A proposal to extend the Gerber format with nested blocks. (The goal is more efficient panel definitions. Rev 11, 2016.05)

By Karel Tavernier and Rik Breemeersch

1. Preface

Ucamco proposes extending the Gerber format to make it more efficient in handling fabrication and assembly panels. The proposed new features will no doubt have other applications.

Printed circuit boards are fabricated in panels. The PCB is repeated a number of times on a production panel. The image file representing a panel must represent all instances of the PCB. One way to represent the PCB instances is with a so-called 'flat' file: the objects representing the PCB are simply copied n times in the file, each time at the appropriate place. While this defines the correct image it blows up the file size and slows down processing the image in CAM and on the production equipment. A more efficient way is to store the PCB objects only once, and add an instruction to step and repeat the PCB over the image. The current SR command in Gerber exactly does that.

However, the assemblers, where the bare boards are populated with components, more and more works in panels themselves, often called an 'array', 'biscuit' or 'assembly panel'. The PCB fabricator then ships *arrays* to the assembler, *not single PCBs*. What he repeats in his bare-board production panel are the arrays. The efficient way to represent this image is by a nested step and repeat: the single PCB is stepped into an array, and the array is stepped into the production panel. With a nested step and repeat the PCB data is only once in a file. The problem with the SR in Gerber is that it supports only one level, no nesting. Thus one has to flatten either the array or the working panel. The resulting big files can become a problem when a small but complex piece of electronics such as a smartphone is fabricated.

To address this issue Ucamco will extend the Gerber language with nested step and repeat. Tests performed together with Via Mechanics in Japan indeed demonstrated dramatic productivity increases. To introduce this new feature in an easier and safer way Ucamco suggest not extend the capability of the existing SR command but to introduce new command, mainly AB. The reason is that legacy Gerber readers that do not yet support nested step and repeat, might overlook subtle changes in the SR command and produce the wrong image, without warning. A new command is safer. Indeed, the conformance section in the Gerber format specification states: "To prepare for future extensions of the format, Gerber file readers must give a warning when encountering an unknown command...". When testing a file with the new command on the well-respected GC-Prevue Gerber viewer an error message duly appeared.

Another issue is that the step and repeat only allows a regular array. To allow more general repeats Ucamco introduces block aperture that that can be flashed in any location and orientation. The new AB command creates a block aperture. The now aperture options set by LM, LR and LS allow to mirror/rotate/scale the block apertures, and all other apertures for that matter.

The new Part attributes value defines unequivocally whether we deal with a single PCB, an array or a fabrication panel.

Nested step and repeat and block apertures will make Gerber more efficient with panelized data. Furthermore, the block aperture is a powerful general construct with no doubt many other applications. Together, they are a powerful extension of the Gerber format.

Ucamco publishes a draft specification and sample file on its website to allow the Gerber user community to review and comment on the new feature before it is cast in concrete. Please send your comments and criticism to gerber@ucamco.com.

Karel Tavernier,
Managing Director,
Ucamco

2. Blocks Overview

A *block* is a sub-stream of graphics objects that can be can be added to the graphics objects stream. Blocks can be mirrored, rotated, scaled, shifted and polarity toggled when adding them to the stream. By using blocks sub-images occurring multiple times must only defined once, thus slashing file size and boosting image processing speed. The information that these sub-images are identical is preserved.

Note that a block is *not* a macro of commands called repeatedly in the command stream. The command stream is processed sequentially, without procedure or macro calls. Gerber is not a programming language. The commands defining the block are processed once, creating the blocks which later commands can append to the object stream.

Blocks can contain objects with different polarities (LPD and LPC). Blocks can overlap.

Once a block is added to the graphics objects stream its objects becomes part of the overall stream. The effect of the objects does not depend on whether they were part of a block or not. Only their order is important. A clear object in a block clears *all* objects beneath it, including objects *not* contained in the block.

There are two commands dedicated to blocks: SR and AB. They open and close block statements, a sequence of commands that define blocks. A block statement can contain other block statements.

⚠ **Warning:** It is recommended to avoid overlapping blocks containing both clear and dark polarity objects. The order in which they are added to the object stream may affect the final image. The order is always correctly implemented in Gerber readers. When all objects in block are dark or the blocks do not overlap the order does not affect the image. This is safe to use.

