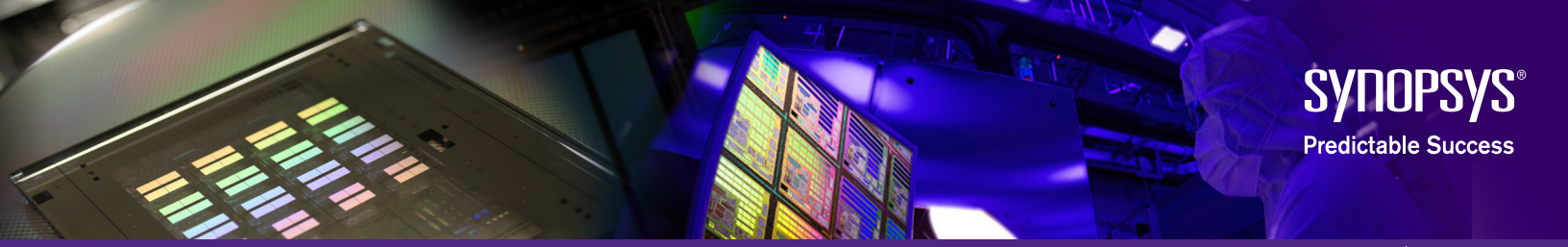


SYNOPSYS®
Predictable Success

Newsletter for data prep engineers

Issue 4 | Spring 2012

# CATS news

## In This Issue



***Dear CATS User:***

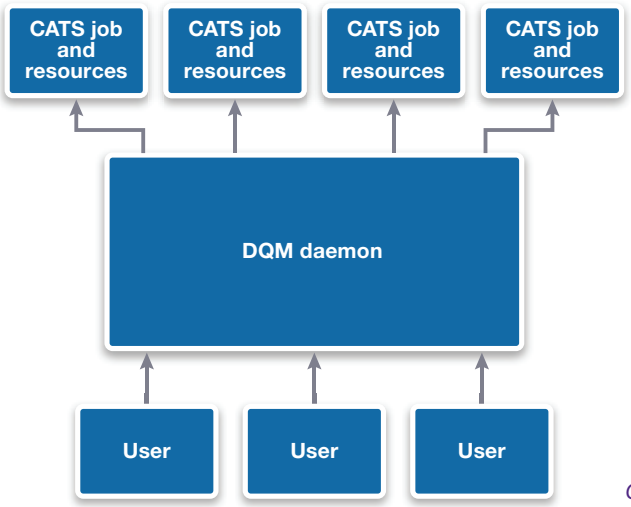
*Thank you for your continued support of CATS for your mask data preparation needs. This newsletter highlights new capabilities in CATS which help increase efficiency of your MDP flows. These new capabilities include dynamic queue management (DQM) for DP jobs, in-die registration and manufacturing rules check and correct (MRCC). We hope you find this information useful and look forward to your feedback.*

***Anjaneya Thakar***
*CATS, Product Marketing Manager*

## Dynamic Queue Management

**Bill Moore,** CATS R&D Engineer

Dynamic Queue Management (DQM) is a new system that is used to manage resources and scheduling for CATS jobs. It also allows a user to monitor the status of the jobs. The DQM configuration is shown in the following figure:




*Continued on page 2*

## CATS Support for In-die Registration Metrology

**Seurien Chou,** CATS Applications Engineer

This article discusses CATS' automated metrology search (AMS) method for discovering suitable registration marking sites inside the die area of a photomask. This methodology can be employed to study wafer overlay registration, which has been identified as one of the major challenges of next generation wafer lithography. This challenge arises due to shrinking feature size and from Double Patterning Lithography (DPL).

An example of how CATS AMS can be used to study the overlay issue is described in a 2010 SPIE paper titled *"In-depth Overlay Contribution Analysis of a Poly-layer Reticle."* Traditional photomask registration metrology focuses on standard maximum placement error in X and Y, which is derived from a population of thirty to three hundred measurement locations in the frame area of the mask. The aforementioned

*Continued on page 4*

*Dynamic Queue Management continued from page 1*

Each user (there can be an arbitrary number) communicates with the DQM Daemon. The user can specify resources, request jobs to be run, make modifications to existing jobs, and view the status of jobs or resources. User-specified jobs are added to the DQM job queue, while user-specified resources are added to the DQM resource pool. The DQM Daemon uses this user input to allocate resources to specific jobs and schedule jobs to be run based both on priority and available resources. The Daemon will also shift resources from one job to another if the user input dictates that this be done.

## Types of Jobs

DQM can be used to run two different types of CATS jobs: distributed processing (DP) and single-processor (SP). Both types can be added to the job queue via the dqm add command. An example of adding an SP job would be:

```
dqm add SP 50 /remote/catsuser/run_00.cinc
/remote/catsuser/working_directory
```

This command requests the Daemon to add an SP job at priority 50 using `/remote/catsuser/run_00.cinc` as the CATS include file and `/remote/catsuser/working_directory` as the base directory for the job. Priority is an integer, with 1 meaning highest priority and 100 meaning lowest priority. Jobs with higher priority are scheduled to be run ahead of jobs with lower priority.

Adding a DP job is similar to adding an SP job, except that the maximum number of resources must be specified. These resources are DP workers and stitchers. At least one worker must be used for a DP job. At least one stitcher must be used for a VSB12 direct output DP job; for other DP jobs the stitching happens “under the hood” and the maximum number of stitchers can be set to zero. An example of adding a DP job is:

```
dqm add DP 50 4/0 /remote/catsuser/run_00.cinc
/remote/catsuser/working_directory
```

The parameters have the same meaning as for the SP job, with the additional specification of a maximum of four DP workers and no stitchers. If an SP or DP job is added successfully, the daemon will send the user confirmation of this, along with the job ID:

```
Job added successfully
JobID: Job000001
```

The status of the job can be checked using the dqm desc command:

```
dqm desc Job000001
```

The Daemon will respond with information about the job, such as:

```
ID:              Job000001
Mode:            DP
Status:          Waiting
Priority:        50
W/S Max:         4/0
W/S Now:         0/0
Created At:      Feb 06 14:17:30
Started At:      -
Completed At:    -
Command Line:    /remote/catsuser/run_00.cinc
Base Directory:  /remote/catsuser/working_directory
Input File:      -
Input Size:      -
Input 2 File:    -
Input 2 Size:    -
Output File:     -
Output Size:     -
```

In this case, the job is waiting to be launched. As such, the start/end times and file information are not yet available. A listing of all jobs can be generated using the dqm ls command.

## Types of Resources

There are four different types of DQM resources: SP, MASTER, WORKER and STITCHER.

SP jobs require only a single SP resource. An example of adding an SP resource to the resource pool using the dqm addr command is:

```
dqm addr host1 SP
```

where `host1` is the name of the machine to be added to the pool as an SP resource.

