deck. After the command is successfully run, the following are generated in the same path given for the input MEBES job deck:

- layout (job deck directory)

  If the “layout” job deck directory already exists, then an error message is issued and the conversion process stops.

- layout .ini file

  The *layoutID* is the same as its script name, excluding the “.mds” extension. The *layout_name* is the input *vsbLayoutName*, if provided, and the “level” option is specified. If multiple mask levels are being converted and MTITLE commands exist in the MEBES job deck, then the MTITLE string on the corresponding level is used as the layout name for that level. Otherwise, it remains the same as *layoutID*.

- *chip.ini* file

  The *chip_name* is the same as *chip_dir*. If the chip name starts with an alphanumeric letter, the *chipID* is the *chip_name*. If the chip name starts with a number, then the name is formatted as *chipID_+ chip_name*.

- Mask drawing script (MDS) file

  Each location section corresponds to one CHIP command in the MEBES job deck. The drawing name for the CHIP command in the MEBES job deck must match the VSB chip directory without conversion. All location sections in an MDS file are in the same order as the CHIP command order, as well as drawing chips in the order of chip location as specified in the Drawing section.

  The name is one of the following:

  - *VSB11_Layout_<level>.mds* for VSB11 chips.
  - *VSB12_Layout_<level>.mds* for VSB12 chips.
  - *VSB12I_Layout_<level>.mds* for VSB12i chips.

After the conversion, the VSB job deck can be loaded into Calibre MDPview directly. The following assumptions are made for conversion:

- The extent of the chip used during the conversion is the same as the extent of the actual VSB chips to be used in the VSB job deck.
- The output chip name used in the VSB job deck is not the same as the input chip name extracted from the MEBES job deck. Since VSB supports only a-z, A-Z, and 0-9 and the underscore (_), all the unsupported characters in the chip name are replaced by an underscore (_) in the output VSB job deck.

- Reverse-tone, scaling, and rotation of chips in the input MEBES job deck are not supported.

MEBES Job Deck to a Micronic Job Deck Conversion

A Calibre MDPview command that converts a MEBES job deck to a Micronic job deck. This process only changes the ASCII formatting, not the actual data.

Usage

jobconvert mebes2micronic *mebesJobPath micronicJobPath*
[-joblib *dir*] [-ef *val*] [-9_2.la] [-filter *cmd_name*]

Arguments

- ***mebesJobPath***

  Specifies the path to the MEBES job deck to be converted.

- ***micronicJobPath***

  Specifies the name of the output Micronic job deck.

- -joblib *dir*

  Optionally instructs the converter to explicitly insert the JOBLIB *dir* command into the output Micronic Maskset file. By default, JOBLIB is not written to output file.

- -ef *val*

  Optionally instructs the converter to explicitly insert the EF = *val* exposure option for all pattern files in PREPAREDCELL commands written to the output Micronic Maskset file.

- -9_2.la

  Optionally instructs the converter to perform automatic transformation of names of MEBES pattern files in accordance with CATS conventions.

  - For instance, if a chip in the input MEBES job deck has the name “AAAAAAA-BB-CC”, which corresponds to file on disk “AAAAAAABB.CC”, then the output Micronic Maskset file contains a reference to the Micronic pattern file named “AAAAAAABB_CC.la” (rather than name of the MEBES file “AAAAAABB.CC”, which is the default behavior). No pattern file conversion is performed during job deck conversion.

- -filter *cmd_name*

  Optionally instructs the converter to apply the given filter (specified in the form of Tcl command in the *.signaext* file) to each chip name in order to obtain path to corresponding pattern file.

To use the -filter option, you must first define a single-argument command (*cmd_name*) in the .*signaext*file. For example, the following code converts MEBES formats for chip names (7-2-2) to Micronic format names (9_2.la):

```
proc meb2mic {chip_name} {
   regsub {^([^-]{7})-([^-]{2})-([^-]{2})$} \
   $chip_name \
   {\1\2_\3.la} \
   file_name
   return $file_name
}
```

The return value for this example is the file name obtained from the chip_name (or the value of chip_name without changes if the transformation could not be performed). Then, pass the single-argument *cmd_name* (for example, meb2mic) using the -filter option.

```
calibremdpv -a jobconvert mebes2micronic a.jb a.ms -filter meb2mic
```

Calibre attempts to apply the filter to all chip names in whatever form they are currently using. The filter has the primary responsibility of recognizing and transforming the appropriate names. For example, the meb2mic filter in the example only transforms names in the form ABCDEFG-HI-JK; all other names (for example, mychip.la) are not transformed. If the resulting file following transformation still cannot be located, Calibre processes files as before (for example, it would pick the existing MEBES chip ABCDEFGHI.JK if the Micronic pattern file ABCDEFGHI_JK.la did not exist).

Description

This command converts a MEBES job deck to a Micronic job deck.

MEBES CHIP entries are transformed into corresponding Micronic PREPAREDCELL entries. However, this conversion cannot always be performed on a one-to-one basis because of the different approaches to the definition of mask writing steps utilized by the two formats. In MEBES, the order in which the levels are listed in the GROUP command is used to determine the proper mask writing sequence, and the Micronic sequence is defined by the order of the chip placement commands (ARRAY). Consequently, if a MEBES CHIP contains composite levels, it may have to be represented by several PREPAREDCELL entries in order to preserve the correct writing sequence in the output Maskset file.

No pattern file conversion is performed during job deck conversion. Therefore, PREPAREDCELL entries in the output Micronic maskset file reference patterns specified in the input MEBES job deck. While Calibre MDPview is able to handle such files, this may be incompatible with Micronic software. Furthermore, if any part of a path to a pattern file contains characters prohibited in the Micronic format, Calibre MDPview will not load the file. Refer to the Micronic maskset documentation for further details. The following limitations apply:

- Chip identification text is not preserved during conversion.

- The only supported exposure option is reverse_tone. All other options are not preserved during conversion.
- If a pattern file is referenced in the input MEBES file, but is not physically available on disk, the reference to the missing file is not preserved during conversion.

JEOL Job Deck to MEBES Job Deck Conversion

A Calibre MDPview command that converts a JEOL job deck to a MEBES job deck. This process only changes the ASCII formatting, not the actual data.

