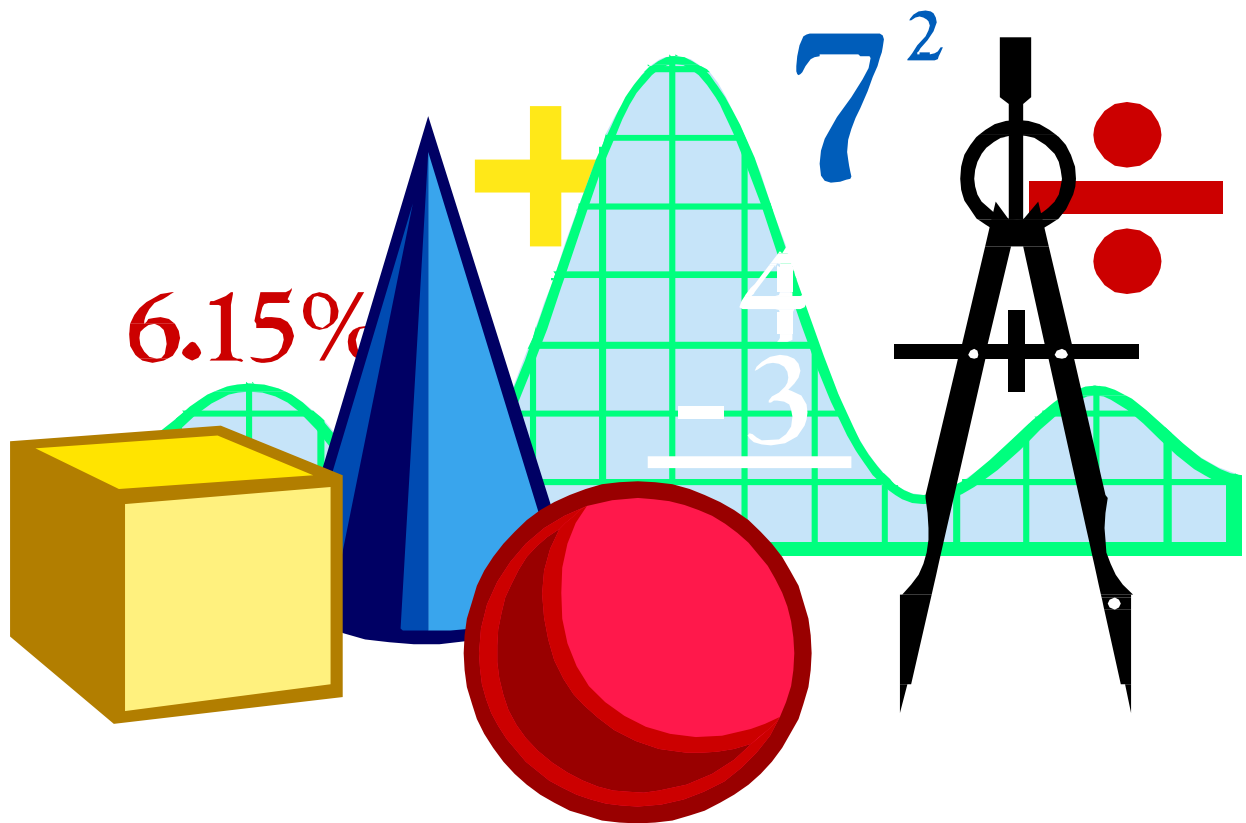


# The Gerber File Format Specification

---

A format developed by Ucamco

April 2013  
Revision I2



# Contents

---

<b>Contents .....</b>	<b>i</b>
<b>Figures.....</b>	<b>v</b>
<b>Tables.....</b>	<b>vi</b>
<b>Preface.....</b>	<b>vii</b>
<b>Acknowledgement .....</b>	<b>viii</b>
<b>1 Introduction .....</b>	<b>9</b>
1.1 Info, Questions & Feedback .....	9
1.2 What is New in Revision I1 .....	9
1.3 What is New in Revision I2 .....	10
1.4 About this Document .....	10
1.5 Scope .....	10
1.5.1 Who Should Use This Specification? .....	10
1.5.2 Conformance.....	10
1.5.3 Formatting and Syntax Rules .....	11
1.5.4 References.....	11
1.6 History of the Gerber File Format .....	11
1.7 About Ucamco.....	12
1.8 Intellectual Property and Trade Name .....	12
<b>2 Overview .....</b>	<b>13</b>
2.1 File Structure .....	13
2.2 Image Generation.....	13
2.2.1 Graphics Objects.....	13
2.2.2 Dark and Clear Polarity .....	14
2.2.3 Operation Codes .....	14
2.2.4 Stroking.....	15
2.2.5 Graphics State.....	16
2.3 Annotated Example Files .....	17
2.3.1 Example 1 .....	17
2.3.2 Example 2 .....	18
<b>3 Syntax .....</b>	<b>23</b>
3.1 Character Set .....	23
3.2 Names.....	23

3.3 Data Blocks .....	23
3.4 Statements .....	24
3.5 Data Types .....	25
3.5.1 Function Codes .....	25
3.5.2 Coordinate Data Blocks .....	26
3.5.3 Parameters .....	27
<b>4 Function Codes .....</b>	<b>29</b>
4.1 Syntax .....	30
4.2 Linear Interpolation .....	32
4.2.1 Data Block Format .....	32
4.3 Circular Interpolation .....	33
4.3.1 Overview .....	33
4.3.2 Definition of arcs .....	34
4.3.3 Numerical instability in multi quadrant (G75) arcs .....	34
4.3.4 Single Quadrant Mode .....	35
4.3.5 Multi Quadrant Mode .....	38
4.3.6 Example .....	39
4.3.7 Using G74 or G75 can result in completely different image .....	40
4.4 Regions (G36/G37) .....	40
4.4.1 Overview .....	40
4.4.2 Simple contour example .....	41
4.4.3 Using levels to create holes .....	43
4.4.4 Cut-in example 1 .....	46