Adding other resources is similar to adding an SP resource, except that a temporary storage directory and a launch type must also be specified. An example of adding a WORKER resource is:

```
dqm addr host2 WORKER /remote/catsuser/tmp rsh
```

where host2 will be added to the resource pool as a CATS worker. The worker will be launched using rsh. The launch type can also be ssh, lsf or sge. These last two launch types indicate that worker will be launched on a grid. The syntax is similar for adding CATS masters and stitchers. If a resource is added successfully, the daemon will report confirmation of this along with the resource ID:

```
Resource added successfully
ResID: Res000001
```

In addition to the computing resources, the user must specify how many CATS dp_start licenses are available using the `dqm numlic` command.

## Launching a Job

Once a job has been added to the job queue, the Daemon will not attempt to launch until it comes to the head of the list of waiting jobs. In order for a job to be launched, the minimum resources required for the job must be available. SP jobs require an SP resource, while DP

jobs require a Master, an available dp_start license and at least one worker. DP jobs also require at least one stitcher if they are VSB12 direct output.

## Modifying a Job: mod, starve, and maxprio

Job priority and maximum resources can be changed using the `dqm mod` command. A DP job may also be modified so that it is reduced to a single worker using the `dqm starve` command. This enables resources to be freed up for other jobs. A job may also be set to maximum priority using the `dqm maxprio` command. This will cause a waiting job to move to the head of the waiting list, and if it is a DP job, it will be given as many workers as are available, up to the maximum specified.

The ability to modify, starve and set a job to maximum priority enables user to control the priority and configuration of jobs while they are waiting and as they are running. The Daemon may also decide to starve a job if it is required to set another job to maximum priority.

## An Example

Consider a small example, where there are two DP masters and four DP workers available, but only a single CATS dp_start license. The following illustrates how these resources are added to the resource pool.

```
% dqm numlic 1
Number of licenses successfully set
%
% dqm addr host1 MASTER ./tmp RSH
Resource added successfully
ResID: Res000001
%
% dqm addr host2 MASTER ./tmp RSH
Resource added successfully
ResID: Res000002
%
% dqm addr host3 WORKER ./tmp RSH
Resource added successfully
ResID: Res000003
%
% dqm addr host4 WORKER ./tmp RSH
Resource added successfully
ResID: Res000004
%
% dqm addr host5 WORKER ./tmp RSH
Resource added successfully
ResID: Res000005
%
% dqm addr host6 WORKER ./tmp RSH
Resource added successfully
ResID: Res000006
%
```

Next, confirm that the resources are in the resource pool using `dqm lsr`.

```
% dqm lsr
ResID      Type   Grid  Status Assignee    Hostname
Res000001  Master RSH   Idle   Unassigned  host1
Res000002  Master RSH   Idle   Unassigned  host2
Res000003  Worker RSH   Idle   Unassigned  host3
Res000004  Worker RSH   Idle   Unassigned  host4
Res000005  Worker RSH   Idle   Unassigned  host5
Res000006  Worker RSH   Idle   Unassigned  host6
%
```

The next step is to add the jobs to be run. Suppose the first job is added as:

```
% dqm add DP 50 4/0 /remote/catsuser/run_00.cinc
/remote/catsuser/wd1
Job added successfully
JobID: Job000001
%
```

It is immediately launched using one of the available masters and all 4 available workers. It can be checked using `dqm desc`.

```
% dqm desc Job000001
ID:              Job000001
Mode:            DP
Status:          Running
Priority:        50
W/S Max:         4/0
W/S Now:         4/0
Created At:      Feb 06 14:43:06
Started At:      Feb 06 14:43:06
Completed At:    -
Command Line:    /remote/catsuser/run_00.cinc
Base Directory:  /remote/catsuser/wd1
Input File:      /remote/catsuser/inputs/input_00.
cref
Input Size:      77129472B (74MB)
Input 2 File:    -
Input 2 Size:    -
Output File:     /remote/catsuser/wd1/output_00.
cref
Output Size:     - (-)
Master:          Res000002
Workers:         Res000003 Res000004 Res000005
Res000006
Stitchers:
Progress:        3.78%
```

Next, a second job is added.

```
% dqm add DP 50 4/0 /remote/catsuser/run1_00.cinc
/remote/catsuser/wd2
Job added successfully
JobID: Job000002
%
% dqm desc Job000002
```

```
ID:              Job000002
Mode:            DP
Status:          Waiting
Priority:        50
W/S Max:         4/0
W/S Now:         0/0
Created At:      Feb 06 14:52:02
Started At:      -
Completed At:    -
Command Line:    /remote/catsuser/run1_00.cinc
Base Directory:  /remote/catsuser/wd2
Input File:      -
Input Size:      -
Input 2 File:    -
Input 2 Size:    -
Output File:     -
Output Size:     -
```

Because all four workers in the resource pool are being used by the first job, the job is waiting. However, suppose that the second job has been deemed very high priority. This can be set to the maximum priority as follows:

```
% dqm maxprio Job000002
Job successfully set to maximum priority
% dqm desc Job000002
ID:              Job000002
Mode:            DP
Status:          Running
Priority:        0
W/S Max:         4/0
W/S Now:         3/0
Created At:      Feb 06 14:55:54
Started At:      Feb 06 14:56:03
Completed At:    -
Command Line:    /remote/catsuser/run1_00.cinc
Base Directory:  /remote/catsuser/wd2
Input File:      /remote/catsuser/inputs/input1_00.
                 cref
Input Size:      177129472B (169MB)
Input 2 File:    -
Input 2 Size:    -
Output File:     /remote/catsuser/wd2/output1_00.cref
Output Size:     - (-)
Master:          Res000001
Workers:         Res000006 Res000005 Res000004
Stitchers:
Progress:        53.65%
```

Job000002 now has three workers that have been reassigned from Job000001. Job0000001 is in “starvation” mode with only a single worker.

```
% dqm desc Job000001
ID:              Job000001
Mode:            DP
Status:          Starving
Priority:        50
W/S Max:         4/0
W/S Now:         1/0
Created At:      Feb 06 14:55:54
Started At:      Feb 06 14:55:54
Completed At:    -
Command Line:    /remote/catsuser/run_00.cinc
Base Directory:  /remote/catsuser/wd1
Input File:      /remote/catsuser/inputs/input_00.
                 cref
Input Size:      77129472B (74MB)
Input 2 File:    -
Input 2 Size:    -
Output File:     /remote/catsuser/wd1/output_00.cref
Output Size:     - (-)
Master:          Res000002
Workers:         Res000003
Stitchers:
Progress:        45.09%
```

After the maximum priority job completes, the workers will be restored to the starving job. After it completes, the resources will be available for more jobs.

## Getting Started with DQM

This article has given a brief introduction to DQM, explaining the basic structure and usage. For more information, refer the CATS DQM How-to Guide. The `dqm help` command will provide more information, including command syntax.

---

*In-die Registration continued from page 1*

paper suggests that a more accurate picture of the overlay error can be achieved through a more complex registration scheme that yields low spatial, high spatial, and random errors instead of the simpler scheme of capturing the systematic errors (X-scale, Y-scale, and orthogonality factor), and residual errors.

The benefit of the more complex scheme is the increased visibility into local in-die registration errors that lead to the paper’s main conclusions. The complex scheme required thousands of site locations to be identified and measured that were directly enabled by CATS AMS. CATS AMS coupled with next generation registration

tools from the leading vendors allows for large scale marking of non-standard registration target sites decoupled from the scribe area. The resulting information could also be used for very accurate mask overlay and mask CD correction on critical layers via mask reprocessing tools, thus reducing mask yield loss and overall processing time.