Usage

jobconvert jeol2mebes *jeol_job_path* [level *x*] ***mebes_job_path***

Arguments

- ***jeol_job_path***

  A required argument that specifies the path of the job deck to be converted.

- level *x*

  Optionally specifies the layer of the job deck you are converting. You can specify only one level of the job deck. The *x* parameter is an integer value. By default, the tool converts all levels in the input job deck.

- ***mebes_job_path***

  A required argument that specifies the output MEBES job deck.

Description

This command converts a JEOL job deck to a MEBES job deck. The following are assumptions made for conversion:

- The extent of the input JEOL chips used during the conversion is the same as the extent of the corresponding output MEBES chips.
- The output MEBES chipname to be used in the MEBESJOB deck is the same as the input chip name in the JEOLJOB deck.
- The layer name in the input JEOLJOB deck is written as MTITLE for that specific level in the output the MEBESJOB deck.
- The global mirror and local mirror from the input JEOLJOB deck are retained in output the MEBESJOB deck.

JEOL Job Deck to a VSB Job Deck Conversion

A Calibre MDPview command that converts a JEOL job deck to a VSB job deck. This process only changes the ASCII formatting, not the actual data.

Usage

jobconvert jeol2vsb *jobName* [-ln *layoutname*] [level *x*] [-vsb11 | -vsb12 | -vsb12i] [-flip_mask_mirror] [-flip_local_mirror]

Arguments

- ***jobName***

  A required argument that specifies the name of the job deck to be converted.

- -ln *layoutname*

  An optional character string without quotations that specifies the layout name for the output job deck.

- level *x*

  Optionally specifies the layer of the job deck you are converting. You can specify only one level of the job deck. The *x* parameter is an integer value. If level is omitted, all levels in the JEOL job deck are converted to corresponding mask drawing script (MDS) files.

- -vsb11 | -vsb12 | -vsb12i

  An optional keyword that specifies the VSB chip type to be placed in output VSB job deck. By default the tool considers VSB12 type chip placements in output VSB job deck.

- -flip_mask_mirror

  An optional keyword that enables flipping of the GLOBAL mask_mirror setting in the output VSB job deck. This keyword flips data with regards to the center of the mask.

  Note

  With this keyword, the input JEOL job deck may create an imperfect overlay with an output VSB job deck.

- -flip_local_mirror

  An optional keyword that flips all chips in the output VSB job deck. This keyword flips each chip in the job deck with regards to the center of the chip.

  Note

  With this keyword, the input JEOL job deck creates an imperfect overlay with the output VSB job deck.

Description

This command converts a JEOL job deck to a VSB job deck. After the command is successfully run, the following are generated in the same path given for the input JEOL job deck:

- layout (job deck directory)

  If the “layout” job deck directory already exists, then an error message is issued and the conversion process stops.

- layout .ini file

  The *layoutID* is the same as its script name, excluding the “.mds” extension. The *layout_name* is the input *vsbLayoutName*, if provided, and the “level” option is specified.

- *chip.ini* file

  The *chip_name* is the same as *chip_dir*. If the chip name starts with an alphanumeric letter, the *chipID* is the *chip_name*. If the chip name starts with a number, then the name is formatted as *chipID_+ chip_name*.

- Mask drawing script (MDS) file

  The name is one of the following:

  - *VSB11_Layout_<level>.mds* for VSB11 chips.
  - *VSB12_Layout_<level>.mds* for VSB12 chips.
  - *VSB12I_Layout_<level>.mds* for VSB12i chips.

After the conversion, the VSB job deck can be loaded into Calibre MDPview directly. The following assumptions are made for conversion:

- The tool does not assume the presence of output VSB chips.

  The extent of the chip used during the conversion is the same as the extent of the actual VSB chips to be used in the VSB job deck.

- The output chip name used in the VSB job deck is not the same as the input chip name extracted from the JEOL job deck. Since VSB supports only a-z, A-Z, and 0-9 and the underscore (_), all the unsupported characters in the chip name are replaced by an underscore (_) in the output VSB job deck.

- Reverse-tone, scaling, and rotation of chips in the input JEOL job deck are not supported.

VSB Job Deck to MEBES Job Deck Conversion

A Calibre MDPview command that converts a VSB job deck to a MEBES job deck. This command only changes the ASCII formatting, not the actual data.

Usage

vsb2mebes *vsbJobPath* [-ln *vsbLayoutName*] ***mebesJobPath***

Arguments

- ***vsbJobPath***

  A required argument that specifies the input VSB job deck.

- -ln *vsbLayoutName*

  An optional argument that generates the MEBES job deck for a single mask in a VSB deck.

- ***mebesJobPath***

  A required argument that specifies the path to the output MEBES job deck.

Description

This command converts a VSB job deck to a MEBES job deck. The default vsb2mebes command converts all VSBJOB deck masks to separate levels in the MEBESJOB deck. It assigns a level number, starting from 1, in the order of masks present in the *layout.ini* file.

The following requirements provide a successful conversion:

- The extent of input VSB chips used during the conversion must be same as the extent of the corresponding output MEBES chips.
- The output MEBES chip name used in the MEBES job deck must be the same as the input chip name in the VSB job deck.

The layout_name in the input VSB job deck is written as MTITLE for that specific level in the output MEBES job deck. The global mirror and local mirror from the input VSBJOB are retained in the output MEBES job.

Note

The tool does not assume the presence of output MEBES chips. The array placements in the input VSB job deck are not guaranteed in the output MEBESJOB deck.

Text to QR Code Conversion

A Calibre WORKbench batch command that creates a 2D barcode (QR code Model 2, version 4) and converts it to OASIS data.

Usage

text2qr {***input_file*** | **-string** ***string***} ***output_file*** **-modulesize** ***value*** **-layer1** ***layer***[.*datatype*] [-layer2 *layer*[.*datatype*]] [-ecl *value*] [-precision *value*]

Arguments

- ***input_file***

  A required parameter that specifies that a text file be used as input. The text file contains alphanumeric characters to be converted to QR code. Either ***input_file*** or **-string** must be used, but not both.

- **-string*string***

  A required parameter that specifies that input be entered as an alphanumeric input string on the command line. Either ***input_file*** or **-string** must be used. A set of 45 characters is allowed as input: 10 numeric digits (0 - 9), 26 uppercase alphabetic characters (A - Z), and 9 symbols (space, $, %, *, +, -, ., /, :). The input string can be specified in single quotes (‘ ’), double quotes (“ ”), or without quotes. However, if the input string contains special characters (space, $, *), the string must be enclosed in single quotes and the special character must be escaped with the backslash character.