4.4.5 Cut-in example 2 – good practice .....	48
4.4.6 Cut-in example 3 – poor practice .....	51
<b>5 Parameters .....</b>	<b>55</b>
5.1 FS – Format Specification .....	55
5.1.1 Coordinate Format .....	56
5.1.2 Zero Omission .....	56
5.1.3 Absolute or Incremental Notation .....	57
5.1.4 Data Block Format .....	57
5.1.5 Examples .....	57
5.2 MO – Mode .....	58
5.2.1 Data Block Format .....	58
5.2.2 Examples .....	58
5.3 IP – Image Polarity .....	58
5.3.1 Positive image polarity .....	58
5.3.2 Negative image polarity .....	58
5.3.3 Data Block Format .....	59
5.3.4 Examples .....	59
5.4 IN - Image Name .....	59
5.4.1 Data Block Format .....	59
5.4.2 Examples .....	60

5.5 AD - Aperture Definition.....	60
5.5.1 Syntax Rules .....	60
5.5.2 Data Block Format.....	61
5.5.3 Aperture Definition with Standard Apertures .....	61
5.5.4 Examples .....	67
5.6 AM - Aperture Macro .....	67
5.6.1 Data Block Format.....	67
5.6.2 Primitives.....	69
5.6.3 Parameter Contents .....	78
5.6.4 Syntax Rules .....	78
5.6.5 Examples .....	82
5.7 SR – Step and Repeat.....	85
5.7.1 Data Block Format.....	85
5.7.2 Examples .....	86
5.8 LP – Level Polarity.....	86
5.8.1 Data Block Format.....	86
5.8.2 Examples .....	87
5.9 LN – Level Name.....	87
5.9.1 Data Block Format.....	87
5.9.2 Examples .....	87
 <b>6 Reported Semantic Errors .....</b>	 <b>89</b>
 <b>7 Poor/Good Practice.....</b>	 <b>91</b>
 <b>8 Glossary.....</b>	 <b>93</b>
 <b>9 Miscellaneous Deprecated Elements.....</b>	 <b>95</b>
9.1 Coordinate Data Blocks without Operation Code .....	95
9.2 Open Contours in Regions .....	95
 <b>10 Deprecated Function Codes .....</b>	 <b>97</b>
 <b>11 Deprecated Parameters .....</b>	 <b>99</b>
11.1 Deprecated Graphics State Variables .....	99
11.2 AS – Axis Select.....	100
11.2.1 Data Block Format.....	100
11.2.2 Examples .....	100
11.3 IR – Image Rotation .....	100
11.3.1 Data Block Format.....	100
11.3.2 Examples .....	101
11.4 MI – Mirror Image.....	101
11.4.1 Data Block Format.....	101