The following figure demonstrates the inputs and the outputs to CATS AMS.

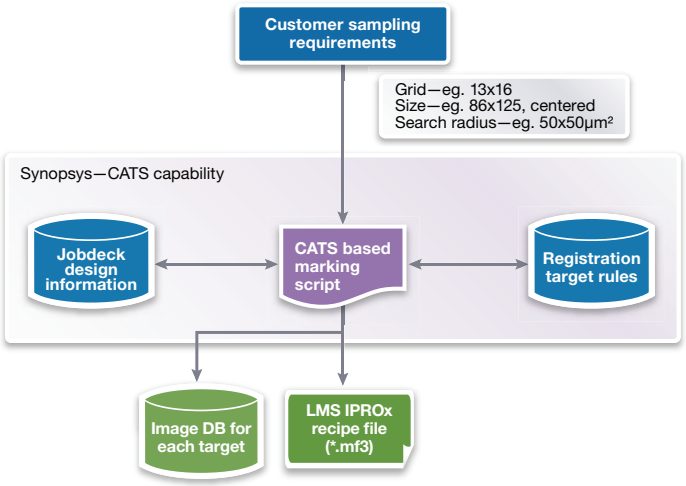



**Figure 1: AMS flow—inputs and outputs**

The automated CATS search method is run through a Perl script interface and is named indie.pl. The script also requires a library of other Perl files in order to run.

The script takes a parameter file as input. The parameter file contains runtime settings (such as the location of the library files, debugging information, and modes of operation), the input jobdeck information (the name, layer, and chip that contains the standard alignment marks), sampling parameters (the grid pitch, grid center, point search radius), and metrology tool parameters (the minimum measurable feature size, minimum nearest neighbor distance, and minimum address unit).

The script must also have access to the jobdeck file and associated pattern files. The jobdeck format currently supported is MEBES with MEBES or CFLT patterns files.

The sampling grid must be specified. Because overlay errors can be extracted from regularly spaced measurements via special algorithms, the entire mask need not to be analyzed, thus saving computation time. Six parameters are required to fully define the grid: the grid size in X and Y, the grid pitch in X and Y, and the grid center in X and Y. For example, a grid size of 13 x 16 will result in 208 target sites. The

```
[10:05][...schou/indie/demo] $ more indie.parmfile
scripts_directory /remote/us01home17/schou/indie_dev
debug_mode no
full_debug_mode no
filter yes
analyze no
mark no
random yes
batch_mode yes

jobdeck_name schou.jb
jobdeck_layer 1
qa_chip qa

grid_pitch_x 8000
grid_pitch_y 8250
grid_size_x 3
grid_size_y 3
grid_center_x 76200
grid_center_y 76200
grid_search_radius 50

address_size 0.001
gapcd_min 0.15
feature_dim_max 13
feature_aspect_ratio_max 20
polygons_max 20
```

**Figure 2: Example of a parameter file**

search radius is the area used around the grid point to search for target features. The larger the search radius, the likelihood to finding a target site increases, but the long the processing time, so a balance must be struck. The algorithm used to search about the grid point is either random or spiral walking. Multiple target features might be found so the search can be stopped after a specified number. Note that multiple grids can be run iteratively as the area of focus (meaning, the grid center and grid size) changes with each feedback cycle of error measurement much like that described in the paper previously mentioned.

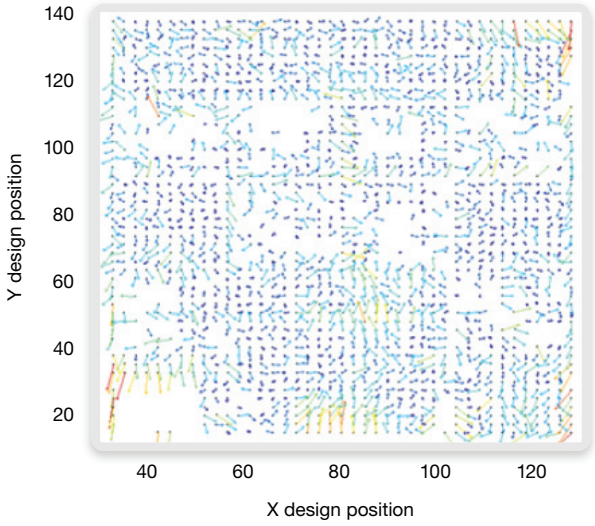



**Figure 3: Example of an output of registration map**

The output of `indie.pl` is a comma-separated value spreadsheet along with supplemental marking information such as images, polygon database files, or other tool-required files needed to make accurate pattern measurements. The script can be easily configured to convert the spreadsheet into tool-specific recipe files that typically uses a subset of the large amount of information collected for each target site. The tool recipe file could then be used to automate the marking on the registration tool, thus saving substantial search and setup times. The CSV file contains not only the standard location and dimensions of the feature, but also information about the nearest neighbor distance, local density, feature symmetry, surrounding area symmetry, information required to measure irregular shapes, and so forth. The large amount of design data collected coupled with the measurement data can be used for complex engineering analysis and characterization not easily achieved before.

The following figure shows the various steps that take place inside of CATS AMS and their relationship to the inputs and outputs. The internal steps are tightly coupled with the CATS fracture engine, requiring various features ranging from Manufacturing Rule Check, Synthesis Package, Advanced Fracturing, Distributed Processing, MEBES jobdeck editing, Advanced Marking Module, and the registration tool-specific output module.

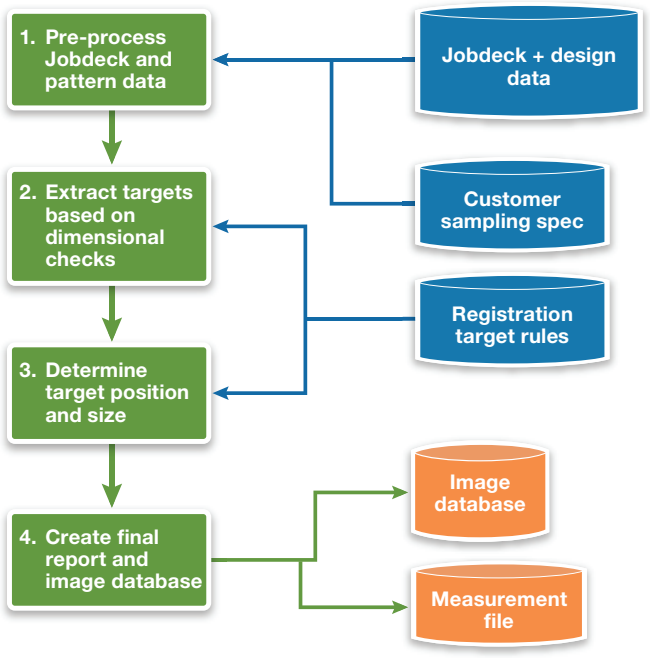



**Figure 4: AMS process flow**

This flow outlines the following steps:

1. Pre-process the jobdeck and pattern design data:
   - Make a copy of the original jobdeck only in the area of interest around the grid defined in the parameter file. Many small CFLT files are created for each grid point, and are referenced by the new jobdeck. Each CFLT is by default a 100x100 micron square. This step is required to improve the performance of subsequent steps.
   - Filter out data that touches the edge of the small CFLT since the center of these features cannot be determined. Filter out features that do not meet the minimum aspect ratio requirement.
2. Analyze and extract the potential targets based on geometric properties and dimensional checks:
   - Perform an internal dimension check of all figures. The results are all internal MRC errors.
   - Filter out features that are larger than the specified width and/or length. SELECT all large rectangles from those figures not selected by the INTERNAL MRC check. Extract data from large rectangles from all data.
   - Filter out features that are closer than the specified minimum space. Run EXTERNAL checks to look for figures larger than the specified minimum distance. Extract final filtered data with the results of MRC checks.
3. Analyze any shaped potential targets to determine the area of the largest, non-rotated rectangle that can be contained by the figure to be measured and the center of the feature. Perform symmetry calculations of the feature and surrounding areas.
4. Create the final report on the final targets along with image database for each target.

Additional capabilities that will be available in the next version of CATS AMS include an adjacent mask analysis that will extend the single layer CATS AMS to multiple layers specifically targeted to DPL that will allow both in-die overlay investigation as well as space CD uniformity investigation, which is another key challenge of next generation wafer lithography.

# CATS Version G-2012.09 MRCC-Based All-In-One Flow

**John Valadez,** CATS Applications Engineer and
**Daniel Salazar,** CATS R&D Engineer

CATS introduced new functionality in Version E-2011.03, with a radically redesigned engine, the ability to filter data based on user criteria, execute Boolean operations and expressions in-memory, and the ability to handle multi-layer mask data.

CATS Version G-2012.09 expands MRC's capabilities by adding polygon sizing, pattern generation and matching, and other data creation commands that operate on all CATS-supported mask data formats. This is why it is now also known as CATS MRCC, or Mask Rule Check and Correction.

Checking and correcting mask data means that a very large amount of data must be handled, which historically has meant complex flows and long runtimes.

Traditional data flows that employ CATS have thus far been based on executing several steps. Each step performs a very specific number of operations, such as sizing and a Boolean function, and after a step is done, the result is saved to file. Then the next step is executed.

Since each temporary file must be both written out and then read back in, and the number of steps to be executed can be high for long flows, IO overhead can become significant.

An MRCC-based flow does not need to save temporary results to file, and then read them back in, step after step. CATS MRCC now has optimized buffer handling that keeps the temporary results in memory for further processing. Intermediate results are kept only as long as they are strictly required, and are released as soon as possible, which keeps resource requirements low, and allows for streamlined processing of even long flows.

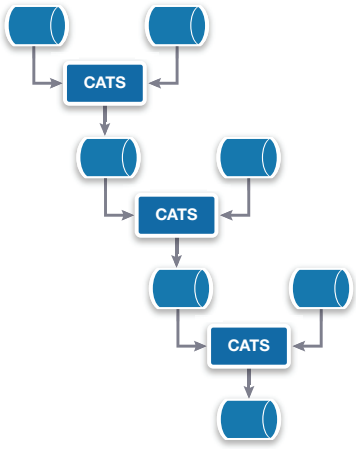



Figure 1: Traditional flow model

## A Simple Operation

A single step of a traditional CATS flow can perform several specific operations, although in a predefined structure. Combining these steps allows for greater flexibility, but the mechanism of passing the data between steps is through a file, which must be read back in by the next step to use it.

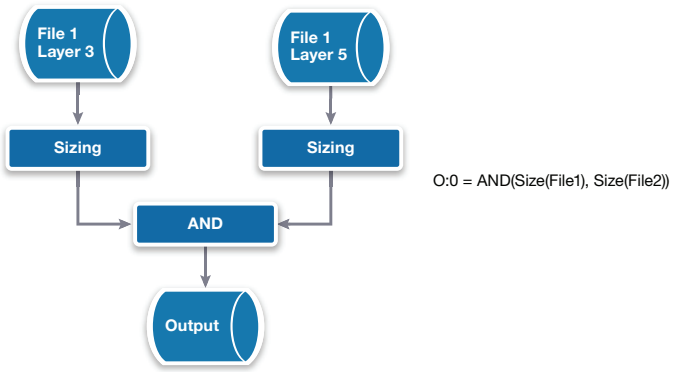



Figure 2: One step in a flow

The MRCC engine can mimic the most important CATS steps by providing the functionality for each individual operation separately. Operations can be combined with intermediate results passed from one operation to another in-memory. With MRCC, individual layers of input files can also be manipulated separately.

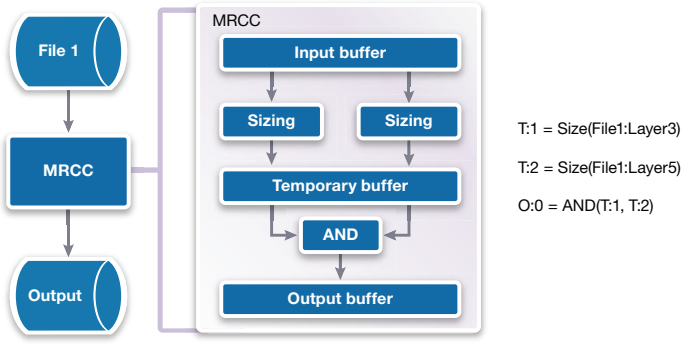



Figure 3: Equivalent MRCC-based step

With CATS MRCC, combining polygon sizing and boolean operations is simple. The commands for the above example are as follows:

```
MRC PSIZE 1 INPUT P:3
MRC PSIZE 1 TEMPORARY 1
MRC PSIZE 1 DISTANCE 0.12

MRC PSIZE 2 INPUT P:5
MRC PSIZE 2 TEMPORARY 2
MRC PSIZE 2 DISTANCE 0.12

MRC AND 3 INPUT T:1 T:2
MRC AND 3 LAYER 0
```

MRCC additionally allows the flexibility of using different sizing values for each layer to be sized, and even different sizing types. In fact, it is even possible to perform overunder sizing or underover, with the same number of commands. Assume that each sizing operation of the above example requires an underover sizing, but the sizing done on Layer 5 requires a larger undersizing, and with WEXTEND type sizing. The commands to do that would be as follows:

```
MRC PSIZE 1 INPUT P:3
MRC PSIZE 1 TEMPORARY 1
MRC PSIZE 1 DISTANCE -0.10 0.22

MRC PSIZE 2 INPUT P:5
MRC PSIZE 2 TEMPORARY 2
MRC PSIZE 2 DISTANCE -0.20 0.32
MRC PSIZE 2 TYPE WEXTEND

MRC AND 3 INPUT T:1 T:2
MRC AND 3 LAYER 0
```

## Simple Multi-Layer Boolean Operations with MRCC

User flows can become lengthy because of automation tools. Some users employ Boolean expressions of data layers in their flows. Consider the following example.