- ***output_file***

  A required parameter that specifies the output OASIS file path.

- **-modulesize *value***

  A required parameter that specifies the desired edge length, in microns, of each dark and light square module that comprise the QR code. This parameter accepts decimal values as long as the product of **-modulesize** and -precision is an integer.

- **-layer1 *layer***[.*datatype*]

  A required parameter that specifies the OASIS layer number on which the dark modules of the QR code are placed. You can optionally specify a datatype number; if not, it will default to 0.

- -layer2 *layer*[.*datatype*]

  An optional parameter that, when specified, places only light modules on a separate layer. You can optionally specify a datatype number; if not, it will default to 0. The layers specified for**-layer1**and -layer2 cannot be the same. This layer can be used as a reverse tone of the QR code.

- -ecl *value*

  An optional parameter that specifies the level of error correction code. Valid values are as follows:

  L — Provides 7% of error correction

  M — Provides 15% of error correction

  Q — Provides 25% error correction

  H — Provides 30% of error correction. This is the default value.

  Note

  The command only supports version 4 of the QR code specification by default.

- -precision *value*

  An optional parameter which defines the precision of the output OASIS file. The default value is 1000. It must be a positive integer and the product of -precision and **-modulesize** value must be an integer.

Description

The text2qr command creates a 2D barcode (QR code Model 2) from alphanumeric input and converts this barcode to OASIS data. This command only supports version 4 of the QR code specification.

For example, a 2D barcode can be used to encode reticle identification information on masks created for EUV steppers.

Examples

```
text2qr -string “QR-DUMP12” output.oas -modulesize 20 -layer1 1.0 \
   -layer2 2.0 -ecl H -precision 1000
```

Index Files

Certain databases (OASIS) and pattern files (Micronic and MEBES) can be “pre-indexed” to speed job deck loading and viewing in Calibre MDPview. Refer to the following sections for descriptions on how to pre-index your files.

OASIS File Indexing

A Calibre WORKbench batch command that pre-indexes an OASIS file to speed job deck loading and pattern file viewing.

Usage

oasisIndex ***oasis_file_path*** [*top_cell_name*] [-noView] [-fastIndex [*layer datatype*]] [-scale {MASK | WAFER}] [-sequential] [-saveLayerExtents] [-parallelCblocks]

Arguments

- ***oasis_file_path***

  A required argument that specifies the path to the OASIS file.

- *top_cell_name*

  Optionally specifies the top cell name.

- -noView

  Optionally specifies that the indexing omits some computations intended only to improve interactive viewing speed. The index is made much more quickly, but the drawing speed in the viewer is substantially slower. Use this option if the file is only intended for direct input to Calibre MDP functions.

- -fastIndex [*layer datatype*]

  An optional keyword that enables a “fast mode” of index generation if the input file is an OASIS.MASK file and the option -noView is specified. The file created in fast mode is valid only if the input OASIS.MASK file contains a single layer-datatype pair. Fast mode index generation should be avoided when there are multiple layers or datatypes in the input. By default, fast mode index generation assumes layer-datatype of 0-0. You can override these values by specifying layer-datatype pair as an argument to the -fastIndex option.

- -scale {MASK| WAFER}

  If the index is intended for viewing, you can assist the indexing process by specifying the scale of the layout. Use the -scale option to state whether the layout is at wafer or mask scale. No other values are currently accepted. This is an inherently difficult quantity to estimate automatically, and knowing the value produces an index with a better cost-benefit factor. MASK is the default setting.

- -sequential

  The indexing operation uses a probability calculation algorithm to improve performance. With the -sequential option, the entire layout is scanned sequentially, ignoring the cell-offset table.

- -saveLayerExtents

  Allows a bounding box for each layer in the top cell to be written in the generated index file. Keeping per-layer bounding box information in the index file for the top cell helps to speed calculation of extents.

- -parallelCblocks

  Allows parallel indexing for non-strict cblock OASIS data. The CBLOCKS cell is parsed in parallel during sequential parsing of non-strict OASIS files for faster index creation.

Description

OASIS files in extended MEBES job decks can be “pre-indexed” to speed job deck loading. A separate index file named “*orig_file*.fvi” is created and must be kept in the same directory as the OASIS file. (The .fvi index files are not backwards-compatible with previous INDEX versions; they are, however, compatible with files generated by previous Calibre WORKbench versions as long as the separate index version is unchanged).

You can change the location in Calibre WORKbench (for instance, with the *.signaext* file) by specifying the Tcl environment variable MDPIndexSearchPath. If you are defining this variable in the *.signaext* file, the syntax is:

```
set env(MDPIndexSearchPath) dir1:dir2:dir3
```

The Linux shell version of this environment variable is:

```
setenv MDPIndexSearchPath dir1:dir2:dir3
```

The specified paths separated by colon (**:**) is searched first, then the OASIS file location is checked. The same algorithm is applied when creating the index file.

The Tcl variable always takes precedence over the Linux shell variable.

The next time the file is viewed, the viewer checks for the index file and use it. If the index file does not exist or appears to be in any way wrong, the viewer automatically recomputes the index before loading. When index files are used, loading any OASIS file generally takes a few seconds.

As generating an index can sometimes take a significant amount of time, you should generate the index (creating the .fvi files) on any file that you intend to read with the disk resident oasis reader(s). This should occur in preprocessing, prior to loading extended job decks with multiple OASIS files.

In general, an OASIS file cannot be parsed in parallel except for one exception: if an OASIS file is in “strict mode” and has a cell-offset record for every cell, the file can be parsed in parallel. The indexing function does this, achieving scalability usually limited only by file system access.