3. SR - Simple Step and Repeat X1

The SR block allows a array of blocks to be generated. SR blocks cannot contain other SR or AB blocks and are generated using the SR command.

The syntax for the SR command is:

<SR command> = SR[X<Repeats>Y<Repeats>I<Step>J<Step>]*

Syntax	Comments
X<Repeats>	<Repeats> defines the number of times the block is repeated along the X axis. It is an integer ≧ 1.
Y<Repeats>	<Repeats> defines the number of times the block is repeated along the Y axis. It is an integer ≧ 1.
I<Step>	<Step> defines the step distance along the X axis. It is a decimal number ≧ 0, expressed in the unit set by the MO command.
J<Step>	<Step> defines the step distance along the Y axis. It is a decimal number ≧ 0, expressed in the unit set by the MO command.

A new SR command closes the current SR command. Used without any parameters (or with both repeats equal to 1 it simply closes the opening SR command. Used with a specification of a number of repeats and step distance it ends the current SR command and immediately starts a new one. In other words an SR command always terminates the current SR block.

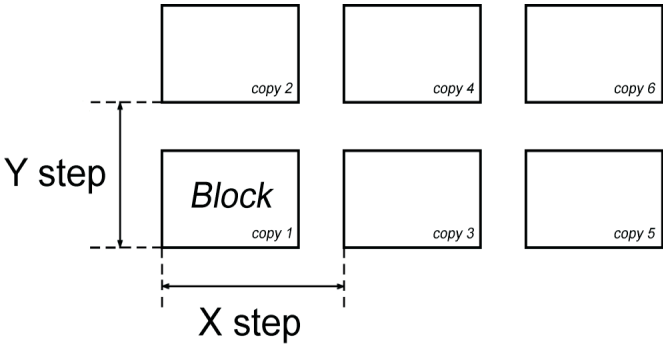
An opened SR block must be closed before the end-of-file command (M02) is encountered.

The number of repeats and the steps can be different in X and Y. The number of repeats along an axis can be 1, which is equivalent to no repeat. If the repeat number is 1 it is recommended to set its step value to 0.

The SR commands step-repeats (copies) blocks in the image plane when closed according to the parameters in the opening SR command. Each copy of the block contains identical graphics object. Blocks are copied first in the Y and then in the X direction.

1.1. Examples

```
%SRX3Y2I5.0J4.0*%
...
%SR*%
```



Step and Repeat (SR blocks)

Other examples:

Syntax	Comments
%SRX2Y3I2.0J3.0*%	Opens an SR block that is repeated 2 times along the X axis and 3 times along the Y axis. The step distance between X-axis repeats is 2.0 units. The step distance between Y-axis repeats is 3.0 units.
%SRX4Y1I5.0J0*%	Opens an SR block that is repeated 4 times along the X axis with the step distance of 5.0 units. The step distance in the J modifier is ignored because no repeats along the Y axis are specified.
%SRX1Y1I0J0*%	Close the previously started SR block. If no block is opened, this does nothing.
%SR	Close the previously started SR block. If no block is opened, this does nothing.

4. Block apertures NEW

4.1 Overview of block apertures

The AB command creates *block apertures*. The command stream between the opening and closing AB command defines a block aperture which is then stored in the aperture dictionary. Thus the AB commands add an aperture to the dictionary without needing a D-code as for a standard or macro aperture. The effect of block apertures is governed by the LM, LR, LS and LP commands as any other aperture. When a block aperture is flashed the transformed - mirrored, rotated and scaled - objects are added to the graphics object stream.

While a standard or macro aperture always adds a *single* graphics object to the stream, a block aperture can add *any number* of objects, with different polarities. A block aperture is not a single graphics object but a set of objects. Standard and macro apertures always have a single polarity while block apertures can contain both dark and clear parts.

Do not use blocks - or macros - when a standard aperture is available. Standard apertures are built-in and therefore are processed faster.

AB statements can contain other AB or SR statements. See 5.

The first purpose of block apertures is to repeat a sub-image without copying the generating commands. It is a much more powerful concept than the SR command. The SR only allows repeats on a regular grid without mirror, rotate or scale, without nesting. Aperture blocks can be repeated at *any* location and *individually* mirrored, rotated and scaled. In PCB fabrication blocks are used to create panels without duplicating the data. Clear (LPC) objects are used to mask out underlying copper balancing patterns in the panel.