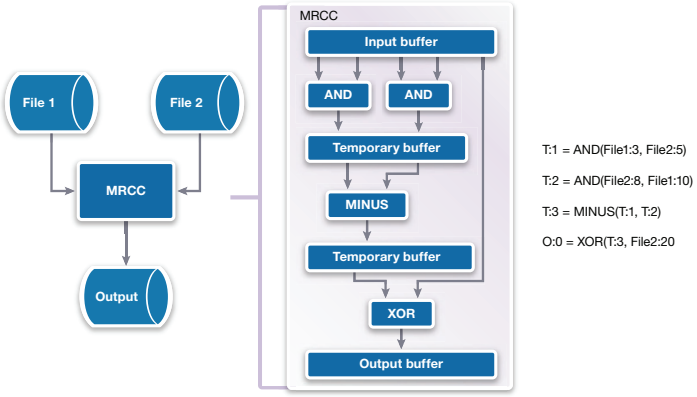



Figure 4: Multi-Layer Boolean operation example

There are four Boolean operations, which in a traditional flow would require four distinct steps, and therefore four files to be written out (three intermediate and one output). With CATS MRCC in CATS Version G-2012.09, this can be processed in just one step without any intermediate files, thus eliminating unnecessary file I/O and improving the TAT significantly. The commands are as follows, assuming that File1 is input in the PRIMARY: slot, and File2 in the SECONDARY.

```
MRC AND 1 INPUT P:3 S:5
MRC AND 1 TEMPORARY 1

MRC AND 2 INPUT S:8 P:10
MRC AND 2 TEMPORARY 2

MRC MINUS 3 INPUT T:1 T:2
MRC MINUS 3 TEMPORARY 3

MRC XOR 4 INPUT T:3 S:20
MRC XOR 4 LAYER 0
```

## Complex Operations with MRCC

The MRCC engine allows you to create combinations of mask data processing commands by chaining or pipelining as required to meet user flows.

Simple dimensional checks allow for Mask Rule Checking, and this is what CATS MRC has been used for historically. But now, combinations of Data creation commands, Boolean operations, and Dimensional checks allow CATS MRCC to do Mask Rule Checking and Correction. Sizing and Boolean operations allow data biasing. Pattern Matching and Data Select enable data searching and filtering, which in turn can be used to correct any problems in the mask data thereby improving the yield.

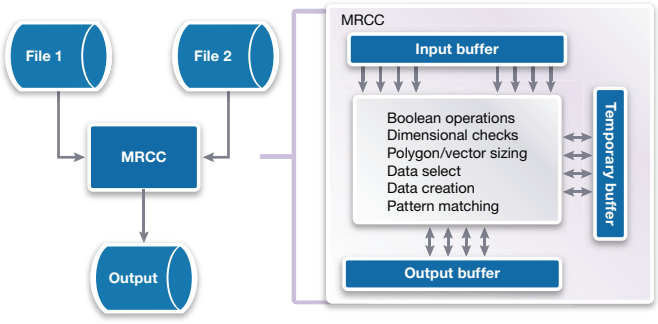



Figure 5: MRCC in CATS, and general architecture

The offered functionality in MRCC with CATS Version G-2012.09 can be summarized as follows:

| | | |
|---|---|---|
| Data Selection | 11 SELECT commands | Polygon selection, selection of parts of polygons, selection of polygon edges |
| Data Creation | 4 Polygon Features commands | |
| Sizing | Polygon single step sizing<br>Polygon dual-step sizing<br>4-way Vector sizing | Dual-step polygon sizing supports underover and overunder |
| Dimensional Checks | 5 Dimensional commands | Comprehensive spacing and corner checks<br>Large battery of control options |
| Boolean operations | 6 Boolean commands | Can be chained to process the equivalent of Boolean expressions |
| Pattern operations | 1/2/3 input pattern generation<br>1/2/3 input pattern matching | Matching supports Range patterns, and data or bbox output |

## Fracture Flow

A fracture flow combines several data layers with sizing-based biasing. In practice, that means that several Boolean operations and several sizing operations are executed in complex combinations. The result is mask data.

It has already been seen how CATS MRCC can perform a simple operation, which in a traditional flow would be just one step in a fracture flow. However, in MRCC the flow steps can be joined and merged, and all operations of all steps can be processed in just one block.

Combining steps to make up an MRCC-based fracture flow is as simple as adding more commands, and chaining them as required to pass data from one to the next via the MRCC internal buffers.

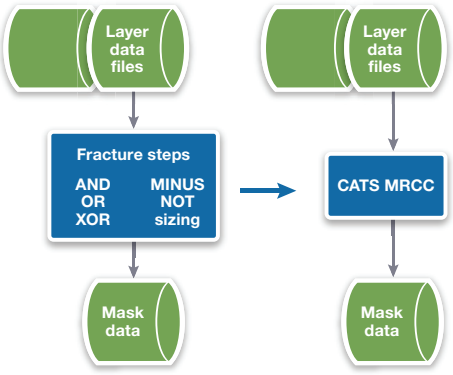



Figure 6: Fracture flow model and MRCC based equivalent

All MRCC commands have the same engine interface for specifying the input and output buffers. Therefore, chaining a Boolean command or a Sizing command is very similar. The only differences are the number of inputs, and the non-engine options, which are specific to each command, such as the sizing distances for a sizing operation.

## Mask Data Prep Flow

In a mask prep flow, CATS reads mask layout data and converts it to litho format. In the standard use model, this process must be accompanied by mask rule checking for manufacturability and verification against input data. Any dimensional violation found must be reported, and the verification against input data must be clean.