Several conditions can cause an existing index to be deemed invalid and need to be replaced. Index versions are never forward-compatible. Only versions with the same major version number are backwards-compatible. Check the Release Notes to see when the index version changes. Using viewer releases with different index major versions can thus cause the index to

be remade. Several characteristics of the OASIS file and index file themselves can also force the re-creation of an index:

- The index does not exist.
- The index file cannot be read.
- The length of the OASIS file does not match the length recorded in the existing index. An OASIS file may have changed due to several possible reasons including:
  - Adding or removing geometries
  - Altering the hierarchy (oasis2oasis)
  - Clipping the OASIS file

  In any of these cases, the input OASIS file has been modified and the index file is regenerated. For example, with hierarchy optimization, although the input OASIS file and output OASIS file are the same, the hierarchy of the OASIS file changed, leading to index regeneration.
- Selected samples of the OASIS file do not match the record in the existing index. This occurs in cases where the OASIS file has been overwritten or altered. For example, you have an OASIS file *a.oas* with its respective index file *a.oas.fvi*. If an alternate OASIS file *b.oas* is renamed to *a.oas*, the contents of *a.oas* and *a.oas.fvi* no longer match and the index is regenerated.
- The current version of the index (depending on the Calibre version set) differs from the index file version.

The default behavior when the input layout contains an undefined cell is to treat the cell as empty and continue the process. By specifying the Tcl variable:

::MDP_MISSING_REFERENCE {0 | 1| 2}

you can specify how the error is handled. There are three possible settings:

- 0 — continues the operation without issuing a message
- 1 — continues the operation and issues a message (this is the default behavior)
- 2 — aborts the indexing operation with an error message

The validity of an index for an OASIS file can be checked by invoking the command:

oasisIndexInfo [-v] *oasis_file_path*

Note that the command argument is the OASIS file, not the index file. This command states whether or not an index file exists, the version of any such file and the version used by the current executable. Finally the command issues a statement if the index is not valid according to

the previously-listed criteria. The -v option outputs the size of the file in a transcript and includes the Remote size and Flat Vertex count.

This table lists OASIS index versions used by their respective Calibre release.

Table 6-1. OASIS Index Version per Release

Calibre Release Version	OASIS Index Version
2016.1- Current	3.0
2015.4	2.11
2015.3	2.10
2015.2	2.9
2015.1	2.7
2014.1-2014.4	2.6
2011.2-2013.4	2.5
2009.3-2011.1	2,4
2009.2	2.3
2009.1	2.2

The on-disk hierarchy injection data can also be precomputed using the following command:

oasisInjection *input_file_name* [-size *bin_size*] [-force]

If an index does not exist when this command is invoked, one is created with a minimal set of options. Creation of the persistent index file can be controlled for OASIS.MASK files only by using the MDP_INDEX_MASK_FILES Tcl variable. See “Control Index File Generation” on page 149 for complete information.

Additionally, the CALIBRE_REGENERATE_INDEX_TIMESTAMP_BASED Tcl variable, when set to true, detects if an index is created after OASIS creation. If the index is detected, the index is regenerated.

The injection parameters are controlled by the Tcl and Linux shell environment variables described in “VBOASIS Injection” on page 146. The -size of the cells that should be binned and the size of each bin (bin_size) can be explicitly specified using the -size option in microns. The default value is 250 microns.

Note

The index file should not be created if it exists no matter what parameters are specified with the vboasis_injection option.

Injection data is not created if it already exists; to overwrite the existing data, use the -force option. The -force option also recreates an .fvi file if one already exists, though it drops the viewer components.

Note

When you generate an index for a layout larger than 4 GB the 64-bit platform should be used. In 32-bit platform an integer overflow can happen and it results in an invalid index.

You can perform RVE hierarchical layer scans on disk resident OASIS layouts from Calibre MDPview. However, if you generate your own RVE database files for this purpose, the rule file DRC CELL NAME statement should not specify the CELL SPACE XFORM options. This causes RVE to instruct Calibre MDPview to open the appropriate cell for each cell's instances, and opening a cell is not supported within disk resident OASIS viewing.

Alternatively, the **RVE Highlight > Highlight in Context** option can be turned off. You should deactivate this option for any hierarchical layer scans of disk resident OSIS data.

Note

For multithreaded processing, delete the existing *.fvi* file when you create a new OASIS file. This prevents the following risks: dropped-data for direct-database-entry fracture, missed or false flags in direct-data-entry MDPVERIFY*format*2DB operations, and data dropping from view in disk-based viewing such as *.ejb* for extended job deck viewing and **View Oasis Layout**.

Examples

These examples demonstrate the use of oasisIndex (to generate the OASIS index) and oasisIndexInfo (to report information on the indices), both the command line invocation and output:

- Create an OASIS index with Calibre 2017.4, OASIS index version 3.0:

```
calibremdpv -a oasisIndex file.oas | tee oasisIndex.file.log

Creating an index, version: 3.0.
SEG TIME: CPU TIME = 0  REAL TIME = 0
MEMORY: LVHEAP = 3/3/4.
Sequential parsing of OASIS file.
PARSE TIME: CPU TIME = 0  REAL TIME = 0
MEMORY: LVHEAP = 3/59/59.
NOTE: Cell GUNTER was automatically selected as the top cell.
NOTE: REMOVED 89 cells, 143 single, 49 array placements.
Note: Viewer components are present.
POST PROCESSING TIME: CPU TIME = 0  REAL TIME = 0
MEMORY: LVHEAP = 3/59/59.
Opening an index file, file.oas.fvi.
INDEX OUTPUT TIME: CPU TIME = 0  REAL TIME = 0
MEMORY: LVHEAP = 3/59/59.
```

- Report OASIS index version:

```
calibremdpv -a oasisIndexInfo file.oas | tee oasisIndexInfo.file.log

Opening an index file, file.oas.fvi.

Note: Existing index file version: 3.0.  Current version: 3.0.
Note: Viewer components are present.
```

- Report OASIS index version and more verbose information:

```
calibremdpv -a oasisIndexInfo file.oas -v | tee \
       oasisIndexInfo-v.file.log

Opening an index file, file.oas.fvi.

Note: Existing index file version: 3.0.  Current version: 3.0.
Note: Viewer components are present.
```

Micronic File Indexing

A Calibre WORKbench batch command that pre-indexes a Micronic file in order to speed loading and viewing.

Usage

micronicIndex *patternFile* [-granularity *threshold*]

Arguments

- ***patternFile***

  Specifies the path to the Micronic pattern file.

- -granularity *threshold*

  An optional value used to achieve a suitable compromise between index size and window query speed.