The second purpose of block apertures is to complement macro apertures. Blocks are simpler to create. However, macros but macros can have parameters and blocks cannot. Macro outline primitives support linear segments only while blocks can contain contours with both linear and circular segments. A block aperture consisting of a single region creates a single object with one polarity - as with standard or macro apertures. Thus single object apertures of any shape can easily be created. In PCB design to fabrication data transfer block apertures can define pads. Such block apertures ideally consist of a single object (region). However, multi-object single polarity blocks can have a use. Pads are sometimes painted or stroked; such jobs are very hard to handle in CAM as pad locations must be reverse-engineered. Defining a block aperture with the painting of a single pad and then flashing it as the pad locations is a big step forward as the pad locations are

now clear. Such a usage may be an intermediate step towards flashing pads with proper single object apertures.

4.2 AB - Aperture Block command NEW

The syntax for the AB command is:

<AB command> = AB[block D-code]*

Syntax	Comments
AB	AB for Aperture Block
<template name>	Name of the block aperture template. This name must be unique and cannot be used by another template.

Examples:

Syntax	Comments
%ABD12*%	Opens of the definition of aperture D12
%AB*%	Closes the current block aperture definition.

As with any other aperture, the flash operation updates the current point but otherwise leaves the graphics state unmodified. (The graphics state is *not* set to the value it had after the block statement defining the block. A block is not a macro command but simply a set of graphics objects.)

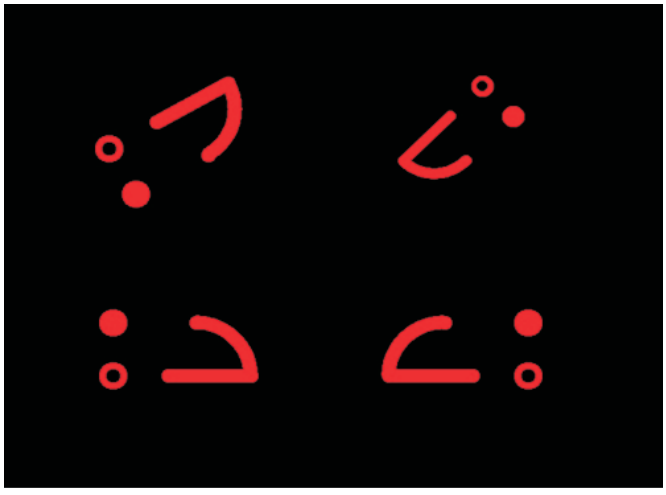
The section between the opening and closing AB commands can contain nested AB commands. The resulting apertures are stored in the library and are available subsequently over the file, also outside the enclosing AB section.

If the polarity is clear (LPC) when the block aperture is flashed the polarity of all objects in the block is toggled (clear becomes dark, and dark becomes clear). This toggling proceeds through all nesting levels. If the polarity is dark (LPD) then the block aperture is inserted as is. In the following example the polarity of objects in the flash of block D12 will be toggled.

```
%ABD12*%
...
%AB*%
....
D12*
%LPC*%
X-2500000Y-1000000D03*
```

4.3 Example

```
G04 Ucamco copyright*
%TF.GenerationSoftware,Ucamco,UcamX,2016.04-160425*%
%TF.CreationDate,2016-04-25T00:00;00+01:00*%
%FSLAX26Y26*%
%MOMM*%
%ADD10C,1*%
%LPD*%
%ABD12*%
%ADD11C,0.5*%
D10*
G1*
X-2500000Y-1000000D03*
Y1000000D03*
%LPC*%
D11*
X-2500000Y-1000000D03*
%LPD*%
X-500000Y-1000000D02*
X2500000D01*
G75*
G3*
X500000Y1000000I-2000000J0D01*
G74*
G1*
%AB*%
D12*
X0Y0D03*
%LMX*%
X10000000D03*
%LMY*%
%LR30.0*%
X0Y8000000D03*
%LMXY*%
%LR45.0*%
%LS0.8*%
X10000000D03*
M02*
```



5. Syntax of SR and AB nesting

We first informally define three variables:

<single command> = all commands except the block commands SR and AB
<sr parameters> = X<Repeats>Y<Repeats>I<Step> J<Step>
<D-code> = D<integer = 10>

Now we are ready to specify the block commands:

<AB open> = %AB<D-code>*%
<AB close> = %AB*%
<AB statement> = <AB open><section><AB close>

<SR set> = %SR<sr parameters>*%
<SR close> = %SR*%
<SR statement> = <SR set><section>{<SR set><section>}<SR close>

<block statement> = <AB statement>|<SR statement>

<section> = {<single command>}|{<single command>}<block statement>{<single command>}

6. D04 Command: Stepped Flash NEW

The D04 commands steps (copies) the current aperture in the image plane according to the parameters in the command.