11.4.2	Examples .....	102
11.5	OF - Offset .....	102
11.5.1	Data Block Format.....	102
11.5.2	Examples .....	103
11.6	SF – Scale Factor .....	103
11.6.1	Data Block Format.....	103
11.6.2	Examples .....	103
<b>12</b>	<b>Deprecated RS-274-D or Standard Gerber.....</b>	<b>105</b>
12.1	Standard Gerber must not be used .....	105
12.2	Origin and purpose of Standard Gerber .....	105
12.3	Standard Gerber is a NC format.....	106
12.4	Standard Gerber is not a standard .....	107
12.5	A fallacy .....	107

# Figures

Linear interpolation using rectangle aperture: example 1 .....	15
Linear interpolation using rectangle aperture: example 2 .....	15
Example 1: two square boxes .....	17
Example 2: various shapes .....	18
Gerber file structure .....	24
Single quadrant mode .....	36
Single quadrant mode example: arcs and draws .....	37
Single quadrant mode example: resulting image .....	37
Multi quadrant mode example: resulting image .....	39
Simple contour example: draws .....	42
Simple contour example: resulting image .....	42
Resulting image: first level only .....	44
Resulting image: first and second levels .....	45
Resulting image: first, second and third levels .....	45
Resulting image: all four levels .....	46
Cut-in example 1: arcs and draws .....	47
Cut-in example 1: resulting image .....	47
Cut-in example 2: good practice .....	49
Cut-in example 2: resulting image .....	50
Cut-in example 3: poor practice .....	52
Cut-in example 2: resulting image .....	53
Circles with different holes .....	62
Rectangles with different holes .....	63
Obrounds with different holes .....	64
Polygons with different holes .....	66
Circle primitive .....	70
Line (vector) primitive .....	71
Line (center) primitive .....	72
Line (lower left) primitive .....	73
Outline primitive .....	74
Polygon primitive .....	75
Moiré primitive .....	76
Thermal primitive .....	77
Rotated triangle .....	84

# Tables

---

Document conventions .....	11
Current codes .....	29
Quadrant modes .....	33
Current parameters.....	55
Arithmetic operators.....	79
Reported Semantic Errors.....	89
Poor/good practices .....	91
Deprecated codes.....	97
Deprecated parameters .....	99
Deprecated graphics state variables .....	99

# Preface

---

The Gerber file format is the de facto standard for printed circuit board (PCB) image data transfer. Every PCB design system outputs Gerber files and every PCB front-end engineering system inputs them. Implementations are thoroughly field-tested and debugged. Its widespread availability allows PCB professionals to exchange image drill and route securely and efficiently. It has been called “the backbone of the electronics manufacturing industry”.

The Gerber file format is simple, compact and unequivocal. It describes an image with very high precision, up to 1 nm. It is complete: one single file describes one single image. It is portable and easy to debug by its use of printable 7-bit ASCII characters. A well-constructed Gerber file precisely defines the PCB image and the functions of the different image elements.