  Typically, the *threshold* value suggests the maximum area (in um^2) of parts of the layout that is treated "atomically". The default value is 250000 um^2. If the expected query window size is rather small (such as in Calibre MDPview at low altitudes), and queries are expected to be performed frequently, you can consider smaller granularity values (for example, 10000 um^2). However, if the expected query window size is relatively large and the intensity of window queries is rather low (such as in Calibre MDPverify), then the default value is suitable.

Description

The micronicIndex command creates the persistent index for a given Micronic pattern file in the directory of that pattern file. The index file is named *patternFile*.mvi and is then used automatically whenever the pattern file is loaded. If the index file cannot be located, or if it appears to be irrelevant (for example, due to the pattern file change), it is created or updated automatically. The general conditions under which OASIS index files are re-computed also apply to Micronic indices.

This table lists the Micronic index versions used by their respective Calibre release.

Table 6-2. Micronic Index Version per Calibre Release

Calibre Release Version	VSB12 Index Version
2016.1- Current	1.0
2009.2-2015.4	0.2
2009.1	0.1

MEBES File Indexing

A Calibre WORKbench batch command that pre-indexes a MEBES file to speed job deck loading and pattern file viewing.

Usage

mebesIndex *input_file* [-noView] [-preserve]

Arguments

- ***input_file***

  A required argument that specifies the path to the MEBES pattern file.

- -noView

  An optional argument that generates an index file that contains only stripe offset data, but not pre-calculate views. The resulting MEBES index file is smaller than with pre-calculated views which may be useful for batch processing of files in Calibre MDPverify, but does not allow the user to have a fast high-altitude view of the MEBES data in Calibre MDPview. This aligns mebesIndex behavior with oasisIndex behavior.

- -preserve

  If -preserve is specified and a compatible MEBES index file exists, then the index file is not be overwritten and no new index file is generated. This allows users to assert a command to pre-index MEBES data in their automation, but does not penalize them if a compatible index file already exists.

Description

MEBES pattern files can be pre-indexed in order to speed up viewing them both individually and as part of job deck. Each time the MEBES pattern file is viewed, the viewer checks for the index file and uses it, if it exists. The mebesIndex utility creates a MEBES index file named *inputfile*.xvi. During viewing, this newly created MEBES index file is used to speed up the viewing of this MEBES pattern file both individually and as part of a job deck. The *inputfile*.xvi has both stripe offset data and viewer components when generated using mebesIndex command. If the *inputfile*.xvi file is generated using any other method, it does not have viewer components and the performance benefits are sub-optimal.

You can control index generation using one of the following methods:

- MDP_INDEX_MASK_FILES {ALL| NONE | *size_in_MBs*} (Tcl Command)
- -index_mask_files {ALL| NONE | *size_in_MBs*} (Command-line option and Calibre MDPverify)

where ALL automatically generates index files for any MEBES pattern file (default), NONE forces the tool to not generate any index file, and *size_in_MBs* specifies a threshold size in MBs for the MEBES pattern file. For any MEBES pattern file that is greater than *size*, an index file will be generated.

The older versions of mebesIndex commands generated a *inputfile*.idx file that can be generated by setting the Linux environment variable MDP_MEBES_READER_VERSION_VALUE to 1.

This command can also be used to create an index file that does not have viewer components but only stripe offset data. To do so specify the optional argument -noView. Such an index file can be used in other MDP products such as Calibre MDPverify.

If any index file is present, the default behavior is to overwrite it and create a new file. However if -preserve is specified and a compatible MEBES index file exists then it is not overwritten and no new file is generated.

If no index file exists (and automatic indexing is turned off) or if the one specified cannot be opened or is out-of-date, then the viewer will continue to work without an index but issues a warning.

If the index file exists but its checksum does not match the information in the header of the index file (due to file corruption), if the value of Linux variable MDP_MEBES_INDEX_FILE_EXCEPTION_POLICY is set to “2,” a fatal error is issued; otherwise, a new index file is created with a warning message.

This table lists the MEBES index versions used by their respective Calibre release.

Table 6-3. MEBES Index Version per Calibre Release

Calibre Release Version	MEBES Index Version
2016.1 - Current	5.0
2013.2 - 2015.4	4.1
2010.4 - 2013.1	4.0
2010.3	3.2
2009.2-2010.2	1.2
2009.1	1.1

JEOL File Indexing

A Calibre WORKbench batch command that pre-indexes a JEOL file in order to speed loading and viewing.

Usage

jeolIndex *inputfile* [-preserve]

Arguments

- ***inputfile***

  Specifies the path to the JEOL pattern file.

- -preserve

  If enabled, instructs Calibre not to overwrite an existing index file and no new file is generated. By default, if any index file is present, then it is overwritten by a newly-created index file.

Description

The jeolIndex utility generates an ***inputfile***.jvi index file. During viewing, this index file is used to speed up the viewing of the JEOL pattern file both individually and as part of a job deck. Viewer components are not included. This index file can be used in other MDP products such as Calibre MDPverify. Each time the JEOL pattern file is viewed, the viewer checks for the index file and uses it, if it exists.

You can change the location of the index file in Calibre WORKbench (for instance, with the *.signaext* file) by specifying the Tcl environment variable MDPIndexSearchPath. If you are defining this variable in the *.signaext* file, the syntax is:

```
set env(MDPIndexSearchPath) dir1:dir2:dir3
```

The Linux shell version of this environment variable is:

```
setenv MDPIndexSearchPath dir1:dir2:dir3
```

The specified paths separated by colon (**:**) is searched first, then the OASIS file location is checked. The same algorithm is applied when creating the index file.

The Tcl variable always takes precedence over the Linux shell variable.

If no index file exists or if the one specified cannot be open or is out-of-date, then viewer will work without an index (but issue a warning).

This table lists the JEOL index versions used by their respective Calibre release.

Table 6-4. JEOL Index Version per Calibre Release

Calibre Release Version	JEOL Index Version
2016.2 - Current	2.1
2016.1	2.0
2011.4-2015.4	1.0

VSB File Indexing

A Calibre WORKbench batch command that pre-indexes a VSB11 or VSB12 file in order to speed loading and viewing.

Usage

vsbIndex *chipDirectory* [*indexFile*]

Arguments

- ***chipDirectory***

  Specifies a path to a directory containing the VSB11 or VSB12 chip.

- *indexFile*

  Specifies the name of the index file.