The syntax for the D04 operation is the following:

<D04 operation> = [X<Number>][Y<Number>]M<Repeats>N<Repeats>I<Step>J<Step>D04*

Syntax	Comments
X<Number>	Coordinate data defining the X coordinate of the aperture origin of the first flash. If missing then the previous X coordinate is used. <Number> is a coordinate number - see section **錯誤！找不到參照來源。**
Y<Number>	Coordinate data defining the Y coordinate of the aperture origin of the first flash. If missing then the previous Y coordinate is used. <Number> is a coordinate number - see section **錯誤！找不到參照來源。**
M<Repeats>	<Repeats> defines the number of times the block is repeated along the X axis. It is an integer ≧ 1.
N<Repeats>	<Repeats> defines the number of times the block is repeated along the Y axis. It is an integer ≧ 1.
I<Step>	<Step> defines the step distance along the X axis. It is a decimal number ≧ 0, expressed in the unit set by the MO command.
J<Step>	<Step> defines the step distance along the Y axis. It is a decimal number ≧ 0, expressed in the unit set by the MO command.
D04	Stepped flash operation code

⚠ **Warning:** D04 operation is not allowed in a region definition.

Example:

```
X12345Y67890M2N3I5000J4000D04*
```

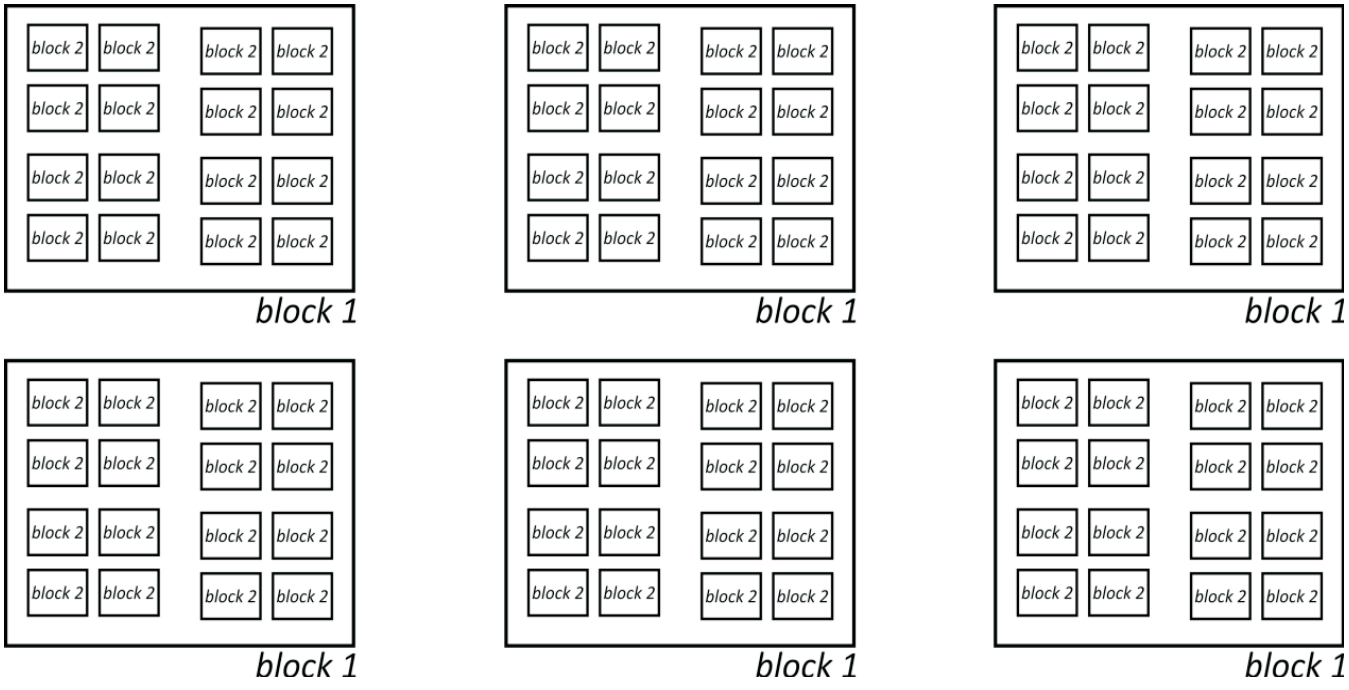
Apertures are copied first in the Y and then in the X direction. The D04 is the shorthand for a number of successive D03's. After the D04 command is finished the current point is set at the point of the last flash.