Unfortunately, some applications generate invalid or poorly constructed Gerber files. Especially troublesome is the use of painting or stroking to create pads and copper areas. Poorly constructed files take longer to process, require more manual work and increase the risk of errors. Such problems are sometimes incorrectly blamed on the Gerber file format itself.

These problems may result from misunderstanding the specification or the capabilities of the format. With more than 25 years of experience in CAM software we at Ucamco know which areas are most often misunderstood. We developed successive revisions of the format specification to clarify these areas and recommend proper constructions. Our aim is to make Gerber files safer and more efficient, making fabrication more reliable, faster and cheaper.

The current Gerber file format is RS-274X or Extended Gerber. Standard Gerber or RS-274-D is deprecated. Standard Gerber does not have a single advantage over Extended Gerber but has many disadvantages. It is an NC format constrained by the technology from the 1960s and 1970s. It is simply not suited any more for reliable automatic CAD to CAM data transfer. Do not use it any longer.

Although other data transfer formats have come into the market, they have not displaced the Gerber file format. The reason is simple. Most of the problems in data transfer are due not to limitations in the Gerber file format but to poor practices. To quote a PCB manufacturer: “If we would only receive proper Gerber files, it would be a perfect world.” The new formats are more complex and less transparent to the user. New implementations inevitably have bugs. Common poor practices in more complex formats make matters worse, not better. The industry has not adopted new formats. Gerber remains the standard.

The emergence of Gerber as a standard for image exchange is the result of efforts by many individuals who developed outstanding software for Gerber files. Without their dedication the widespread acceptance of a de-facto standard could not have been achieved. Ucamco thanks these dedicated individuals.

Karel Tavernier  
Managing Director,  
Ucamco



# Acknowledgement

---

This revision of the specification was developed by

Karel Tavernier

Rik Breemeersch

advised by

Ludek Brukner

Artem Kostyukovich

Jiri Martinek

Adam Newington

Denis Morin

Karel Langhout

We thank anyone who has helped us with questions, remarks or suggestions - they are too many to mention by name. However, we wish to explicitly thank Paul Wells-Edwards who contributed substantially with insightful comments.

# 1 Introduction

---

## 1.1 Info, Questions & Feedback

Correspondence regarding this publication or questions about the Gerber File Format can be mailed to [gerber@ucamco.com](mailto:gerber@ucamco.com)

or sent to

Ucamco NV  
Bijenstraat 19,  
B-9051 Gent,  
Belgium

For more information see [www.ucamco.com](http://www.ucamco.com)

## 1.2 What is New in Revision I1

**General.** The entire specification has been reviewed for clarity. Existing warnings and notes were clarified and new ones added. The quality of the text and the drawings has been improved.

**Deprecated elements.** Format elements that are rarely used and superfluous or prone to misunderstanding have been deprecated. They are grouped together in the second part of this document. The first part contains the current format, which is clean and frugal. *We urge all creators of Gerber files no longer to use deprecated elements of the format.*

**Graphics state and operation codes.** The underlying concept of the *graphics state* and operation codes is now explicitly described. See section 2.2.3 and 2.2.2. *We urge all providers of Gerber software to review their implementation in the light of these sections.*

**Defaults.** In previous revisions the definitions of the default values for the modes were scattered throughout the text, or were sometimes omitted. All default values are now unequivocally specified in an easy-to-read table. See 2.2.2. *We urge all providers of Gerber software to review their handling of defaults.*

**Rotation of macro primitives.** The rotation center of macro primitives is was clarified. See 5.6.2. *We urge providers of Gerber software to review their handling of the rotation of macro primitives.*

**G36/G37.** The whole section is now much more specific. An example was added to illustrate how to use of polarities to make holes in areas, a method superior to cut-ins. See 4.4. *We urge all providers of Gerber software to review their handling of G36/G37 and to use layers to create holes in areas rather than using cut-ins.*

**Coordinate data blocks.** Coordinate data without D01/D02/D03 in the same data block create some confusion. It therefore has been deprecated. See 3.5.2. *We urge all providers of Gerber software to review their output of coordinate data in this light.*

**Maximum aperture number (D-code).** In previous revisions the maximum aperture number was 999. This was insufficient for current needs and numerous files in the market use higher aperture numbers. We have therefore increased the limit to the largest number that fits in a signed 32 bit integer.