Description

VSB chips may be pre-indexed in order to speed viewing them both individually and as part of job deck. Each time the chip is viewed, the viewer will check for the index and use it. In contrast with OASIS files, the viewer will **not** index VSB chips automatically before loading. This was done as indexing could slow things down significantly in cases if you are only interested in viewing VSB chips in “extents only” mode, or if the you only want to look at a small portion of the chip. The viewer will not create a VSB index automatically at load time.

This table lists the VSB12 index versions used by their respective Calibre release.

Table 6-5. VSB12 Index Version per Calibre Release

Calibre Release Version	VSB12 Index Version
2016.1 - Current	2.0
2009.2 - 2015.4	1.2
2009.1	1.1

This table lists the VSB11 index versions used by their respective Calibre release.

Table 6-6. VSB11 Index Version per Calibre Release

Calibre Release Version	VSB11 Index Version
2016.1 - Current	2.0
2009.2 - 2015.4	1.2
2009.1	1.1

Appendix A
MEBES Job Deck Warning Messages

When you load a MEBES job deck, the application parses the job deck file, looking for commands it recognizes. Should it encounter a string or command it does not recognize, it responds by writing a warning message to the log file. Note that this does not imply that there is a problem with the file.

The viewer also issues warning messages when it encounters certain machine-specific limitations or other situations you may want to know about. The run is not compromised, and these warnings do not signal any need for user action.

For more information on how specific commands may influence mask exposure, refer to "MEBES Data Files," Etec Document 0900-0024, available from Applied Materials, Co., Hillsboro, Oregon.

List of MEBES Warning Messages

The following table lists the possible warning messages that can be issued by Calibre MDPview.

Table A-1. MEBES Job Deck Warning Messages Summary

"WARNING: chip <chip name string>: input AU = <au value>, exposure AU = <au value>"
"WARNING: command <string> is not supported in display at line <line_number value>"
"WARNING: title <title string> at x = <x location>, y= <y location> is not inside work piece area"
"WARNING: option <option string> is not supported for <command string> command at line <line_number value>"
"WARNING: always do arbitrary overlap instead of double-fetch at line <line_number value>"
"WARNING: chip identification included non-alphabetic-numeric character, it won't work on older mebes machine at line <line_number value>"

Table A-1. MEBES Job Deck Warning Messages Summary (cont.)

“WARNING: title name is not supported in ORIENT command at line <line_number value>”
“WARNING: record length is greater than 80 at line <line_number value>”
“WARNING: title character <character> is not allowed and will be ignored at line <line_number value>”
“WARNING: circular array limit commands are no longer supported on the MEBES/ ALTA platforms”

Appendix B
Calibre MDPview Tcl Extensions

The *wbinit.tcl* file is an application extension file that contains Tcl-based macro scripts to modify the behavior of the Calibre MDPview and Calibre WORKbench layout viewers.

The *wbinit.tcl* file is described in the *Calibre DESIGNrev User's Manual*.

Note

The *wbinit.tcl* file is the current extension file to be used instead of the older *.signaext* extension file.

The file must be created by the user and placed in the following directories where it is sourced by the layout viewer in the following order:

- *$HOME/.calibrewb_workspace/.wbinit.tcl*
- The directory where the layout viewer is invoked.

View Extended Job Decks Without Data Clipping

Typically with extended job decks, when you designate the coordinates of the input data, the data cannot extend beyond the described window (clipping data beyond window boundaries).

However, if you want to display a file with data larger than the described extent in the job deck, you can use the *wbinit.tcl* file to allow the display of data larger than the designated window.

In your *wbinit.tcl* file, you can add the following environment variable:

::EJOBDECK_CLIPPING_SEVERITY {ERROR | WARNING | IGNORE}

where:

- ERROR — Produce an error message (because the data is beyond the window boundary) and does not load the job deck.
- WARNING — Allows the extended job deck to load with a warning message issued.
- IGNORE — Allows the job deck to be displayed without issuing any warning, ignoring the error condition. This considers data clipping a normal situation.

Alternatively, the WARNING setting can be issued without the *wbinit.tcl* file from the Calibre MDPview command line:

```
setenv EJOBDECK_CLIPPING_SEVERITY WARNING
```

However, if EJOBDECK_CLIPPING_SEVERITY is already defined in the *wbinit.tcl* file, that setting takes precedence.

Unrestricted Linux Pattern File Name

You can specify a pattern file with an unrestricted Linux style name using the EJOBDECK_EXPANDED_NAMESPACE variable. This allows you to have unrestricted file names for OASIS files. This option cannot be simultaneously specified with EJOBDECK_PATTERN_SUFFIX.

set ::EJOBDECK_EXPANDED_NAMESPACE {YES | NO}

Specifying YES (default) enables unrestricted Linux-style names; specifying NO keeps the restrictions.

The priority order by which Calibre MDPview searches for a particular extended MEBES file name is described in “Pattern File Name Recognition in MEBES Job Decks” on page 59.

Nonexistent Pattern File Warning Message Severity

Typically an error message is printed to standard output if a job deck defines a nonexistent pattern file, but ignores all references to the file later. This behavior can only be changed for MEBES job decks.

In your *wbinit.tcl* file, you can control error messaging using the following environment variable:

::EJOBDECK_NOCHIP_SEVERITY {ERROR | WARNING | IGNORE}

where:

- ERROR — Does not allow loading with missing chips. The tool terminates.
- WARNING — Allows loading of the job deck, but with warnings for missing chips.
- IGNORE — Allows loading of the job deck by ignoring the missing chips. No warning messages are issued.

Incorrect Chip Placement Warning Severity Level

You can control the behavior of Calibre when it encounters chips that are off the reticle (inside substrate at placement).

In your *wbinit.tcl* file, you can control the severity level using the following environment variable:

::EJOBDECK_OUTSIDE_SUBSTRATE_SEVERITY {ERROR | WARNING |IGNORE}

where:

- ERROR — Aborts the run and issues an error message.
- WARNING — Issues a warning message and continues the run. This is the default setting.
- IGNORE — Continues the run without issuing messages.

View OASIS Pattern Files in MEBES Extended Job Decks

OASIS pattern files with a .oas extension can be read in MEBES extended job decks if you add the following line to the *wbinit.tcl* file (either locally or in your user directory):

set ::EJOBDECK_PATTERN_SUFFIX oas

As an alternative, you can also view extended job decks using the flexible MEBES job decks options.