The number of repeats and the steps can be different in x and y. The number of repeats along an axis can be 1, which is equivalent to no repeat. If the repeat number is 1 it is recommended to set its step value to 0.

Example 1: a schematic example

The image below can be generated as follows:

```
G04 Create Block 2, the basic building block, as ape 100
%ABD100*%
... Gerber commands generating the block 2 image, includes border
%AB*%
...
G04 Create an intermediate block as ape 101, a 2x2 repeat starting at the block origin
D100*
ABD101*%
X0Y0M2N2I1.5J1D04*%
%AB*%
...
G04 Create Block 2 as ape 102; a 2x2 repeat of 101 plus border, add border starting at the
origin to create a block 1, ape 102
ABD102*%
D101*
X0Y0M2N2I.5J3D04*%
... Gerber commands creating the border
%AB*%
...
G04 Repeat ape 102 2x3 times
D0102*
X0Y0M3N2I10.0J7.0D04*
```

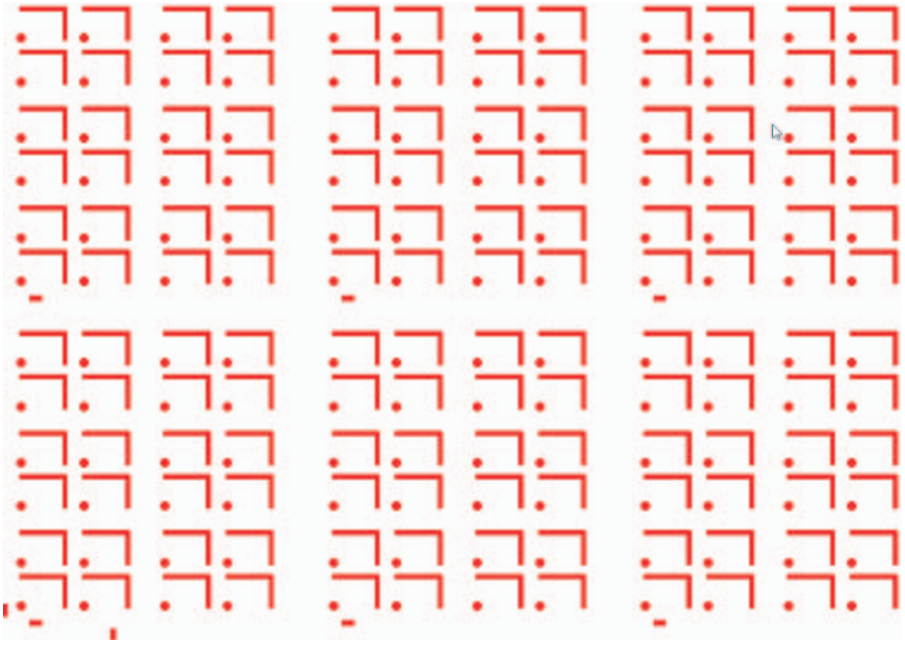


Nested Step and Repeat: schematic example

Example 2: a Gerber file using D04

The following code illustrates how blocks can be used in a real Gerber file:

```
%FSLAX36Y36*%                          G04 Set precision and zero omission*
%MOMM*%                                G04 Set unit*
%ADD10C,7.500000*%                     G04 Define apertures*
%ADD11C,15*%
%ADD12R,20X10*%
%ADD13R,10X20*%
%LPD*%                                 G04 Set layer polarity*
%ABD100*%                              G04 Define elementary block aperture 100
D10*                                   G04 Select aperture 10*
X65532000Y17605375D02*                 G04 Move*
Y65865375D01*                          G04 Draw with aperture 10*
X-3556000D01*                          G04 Draw with aperture 10*
D11*                                   G04 Select aperture 11
X-3556000Y17605375D03*                 G04 Flash with aperture 11 inside block 100*
%AB*%                                  G04 Close definition of block aperture 100*
D13*                                   G04 Select aperture 13*
X-30000000Y10000000D03*                G04 Flash aperture 13 outside of blocks*
%ABD101%*                              G04 Define block 101, 2x2 flashes of 100
D100*                                  G04 Select aperture 100
X-30000000Y10000000M2N2I100J70D04*%    G04 Flash it 2x2 times
%AB*%                                  G04 Close block 101*
%ABD102%*                              G04 Define block 102, 2x3 flashes of 101 and one of 12
D101*                                  G04 Select aperture 101
X-30000000Y10000000M2N3I230J160D04*    G04 Flash it 2x3 times
D12*                                   G04 Select aperture 12
X19500000Y-10000000D03*                G04 Flash aperture 12 inside the block definition*
%AB*%                                  G04 Close definition of block aperture 102*
D102*                                  G04 Select aperture 102
X-30000000Y10000000N32I500J520D04*%    G04 Flash it 3x2 times
D13*                                   G04 Select aperture 13
X143000000Y-30000000D03*               G04 Flash aperture 13 outside the block repeats*
M02*                                   G04 End of file
```



Nested blocks: real Gerber code