**Standard Gerber.** We now define Standard Gerber in relation to the current Gerber file format. Standard Gerber is deprecated because it has many disadvantages and not a single advantage. *We urge all users of Gerber software not to use Standard Gerber.*

**Incremental coordinates.** These have been deprecated. Incremental coordinates lead to rounding errors. *Do not use incremental coordinates.*

**Name change: area and contour instead of polygon.** Previous revisions contained an object called a polygon. As well as creating confusion between this object and a polygon aperture, the term is also a misnomer as the object can also contain arcs. These objects remain unchanged but are now called areas, defined by their contours. This does not alter the Gerber files.

**Name change: level instead of layer.** Previous revisions of the specification contained a construct called a layer. As these were often confused with PCB layers they have been renamed as levels. This does not alter the Gerber files.

## 1.3 What is New in Revision I2

The semantics of “Exposure on/off” in macro apertures and holes in standard macro is sometimes incorrectly implemented. It is now more explicit in the specification. The errors were added to the ‘Reported Semantic Errors’ section. Readers and writers of Gerber files are urged to review their implementation in this light.

## 1.4 About this Document

## 1.5 Scope

This specification describes a digital format for representing a bi-level image. The specification is intended for the developers of the software that reads and writes Gerber files as well as for user of such software.

The document does not specify the following:

- ☐ Specific physical methods of storing Gerber files
- ☐ User interface or implementation details of rendering Gerber files
- ☐ Designs of the software being created for handling Gerber files
- ☐ Required computer hardware or operating system

### 1.5.1 Who Should Use This Specification?

This specification is intended for:

- ☐ PCB designers preparing Gerber files
- ☐ PCB fabricators creating or using Gerber files
- ☐ Developers of software applications using Gerber files

The specification widely uses the terminology of PCB CAM industry because it is tightly related to the semantics and interpretation of the Gerber file format. It is assumed that a reader of this document has the basic knowledge about computer-aided manufacturing (CAM) and printed circuit boards (PCB).

### 1.5.2 Conformance

A Gerber file must follow all requirements of the specification. However, there is no obligation to use any feature other than those explicitly needed.

If a Gerber file violates any requirements of the specification or contains invalid parts then the file is wholly invalid. An invalid Gerber file is meaningless and does not represent an image. A Gerber file reader may produce an image for an invalid file, as a diagnostic help, or to help the user by guessing what the intended image may be, or because the invalidity was not detected, but this image is never the representation of the file; and as such it is neither right nor wrong.

If the interpretation of a construct is not specified and there is more than one possible interpretation then the construct is invalid and must not be used.

Some elements of the format are deprecated. Gerber writers (creators of Gerber files) must no longer generate them. However such elements may still be present in legacy files. It is up to a Gerber reader (consumer of Gerber files) to implement their handling or not.

### 1.5.3 Formatting and Syntax Rules





The following font formatting rules are used in this specification:

- ❑ Examples of Gerber file content are written with monospaced font, e.g. `X0Y0D02*`
- ❑ Syntax rules are written with bold font, e.g. **<Elements set>: {<Elements>}**

The syntax rules are described using the following conventions:

- ❑ Optional items enclosed in square brackets, e.g. [**<Optional element>**]
- ❑ Items repeating zero or more times are enclosed in braces, e.g.
- ❑ **<Elements set>: <Element>{<Element>}**
- ❑ Alternative choices are separated by the '|' character, e.g.
- ❑ **<Option A>|<Option B>**
- ❑ Grouped items are enclosed in regular parentheses, e.g. **(A|B)(C|D)**

The following conventions are used:

 <b>Note:</b>	Provides essential extra information.
 <b>Tip:</b>	Provides useful extra information.
 <b>Example:</b>	Contains examples of file syntax and semantics.
 <b>Warning:</b>	Contains an important warning.