The same naming rules as used for MEBES pattern files apply, but you can use the .oas suffix instead (for example, FRAMEXXXX.OAS instead of FRAMEXXXX.PF).

When EJOBDECK_PATTERN_SUFFIX and 7-2-2 format are specified in the job deck, the search priority is as follows:

1. 9_2.oas
2. 9.2

VBOASIS Injection

Some layouts may have very large cells with a lot of data which may degrade viewing performance. To improve the efficiency of viewing (including disk-resident OASIS viewing), you can use a feature known as “VBOASIS injection.” Though this injection takes additional run time, subsequent viewings can be faster. VBOASIS (Vector-Based OASIS) is a restricted form of OASIS DDE (Direct Data Entry) files.

This feature is controlled by the following TCL variables specified in the *wbinit.tcl* file:

- MDP_VBOASIS_INJECTION {1 | 0} — Setting this variable to 1 enables VBOASIS injection.
- MDP_VBOASIS_INJECTION_SIZE val_in_microns — This variable specifies the bin size in microns.
- MDP_VBOASIS_INJECTION_PRESERVE {1 | 0} — This specifies whether existing VBOASIS injection has to be re-used or not. The default value is 1 and the existing VBOASIS bin injection is preserved. When set to 0, existing bin injection is ignored and bin injection is performed again. Note that bin injection can take a significant amount of time.

Setting the Location of the Index File

OASIS files in extended MEBES job decks can be “pre-indexed” to speed job deck loading. A separate index file named “*orig_file*.fvi” is created and must be kept in the same directory as the OASIS file. However, the search location of the bin injected files and .fvi files can be changed by specifying the environment variable MDPIndexSearchPath. The path for a data search is also defined using the environment variable MDPDataSearchPath.

If you are defining the MDPIndexSearchPath in the *wbinit.tcl* file, the syntax is:

set env(MDPIndexSearchPath) *dir1*:*dir2*:*dir3*

The specified paths separated by colon (**:**) are searched first, then the OASIS file location is checked. The same algorithm is applied when creating the index file.

The next time the file is viewed, the viewer checks for the index file and uses it. If the index file does not exist or appears to be in any way incorrect, Calibre aborts the display if there is no write permission or MDPIndexSearchPath was not otherwise set. When index files are used, loading any OASIS file generally takes a few seconds.

If you are defining the MDPDataSearchPath in the .*signaext*file, the syntax is:

set ::MDPDataSearchPath *dir1*:*dir2*:*dir3*

The following example demonstrates how “temp” data (.fvi and bin injected data) can be stored in a writable temp directory, while the “golden” data (OASIS files) are stored in a read-only directory.

In this scenario:

- Pattern file and job decks are in a read-only directories, with writable subdirectories that hold “disposable data,” the indices and bin injection files. The path for an index search is defined using MDPIndexSearchPath.
- The example chip (tsisram40_00.oas) is stored in working_directory. A peripheral chip (tsisram40_01.oas) stored in secondary_storage. The path for a data search is also defined using MDPDataSearchPath.
- Both the working_directory and secondary_storage are read-only. The only directory that is writable is the working_directory/calibre_temp specified in MDPIndexSearchPath.

Procedure

1. Start in the working_directory.

```
$ cd working_directory
```

2. Invoke Calibre MDPview, which loads the *wbinit.tcl* file. The following example *wbinit.tcl* file shows the bin injection and Data/Index search path settings.

```
   set ::EJOBDECK_CLIPPING_SEVERITY WARNING
   set ::EJOBDECK_PATTERN_SUFFIX oas
   set ::MDPDataSearchPath ../working_directory:../
secondary_storage:../third_storage
   set env(MDPIndexSearchPath) ../working_directory/calibre_temp
   set ::MDP_VBOASIS_INJECTION 1
   set ::MDP_VBOASIS_INJECTION_SIZE 10
   set ::MDP_VBOASIS_INJECTION_PRESERVE 1
```

3. Check the permissions of the file structure. For instance:

```
$ ls -l .. ../* ../*/*
```

The working_directory and secondary_storage are read-only (chmod 555) and the subdirectory (calibre_temp) is writable (chmod 777).

```
$ chmod 555 ../working_directory ../secondary_storage
$ chmod 777 ../working_directory/calibre_temp
```

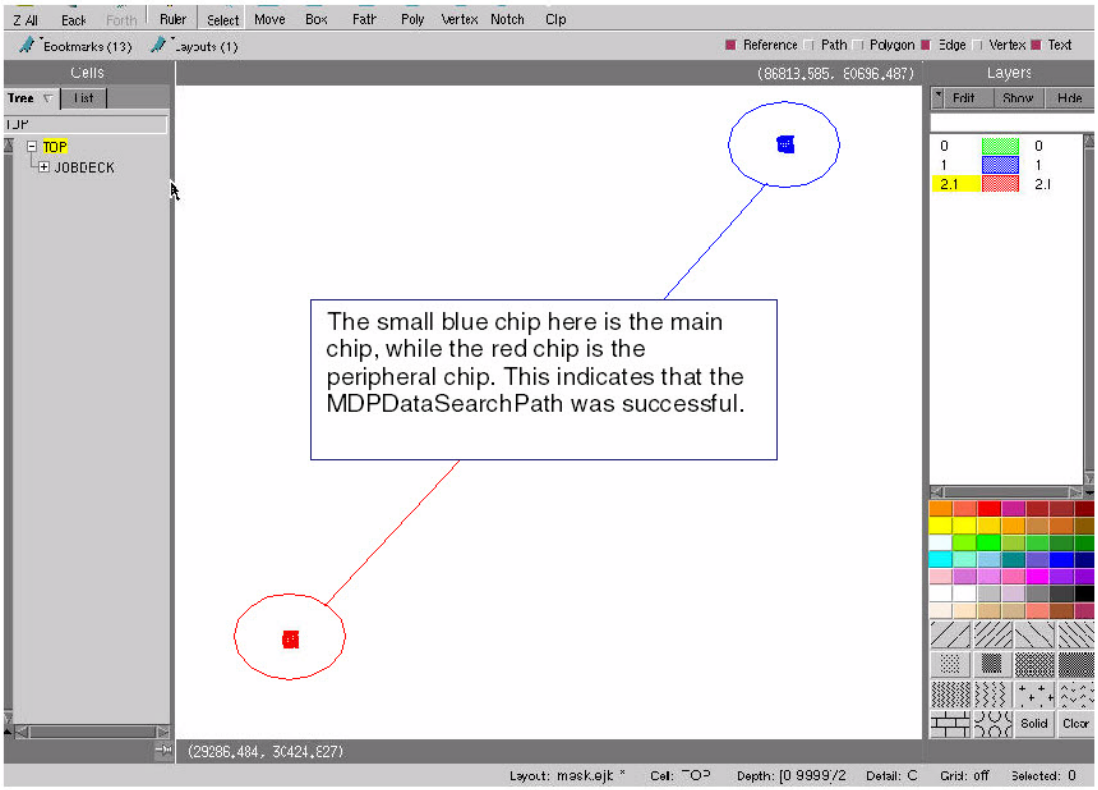
4. Open the job deck in Calibre MDPview.