7. Aperture State Parameters

7.1 Overview

The commands LP, LM, LR and LS load the aperture graphics state parameters. These parameters transform the current aperture as used in an operation command and take effect immediately after they are loaded and remain in effect until a new value is loaded. Selecting a new current aperture does not reset these parameters, nor does any other command except LP, LM, LR and LS.

The parameters do not change the aperture definition in the aperture dictionary; in this sense they are volatile; when a new current aperture is selected the original definition is taken, and the current aperture states are applied.

An example D123 is a rectangle. It is first flashed in the original orientation. Then an LR command sets the current rotation at 90 degrees and it is flashed again. Both flashes have D code 123 and inherit the attributes of D123 but have different rotations. D123 remains unchanged. Attributes are attached to the D code and are fixed for that D code. Aperture states are attached to the current aperture and are volatile.

These transformations affect the apertures only, not the coordinate data.

The defaults are defined in the graphics state table:

Graphics state parameter	Value range	Fixed or changeable	Initial value
Unit parameters			
Coordinate format	See FS command in xxx	Fixed	Undefined
Unit	Inch or mm - See MO command in xxx	Fixed	Undefined
Drafting parameters			
Current point	Point in plane	Changeable	Undefined
Interpolation mode	Linear, clockwise circular, counterclockwise circular See G01/G02/G03 commands in and xxx	Changeable	Undefined
Quadrant mode	Single-, multi-quadrant See G74/G75 commands in xxx	Changeable	Undefined
Aperture parameters			
Current aperture	See the AD command in xxx	Changeable	Undefined
Current polarity	See the LP command in xxx	Changeable	Dark
Current mirroring	See LM command in xxx	Changeable	No mirror
Current rotation	See LR command in xxx	Changeable	No rotation
Current scaling	See LS command in xxx	Changeable	No scaling (=1.0)
Region parameter			
Region mode	On/Off - See G36, G37 commands in **錯誤! 找不到參照來源。**	Changeable	Off

7.2 Load Polarity (LP)

The LP command sets the *current polarity* graphics state parameter. This command can be used multiple times in a file. The current polarity applies to all subsequent apertures used until changed by another LP command.

Polarity can be either *dark* or *clear*. Its effect is explained in xxx. Section xxx gives an example of its use.

The syntax for the LP command is:

<LP command> = LP(C|D)*

Syntax	Comments
LP	LP for Load Polarity
C|D	C - clear polarity D - dark polarity

7.3 Load Mirroring (LM) NEW

The LM command sets the *current mirroring* graphics state parameter. This command can be used multiple times in a file. The mirroring applies to all subsequent apertures used until changed by another LM command.

The current mirroring defines the mirroring axis. The current aperture is mirrored around its *origin* (which may not be its geometric center). Mirroring is performed on the original aperture as defined in the aperture dictionary, it is not cumulative.