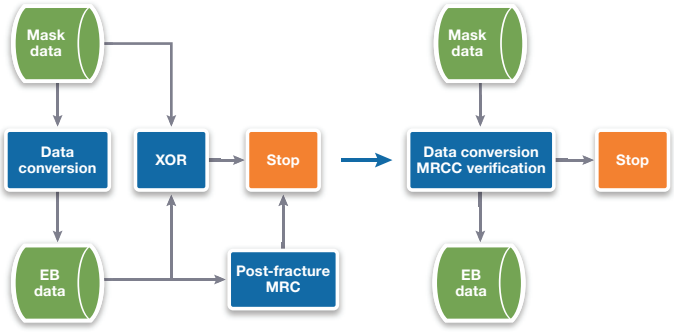



Figure 7: Standard use model, and its equivalent optimization with MRCC

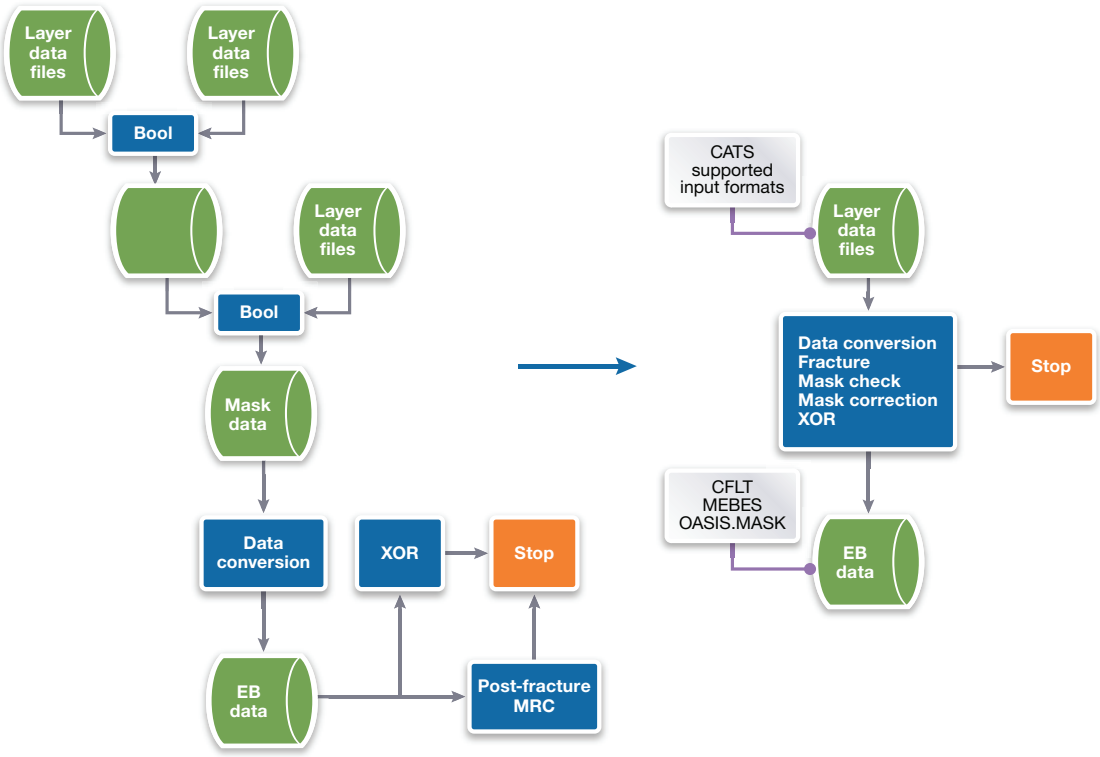



Figure 8: All-In-One flow, all verification and fracture operations can be processed in one step

CATS MRC has offered dimensional commands for manufacturability for a long time. Verification against input data essentially means an XOR Boolean operation, which CATS MRCC offers. Therefore, the flow can be optimized to employ a comprehensive verification step that is based on MRCC. As a result, the new CATS MRCC use model offers a distinct advantage, both in terms of performance and ease of use.

## All-in-One Flow

We have examined the fracture and mask data prep flows separately, and have shown how CATS MRCC can optimize each flow in the verification process. Now, let's look at joining the two flows, and see how MRCC can optimize and improve upon those as well.

The capabilities of manipulating individual layers and processing in-memory, coupled with the large suite of commands offered, provide the tools for a powerful simplification of the overall flow structure, while maintaining the flexibility of a traditional flow, which naturally results in an All-in-One flow.

## Summary

As technology advances and mask data becomes smaller and smaller, the flows become ever more complex, and performance and flexibility become very important.

Whether in fracture or verification, Boolean operations or data creation, pattern matching or data filtering, MRCC in CATS Version G-2012.09 has evolved into a powerful tool to meet increasing user demands. It is flexible enough to meet the needs of a fracture flow, or the needs of a mask prep flow. It can also combine both and provide an All-in-One flow capability. If a customized application is required for a unique situation, MRCC can meet that challenge. If DRC-like processing is required (whether for mask corrections, modifications, or standard MRC), MRCC can handle that as well. Add the various processing modes available such as MT, DP, or Hybrid mode, coupled with the capabilities that CATS has, to read a wide range of data formats, as well as the powerful format readers, and the possibilities are endless.

# Applications of MRCC

**John Valadez,** CATS Applications Engineer

A year ago, CATS introduced a new MRCC product in Version C-2011.03 which provided unique capabilities to not only do checks but also to perform corrections. In CATS Version G-2012.09, due out this summer, polygon sizing will be added to this suite of tools, giving mask data prep users a toolbox that provides basic functionality such as dimensional checks, and for more advanced users, data filtering and manipulating techniques. This article covers both situations with two related examples, and will hopefully give our users enough information to apply MRCC to solve other mask data problems whether it is lithography, inspection, or metrology.

## Example #1 – Bias mask layer data but if any spacing violations are found during process, the fracture is stopped.

Consider some of the basic choices that CATS data prep users are sometimes faced with regarding incoming data. They can choose to run dimensional checks on incoming data to locate violations before continuing the data processing, or, they can choose to run dimensional checks during data processing. There is nothing incorrect with either decision, but how would a user perform the latter? Consider a basic real world example in which an incoming mask layer's data must be checked for violations but must also be biased up for process reasons.

The steps involved include:

1. Input the data.
2. Configure the global CATS fracture options such as RESOLUTION, LIMITS, and so forth.
3. Perform CATS MRCC Polygon sizing.
4. Use CATS MRCC to locate spacing violations.
5. Generate the output.
6. Exit CATS.

A closer look at what happens in steps #2 and #3 in this example is as follows:

```
MRC PSIZE 1 INPUT P:100
MRC PSIZE 1 DISTANCE 0.005
MRC PSIZE 1 LAYER 0
!
MRC EXTERNAL1 INPUT O:0
MRC EXTERNAL1 SPACING < 1
MRC EXTERNAL1 LAYER 1
```

The first block of MRC commands accepts layer 100 of the input file, applies a positive sizing of 0.005 microns to each side, and writes data to layer 0 in the output (permanent buffer). In the second block, it accepts layer 0 of the output and uses it as input. Therefore, spacing checks looking for violations smaller than a micron are performed on biased data. If violations are found, they are placed on layer 1 of the output. Either MRC PSIZE or MRC EXTERNAL1 steps can be performed first, but the values used for spacing violation will need to be revised, depending on what data is being input.

Now consider one additional touch to this streamlined approach. Add a flag so that if any violations are located, the process is stopped completely as follows:

```
MRC OUTPUT MAX_ERRORS 1
```

This command states that a maximum of one error is allowed. When this error is found, CATS will continue processing the current block of data, but will stop immediately afterward.

## Example #2 – Bias mask layer data but if any spacing violations are found during process, make a correction.

Now consider a case where, instead of stopping once a spacing violation is located, you make a small correction to the input data. This example involves assist features (AFs) and data within proximity.

**Figure 1: 3 AF's shown, 2 vertical and 1 horizontal**

AFs are sometimes sized independently because they are typically the smallest whole features on a mask, and therefore require special attention (intended for the photomask but not on silicon). MRCC can be used to extract AFs out of a normal data set using various methods (such as with a combination of dimensional checks and data filtering).

The steps involved include:

1. Input the data.
2. Configure the global CATS fracture options such as RESOLUTION, LIMITS, and so forth.
3. Perform CATS MRCC Polygon sizing.
4. Use CATS MRCC to locate spacing violations.
5. Flag the biased edge of the violation.
6. Move the violating edge inward.
7. Specify the final Booleans.
8. Generate the output.