   a. In Calibre MDPview, select **File >View Mask Jobdeck > Flex. MEBES.** In the dialog that appears, select mask.ejb (an example extended job deck) in the working_directory (make sure the **File Type** is set to **All Files**), then click **Open**.

   b. Set Calibre MDPview to the lowest view depth by typing “99.”

      Calibre MDPview displays two chips: blue is main chip, red is peripheral, indicating that the MDPDataSearchPath was correctly followed.

Figure B-1. Results of MDPDataSearchPath



5. Exit the viewer by selecting **File > Close Window > No > OK**.

6. Once again, check the permissions.

```
$ ls -l .. ../* ../*/*
```

The results are similar to the following:

```
...
../working_directory/calibre_temp:
total 416
-rwxr-xr-x  1 user grp 202417 Jul 20 11:31 tsisram40_00.oas.fvi
drwxr-xr-x  2 user grp 4096 Jul 20 11:31 tsisram40_00.oas.lcb
-rwxr-xr-x  1 user grp 202302 Jul 20 11:31 tsisram40_01.oas.fvi
drwxr-xr-x  2 user grp 4096 Jul 20 11:31 tsisram40_01.oas.lcb
```

Note that the calibre_temp contains the .fvi files and the bin injected .lcb directories, so the MDPIndexSearchPath was also correctly followed.

Overwrite the JOBLIB Path in Micronic Job Decks

You can overwrite the JOBLIB path name specified in a Micronic job deck using the MDP_MICRONIC_JOBLIB_WD environment variable.

The basic syntax is as follows:

set ::MDP_MICRONIC_JOBLIB_WD *pathname*

This allows you to change the JOBLIB command in Calibre MDPview without having to modify your job deck.

Control Index File Generation

You can control the index generation for pattern file viewing for MEBES and VBOASIS using a number of methods.

For Calibre MDPview batch:

MDP_INDEX_MASK_FILES {ALL| NONE | *size_in_MBs*}

For Calibre MDPverify:

index_mask_files {ALL | NONE | *size_in_MBs*}

This supports MEBES, VBOASIS input of OASIS.MASK files to MDP EMBED (through extended MEBES job deck), Calibre MDPverify (through VBOASIS to input_file).

There is also a -index_mask_files invocation syntax option for Calibre MDPview that performs the same function.

- ALL — All files are indexed. This is the default.

- NONE — Suppresses index file generation for any file (MEBES and OASIS).
- *size_in_MBs* — For any pattern file that is greater in size than *size_in_MBs*, the file is indexed.

In addition, a pop-up dialog box appears asking to continue when loading a MEBES or View Only Oasis file if an index is generated. You can activate or deactivate this popup by setting **Preferences > Misc > General > Prompt before MDP index file updates** (or use the **prefs_misc_promptOnMdpIndexFileUpdate** option, which is set to "false" by default).

However, the popup only appears in an interactive session and not in batch command mode, even if the preferences are marked for regeneration. If you load the file from the command line (for example, "calibremdpv -m *filename*" or "calibremdv *filename*"), Calibre MDPview loads without prompting if you want to regenerate the file.

MEBES Index File Checksum Failure Control

When a MEBES pattern file is loaded, Calibre MDPview automatically checks if an index file exists.

If the index file exists but its checksum does not match the information in the header of the index file (due to file corruption), you can control how the viewer behaves when it encounters this scenario using the following environment variable:

::MDP_MEBES_INDEX_FILE_EXCEPTION_POLICY [2]

If set to 2, Calibre MDPview issues an error message (as the data is beyond the window boundary) and does not load the file. If not set to 2, a warning message is issued instead and the file is loaded (this is the default).

Control Size of MALY SERIAL and DATE Strings

You can control the size (in mm) of MALY job deck SERIAL and DATE text strings using a series of *wbinit.tcl* variables.

In your *wbinit.tcl* file, you can add the following environment variables:

- set :: MALY_FONT_PITCH *pitch_in_mm*

  Controls the distance between the characters for a MALY SERIAL and DATE string. If not specified, 1 mm is the default.

- set :: MALY_FONT_HEIGHT *height_in_mm*

  Controls the height of a MALY SERIAL and DATE string. If not specified, 1 mm is the default.

- set :: MALY_FONT_WIDTH *width_in_mm*

  Controls the width of a MALY SERIAL and DATE string. If not specified, 1 mm is the default.

The larger value of either ::MALY_FONT_WIDTH and ::MALY_FONT_HEIGHT should be an integer multiple of the smaller value. The tool automatically adjusts one of them to reach this limitation.

Chip Placement Outside the Substrate

You can control the behavior of error handling when the extent of any chip placement goes outside the chip substrate.

In your *wbinit.tcl* file, you can control error handling using the following environment variable:

::EJOBDECK_OUTSIDE_SUBSTRATE_SEVERITY {ERROR | WARNING | IGNORE}

where:

- ERROR — Does not allow loading of the job deck with missing chips. The tool terminates.
- WARNING — Allows loading of the job deck and issues warnings for the chips that go out of the substrate.
- IGNORE — Allows loading of the job deck without issuing warnings for the chips that go out of the substrate.

Index

Third-Party Information

Details on open source and third-party software that may be included with this product are available in the *<your_software_installation_location>/legal* directory.