*Document conventions*

### 1.5.4 References

*American National Standard for Information Systems — Coded Character Sets — 7-Bit  
American National Standard Code for Information Interchange (7-Bit ASCII), ANSI X3.4-1986*

Bible, Mark 7:35

## 1.6 History of the Gerber File Format

The Gerber file format derives its name from the former Gerber Systems Corp., a leading supplier of photoplotters in its time.

Originally, Gerber used a subset of the EIA RS-274-D format as standard input format for its photoplotters. This subset became known as Standard Gerber. Photoplotters were NC machines at that time. Standard Gerber is not an image description format, but an NC format to drive mechanical machine tools, where aperture shapes were physical apertures in a so-called aperture wheel.

In subsequent years, Gerber extended the input format for its range of PCB devices and it actually became a family of capable image description formats. In 1997 the formats were pulled

together and standardized by the publication of the first version of this document under the name Extended Gerber Format or RS-274X. It has become the de-facto standard for PCB image data. It is sometimes called “the backbone of the electronics industry”.

In 1998 Gerber Systems Corp. was taken over by Barco and incorporated in its PCB division Barco ETS, now Ucamco. Several revisions of the specification were published over the years, clarifying it and adapting it to current needs. The obsolete Standard Gerber or RS-274-D format was deprecated.

## 1.7 About Ucamco

Ucamco (former Barco ETS) is a market leader in PCB CAM software and imaging systems. We have more than 25 years of continuous experience developing and supporting leading-edge front-end tooling solutions for the global PCB industry. We help fabricators world-wide raise yields, increase factory productivity, and cut enterprise risks and costs.

Today we have more than 1000 laser photoplotters and 5000 CAM systems installed around the world with local support in every major market. Our customers include the leading PCB fabricators across the global spectrum. Many of them have been with us for more than 20 years.

Key to this success has been our uncompromising pursuit of engineering excellence in all our products. For 25 years our product goals have been best-in-class performance, long-term reliability, and continuous development to keep each user at the cutting-edge of his chosen technology.

For more information see [www.ucamco.com](http://www.ucamco.com).

## 1.8 Intellectual Property and Trade Name

© Copyright Ucamco NV, Gent, Belgium

All rights reserved. This material, information and instructions for use contained herein are the property of Ucamco. The material, information and instructions are provided on an AS IS basis without warranty of any kind. There are no warranties granted or extended by this document. Furthermore 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.

The information contained herein is subject to change without prior notice. Revisions may be issued from time to time to advise of changes and/or additions.

No part of this document may be reproduced, stored in a data base or retrieval system, or published, in any form or in any way, electronically, mechanically, by print, photo print, microfilm or any other means without prior written permission from Ucamco.

This document supersedes all previous versions.

All product names cited are trademarks or registered trademarks of their respective owners.

Ucamco developed the Gerber file format and improves it from time to time with updates. The Gerber file format is Ucamco intellectual property. No derivative versions, modifications or extensions can be made without prior written approval by Ucamco. Developers of Gerber software must make all reasonable efforts to comply with the latest specification.

Gerber Format is an Ucamco trade name. Users of Gerber file format will not rename it, associate it with data that does not conform to the format or modify the graphical interpretation of the format.

## 2 Overview

### 2.1 File Structure

The Gerber file format is a vector format: the image is defined by resolution-independent graphics objects.

A single Gerber file specifies a single bi-level image.

A Gerber file is complete: it does not need external files or parameters to be interpreted. One Gerber file represents one image. One image needs only one file.

A Gerber file is a stream of *statements*. A statement can contain *function codes*, *parameters* and *coordinate data*.

The stream of statements generates a stream of graphics object which combined produce the final image.

A Gerber file can be processed in a single pass. This imposes a certain sequence in the statements. For example, *the coordinate format* and *unit* must be known to be able to convert coordinate data to coordinates. They are set by the FS and MO parameters. The FS and MO parameters must therefore be before the first coordinate.