Here, steps #3 and #4 are the biasing and spacing checks just as in Example #1. This is displayed graphically in Figure 2, which shows AFs after sizing as well as a violation found between a corner and one of the AFs.

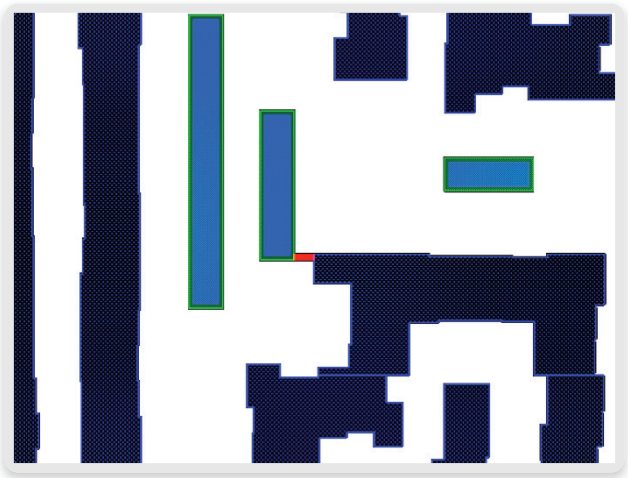


**Figure 2: Sized AF's and violation found.**

Since the violation is touching both the original input and the sized AF, the edge of the AF where the violation is touching can be flagged using:

```
MRC SELECT EDGE2 INPUT P:0 P:1
```

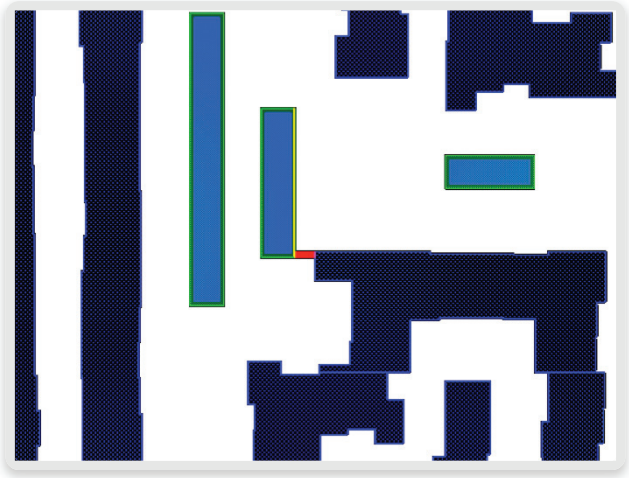


**Figure 3: Flagged right edge of AF.**

For this rule's inputs, use the output from polygon sizing (P:0) and the spacing violation itself (P:1). Figure 3 shows the right edge of the smaller vertical AF now flagged. This is step #5 in the flow.

This edge placed along the right edge of the AF is adjustable. This can be expanded, shrunk, or extended using the MRC VIZE (vector sizing) command. However, for this example, expand it inward 0.001 microns (or whatever value it must be to avoid a spacing violation). Here, step #6 would be:

```
MRC VSIZE INSIDE 0.001
```

The final Booleans will remove the expanded edge and merge everything back together to be on the same layer and to be lithography-ready. MRCC contains most Booleans relevant to mask data prep users. These include XOR, MINUS, NOT, OR, AND, as well as a COPY command:

- MRC XOR—two-layer input
- MRC MINUS—two-layer input
- MRC NOT—single-layer input
- MRC OR—two-layer input
- MRC AND—two-layer input
- MRC COPY—single-layer input

Finally, Figure 4 shows the data now sized, and the violation corrected. This is where the term *Correct* in *Check and Correct* is derived.

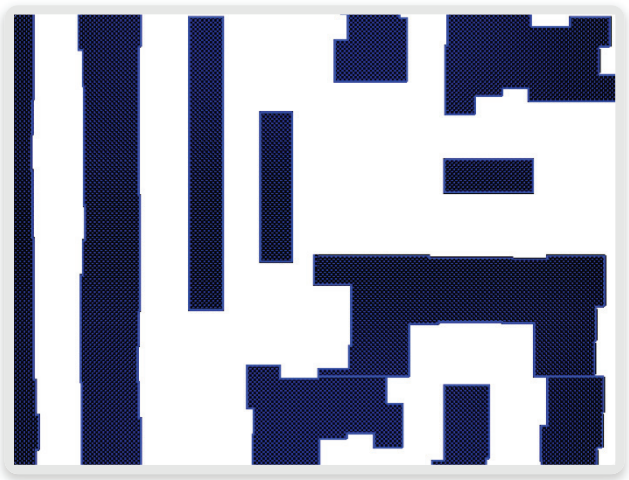


Figure 4: AF's sized and violation removed.

## Summary

Whether it is basic everyday tasks such as biasing and dimensional checking, or more advanced undertakings such as edge or line end treatment, CATS MRCC can handle the majority of today's mask data prep user needs.

Contact CATS Support at cats@synopsys.com for information and to get started on an all-in-one flow using MRCC.

# Tips for Migration from the CATS 2008.03 (V30) GUI to the 2009.03 (V31+) GUI

**Hari Konnanur,** CATS Applications Engineer

The graphical interface in CATS Versions C-2009.03 (V31) and later has a completely different form and feel compared to CATS Version A-2008.03 (V30) and earlier interfaces. The newer GUI is completely customizable with a windows-like graphical interface.

Here is a basic outline of issues to be considered when migrating to a newer version of CATS.

Create a checklist with a list of operations performed using the GUI, as to which groups are focused on what part of the specific use model. Typically this will be a one-time CAD setup or working with the CATS Support team.

The following list describes the requirement-gathering phase of what must be done:

- Whether the job role involves automation (GUI is driven on scripts).
- Whether the Job role involves Manual work (such as browsing the layout).
- Understand the pain points with the old graphical interface.
- What are the most common functions performed (customization—is it sufficient?)
- What is the most common user background base (command-line driven or click-and-GUI driven). This is a very important feature to understand as typically a user will be one or the other and at the same time it is important to *manifest* the GUI to adapt to that user in order to gain traction for adoption.

The next steps elaborate on how to migrate:

1. Understand the user's use model for what the GUI is currently being used. A detailed understanding of this can lead to specific customization that can be built for a broader user base, which will have a larger adoption appeal.
2. Revisit all the legacy startup scripts/ CINC file. An example is illustrated on one sample application migration activity.
3. Customize the GUI (**Tools > Options** command) for possible defaults that the user requests. This can be automated and placed in the `$HOME/.synopsys/cats` directory.
4. Collect all the CINC files from various locations that are being used.
   - The first goal is to migrate all the CINC files in V31+. (There would be a general tendency to simply run old CINC files in the new environment but it is better to stop, look at the use model, and use this opportunity to insert *best practices* that CATS can enforce to the user to use the tool more efficiently).

- Check for any “deprecated commands” and their impact on the data.
- If the commands are deprecated, understand whether there is an “equivalent one” in the new CATS version or if these commands are default in the CATS engine.
- Sometimes, users are not aware of the CINC files that are being used through automation. A detailed thread-based analysis of the calls must be traced down and data flow diagram must be created to look for more optimal ways to reduce the redundancy.