The syntax for the LM command is:

<LM command> = LM(N|X|Y|XY)*

Syntax	Comments
LM	LM for Load Mirroring
N|X|Y|XY	N - No mirroring X - Mirroring *along* the X axis; mirror left to right; the signs of the x coordinates are inverted Y - Mirroring *along* the Y axis; mirror top to bottom the signs of the y coordinates are inverted XY - Mirroring *along* both axes; mirror left to right and top to bottom; the signs of the x and y coordinates are inverted

Mirroring is performed before the rotation.

7.4 Load Rotation (LR) NEW

The LR command sets the *current rotation* graphics state parameter. This command can be used multiple times in a file. The current mirroring applies to all subsequent apertures used until changed by another LR command.

The rotation defines the rotation angle. The current aperture is rotated around its *origin* (which may not be its geometric center). Rotation is performed on the original aperture as defined in the aperture dictionary, it is not cumulative.

The syntax for the LR command is:

<LR command> = LR<Rotation>*

Syntax	Comments
LR	LR for Load Rotation
<Rotation>	The rotation angle is specified by a decimal, in degrees.

Mirroring is performed before the rotation.

7.5 Load Scaling (LS) NEW

The LS command sets the *current scaling* graphics state parameter. This command can be used multiple times in a file. The current scaling applies to all subsequent apertures used until changed by another LS command.

The scaling defines the scale factor applied apertures. The current aperture is scaled from its *origin* (which may not be its geometric center); in other words the origin remains in the same position, whatever the scaling.

The syntax for the LS command is:

<LS command> = LS<Scale>*

Syntax	Comments
LS	LS for Load Scaling
<Scale>	The scale factor is specified by a decimal, in degrees.

7.6 Examples

Syntax	Comments
%LPD*%	Sets the current polarity to dark
%LPC*%	Sets the current polarity to clear
%LMX*%	Sets current mirroring to mirroring along the X axis
%LMN*%	Sets current mirroring to no mirroring
%LR45.0*%	Sets current rotation to 45 degrees counterclockwise
%LR-90*%	Sets current rotation to 90 degrees clockwise
%LS0.8*%	Sets current scaling to 80%

8. New Attributes

Aperture attributes can be associated with block apertures as with any other apertures: a block aperture takes on the attributes in the attribute dictionary at the time of the %AD defining it, as with other attributes.

The %SR implicitly defines an unnamed block aperture. This unnamed block aperture takes on the attributes in the aperture attribute library at the time of the %SR.

8.1 The .AperFunction value Part NEW

A block aperture will often be used to represent a part or stage in the overall PCB production such as a single PCB of an array. It serves the same purpose and follow the values for a block as .Part attribute for the file as a whole. It takes the same values and semantics.

.AperFunction value	Usage
Part,Single	Single PCB.
Part,Array	A.k.a. customer panel, assembly panel, shipping panel, biscuit.
Part,FabricationPanel	A.k.a. Working panel, production panel.
Part,Coupon	A test coupon.
Part,Other,<mandatory info>	None of the above. The mandatory info string informally indicates the part.

Part aperture function values

Example:

```
%TA.AperFunction,Part,Array*%
```

A file describing a full fabrication panel typically has the following structure:

- A block describing the single PCB is created with aperture function Part,Single.
- Then a block is describing the customer panel is created. It contains flashes or a step&repeat of the single PCB blocks, plus tooling holes, fiducials. Its function is Part,Array.
- Finally the fabrication panel is created. It contains flashes or a step&repeat of the customer panel blocks, a border, background pattern etc. The whole file takes the .Part value .FabricationPanel.

Such a structure clearly identifies the parts and avoids data repetition.

8.2 The .FlashText aperture attribute NEW

Gerber intentionally does not contain fonts or typographic text this would introduce a complexity out of proportion to its benefits. The image of any text can be represented with the available graphic constructs, especially by contours. However, loses the information which text string they represent; this is sometimes a disadvantage.

The .FlashText aperture attribute defines this lost information. Bar codes are handled as text - one can view a barcode as a special font. .FlashText assumes is designed for text image created with a flash, typically with a block aperture.

Syntax and semantic of the attribute value is as follows:

<Text>,(B|C)[,<Optional info>[,<Top,Bot,Lft,Rgt>]]

.AperFunction Value	Usage
<text>	The text string represented by the aperture image.
(B|C)	Indicates if the text is represented by a *barcode* B -or by *characters* - C.
Optional info	Any extra information one wants to add. Font and characters sizes may be specified informally here.
Top,Bot,Lft,Rgt	The top, bottom, left and right coordinate of the text box relative to the origin of the block. The values are decimals in the units of the MO command.

The text must be readible when the aperture is not mirrored or rotated.

An empty field means that the corresponding meta-data is not specified.

Examples:

```
%TA.FlashText,Comp side,C,Courier size 10,2.5,-2.5,-100,+100*%
```

Text: Comp side

B|C: Characters,

Info: Courier size 10

Box: Ymax = +2.5, Ymin = -2.5, Xmin = -100, Xmax = +100

```
%TA.FlashText,XZ12ADF,B,Code128*%
```

Text: XZ12ADF

B|C: Carcode

Info: Code128

Box: Not specified

9. Revisions

Rev 1 Initial version, SN command only. This draft was developed with the assistance of Thomas Weyn

Rev 2 Simplified SN syntax as suggested by Remco Poelstra

Rev 3 Added AB command

Rev 4 Added .Blockpart attribute as suggested by Filip Vermeire
Rewrite/reorganize overviews
Clarified order of mirror/rotate as suggested by Remco Poelstra

Rev 5 Clarified aperture attributes and %SN and %SR
Corrected errors in examples indicated by Helmut Mendritzki
Added preface

Rev 6 Added text attribute and made clarifications as suggested by Bruce McKibben. Cleaned text and formatting.

Rev 7 Corrections after input from Helmut Mendritzki. Add barcode/character field to AperText after discussion with Bruce McKibben. BlockPart becomes part of the .Aperfunction value Part rather than a new attribute see 8.1.

Rev 8 Added words on proper use of the block aperture triggered by comments from Paul Wells-Edwards - see 4.1.
Specified the effect of flashing a block aperture under clear polarity - see 4.2
Replace name .AperText by the more specific .FlashText to avoid future name clashes and clarified the convention for "unspecified" in .FlashText - see 8.2

Rev 9 Added EBNF syntax of mixing and nesting of AB, SN and SR. Copy-edits.

Rev 10 Handling of mirror/rotate/scale was complete overhauled based on a suggestion by Masao Miyashita. Blocks are now defined directly as apertures rather than as templates with mirror/rotate/scale parameters. The new commands LM, LR and LS hander mirror/rotate/scale for all apertures. This is more powerful, more consistent with the LP command and more according to the Gerber style. The .FlashText attribute was adapted accordingly.
Specify the effect of flashing a block aperture on the graphics state; this was triggered by Mark Sims. Add image of AB example.

Rev 11 Retract the new SN statement - the set of AB/LM/LR/LS commands is now so general and powerful that SN is not needed anymore. Add stepped flash D04 - some application benefit from 'knowing' that flashes are one a grid; D04 applies to all apertures rather than SN blocks only; it can for example also be applied to define component footprints.

10. Copyright

The information contained herein is subject to change without prior notice. Revisions may be issued from time to time. This document supersedes all previous versions. Users of the Gerber Format®, especially software developers, must consult www.ucamco.com to determine whether any changes have been made.

Ucamco developed the Gerber Format®. The Gerber Format®, this document and all intellectual property contained in it are solely owned by Ucamco. Gerber Format® is a Ucamco registered trade mark. By publishing this document Ucamco does not grant a license to the intellectual property contained in it. Ucamco encourages users to apply for a license to develop Gerber Format® based software.

By using this document, developing software interfaces based on this format or using the name Gerber Format®, users agree not to (i) rename the Gerber Format®; (ii) associate the Gerber Format® with data that does not conform to the Gerber file format specification; (iii) develop derivative versions, modifications or extensions without prior written approval by Ucamco; (iv) make alternative interpretations of the data; (v) communicate that the Gerber Format® is not owned by Ucamco or owned by anyone other than Ucamco. Developers of software interfaces based on this format specification commit to make all reasonable efforts to comply with the latest specification.

The material, information and instructions are provided AS IS without warranty of any kind. There are no warranties granted or extended by this document. Ucamco does not warrant, guarantee or make any representations regarding the use, or the results of the use of the information contained herein. Ucamco shall not be liable for any direct, indirect, consequential or incidental damages arising out of the use or inability to use the information contained herein. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Ucamco. All product names cited are trademarks or registered trademarks of their respective owners.