5. Leverage the Framework-based “TCL Interface” for bringing in the automation specific to GUI Drawing, Rendering, default coloring schemes, look and feel, and so forth.
6. The V31+ GUI leverages heavily on cache-files for faster loading of Oasis/GDSIII and Machine data formats. The details of cache files is outside the scope of this article, but it is again best practice to have all these cache files pre-generated in the user environment to significantly speed up the loading time for jobdecks and pattern files. As a tip in configuring the user interface, make certain the default is set to “.” (TED) where the cache files can be generated if they are not present. This is important because cache files by default write into $HOME/.synopsys/cats/layoutCache.
7. Review all the environment variables, as some of them could be deprecated or some of them could be added from time to time based on the version. Especially some of the TE_* variables. It would be best to run the variable setup with your CATS support.

## Case Study: Migration Activity of PCI (Process Control Image) from V30 to V32+

A user was using V30 to analyze the Process Control Images specific to the jobdecks being loaded. Think of these as special structures located at various locations that a user would need to inspect at these specific locations.

While understanding the use model, here were this user’s pain points:

1. Shell driven – Manually viewing the list of locations. The number of locations to inspect could be more than 250+ on specific job orders, and if for some reason that user could not or did not capture the disposition and clicked on the next, there was no easy way to go back to the error.
2. There was no easy method to share data with other users on the review team other than “window (coordinates).“
3. If a user had to go to Location #245, that user had no control to drive to that location but to step from #1 to #245. With the new implementation of the coordinate browser, a user is able to jump immediately to the location of interest.

While using the new GUI, one additional caveat was the use model for performing the review should not change drastically based on the V30 use model.

Below is a figure showing the V30 use model “stepping through each location.”

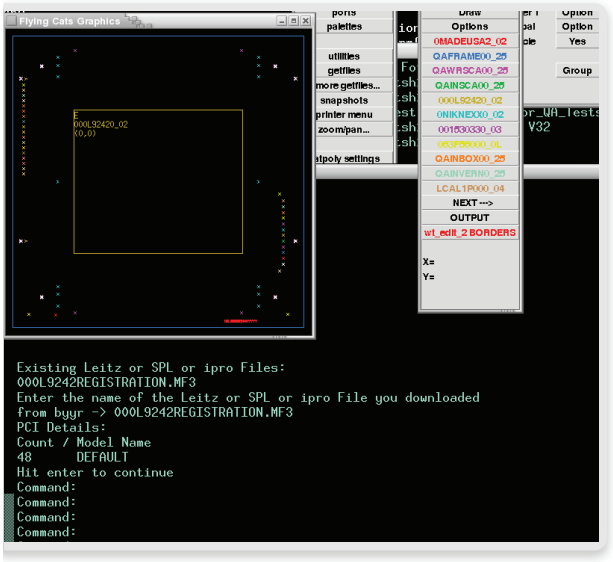



**Figure 1: V30 use model for stepping through each coordinate location**

After automation, and using the new V31+ GUI, gave the user:

1. The advantage of jumping to any predetermined location
2. The flexibility that the old use model of window (coordinate) would still work.
3. Additional options to navigate into the coordinate space.
4. The ability to “overlay” any number of jobdecks to observe the interactions.

The following figure shows the V31+ GUI use model with the coordinate browser:

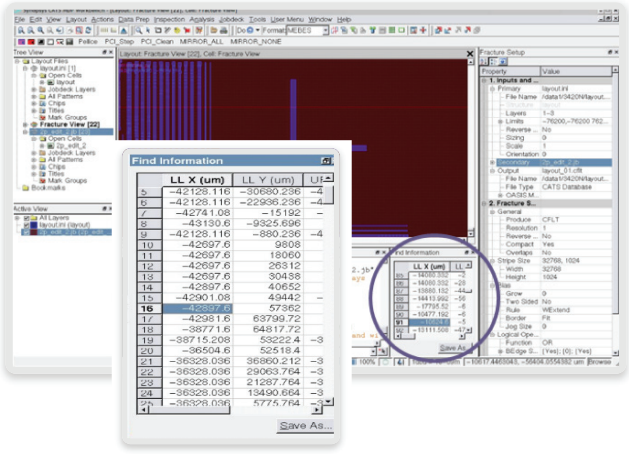



**Figure 2: V31+ GUI with coordinate browser**

# CATS Cache Files and Direct Input Improvements

**Gary Meyers,** CATS Applications Engineer

CATS Version C-2009.03 was the first release in which CATS utilized a cache file. After a cache file is created, there are performance benefits during fracture, drawing, and further re-inputs of the file. Since the initial release, numerous enhancements have been added to the cache creation process. Cache file creation for Oasis input files is now threaded, and many improvements have been made to speed up the creation of cache file for native input files (JEOL, VSB, MEBES, and so forth). Although initial cache file creation requires a small time investment, there are methods that can be implemented to minimize any TAT impacts that cache file creation might have on the overall MPD flow.

Here is a simple script that could be utilized to create the cache file at the time of order receipt:

```
#! /bin/csh -f

setenv CATS_HOME /path/to/install/E-2011.03-SP6
set path = ($CATS_HOME/scripts $path)

echo “Say CATS version used was {version}” >>
cache_create.cinc
echo “tcl default createcacheonopen on” >>
cache_create.cinc
echo “threads 8” >> cache_create.cinc
echo “input ./$1” >> cache_create.cinc
echo “say cache file for $1 created” >> cache_
create.cinc
echo “Exit” >> cache_create.cinc

cats -nodisplay -include ./cache_create.cinc >&
cache_create_$1.log &
```

with the usage being:

```
% create_cache.csh <file.oasis>
```

The THREADS command enables CATS to use threads when creating a cache file during an Oasis file input.

CATS offers a variety of options to manage your cache files. These can be found in the **Tools > Options** menu command’s **Layout > File > Cache node**.

Multiple locations can be specified for CATS to search for pre-existing cache files, and there are a number of options to delete existing cache files. For added safety, all cache file deletion options are user-specific.

Even with a cache file present, users running on a remote server might still experience some performance delays due to other issues, such as network infrastructure inefficiencies, poor network bandwidth between the CATS server and user location, and overhead or inefficiencies in the virtual application in use to connect to the CATS server.

Prior to the release of CATS Version C-2009.03, the standard suggestion was that users convert a GDS file to the CATS CLIB format using READFILE. The older CATS infrastructure was more efficient when working with the CATS internal format files. At that time, if you had the need to input, draw, or fracture the same GDS file multiple times, you could save time by first converting GDS files to CLIB files. With the new CATS infrastructure, there are optimized readers for the various types of input files supported by CATS. These readers also support the machine formatted files for NuFlare, JEOL, Micronic, and Vistec lithography tools. These optimized readers, along with cache file usage, have largely eliminated the need for any intermediate file conversion before inputting into CATS. Therefore, we now strongly suggest that you simply directly input the file as delivered to obtain the best possible performance.

**For questions about CATS please send email to cats@synopsys.com.**

SYNOPSYS®
Predictable Success
Synopsys, Inc. • 700 East Middlefield Road • Mountain View, CA 94043 • www.synopsys.com