Each file must end with the end of file data block 'M02\*'.



#### Example

```
G04 Set coordinate format and units in the file header*  
%FSLAX25Y25*%  
%MOIN*%  
G04 From here coordinate data can be interpreted*  
...  
M02*
```

### 2.2 Image Generation

#### 2.2.1 Graphics Objects

A Gerber file creates an ordered stream of graphics objects. A graphics object has an image (shape, size), a position in the plane and a polarity (dark or clear).

There are four types of graphics objects:

- ❑ A *draw* is a straight line segment with a given thickness and either round or square line endings.
- ❑ An *arc* is circular arc segment with a given thickness, always with round endings.
- ❑ A *flash* is a replication of an aperture at a given location. An aperture is a basic geometric shape defined earlier in the file. Apertures are typically flashed many times.
- ❑ A *region* is an area of defined by its contour. A contour is constructed with of linear and circular segments.

In PCB copper layers, draws and arcs are typically used to create tracks, flashes to create pads and regions to create copper areas.

## 2.2.2 Dark and Clear Polarity

The final image of the Gerber file is created by superimposing the objects in order of the stream. Objects can overlap. A dark object darkens (marks, paints, exposes) its image in the plane. A clear object clears (unmarks, rubs, erases, scratches) its image in all the lower levels. In other words, after superposing a clear object, its image is clear, whatever objects were there before. Subsequent dark objects may again darken the cleared area.

A Gerber file consists of a sequence of levels. From a syntactic point of view a level is a set of consecutive statements. From the image generation point of view a level is a consecutive block of graphics object with the same polarity.

The order of the objects within a level does not affect the final image. The order of the levels, however, typically affects the final image. A Gerber file can be viewed as a sequence of levels that are superimposed in the order of appearance in the file.

The LP parameter starts a new level and sets its polarity, see 5.7.

## 2.2.3 Operation Codes

D01, D02 and D03 are called the *operation codes*. The operation codes create the graphics objects by operating on a coordinate pair.

Syntactically a coordinate data block contains the coordinate data followed its operation code. A coordinate data block must contain a single (1) operation code: each operation code is associated with a single coordinate pair and vice versa. (Coordinate data blocks without operation codes are deprecated.)



### Example:

```
X100Y100D01*  
X200Y200D02*  
X300Y-400D03*
```

The operation codes have the following effect.

- ❑ D01 plots a straight line or an arc segment from the current point to the coordinate, also called a lights-on move
- ❑ D02 moves the current point to the coordinate, also called a lights-off move
- ❑ D03 creates a flash object by replicating the current aperture at the coordinate

The operation code D03 directly creates a flash object. Sequences of D01 and D02 create segments that are turned in graphics by object one of two following methods:

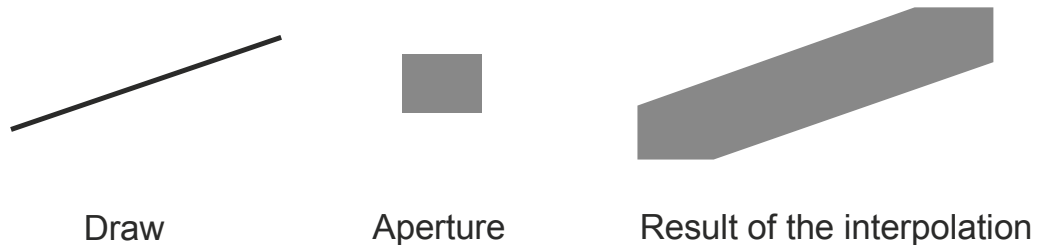
- ❑ Stroking. The segments are stroked with the current aperture, see 2.2.4.
- ❑ Region building. The segments form contour that defines a region, see 4.4.

The region mode setting determines which object generating method is used. When region mode is *off* stroking is used, when region mode is *on* region building is used.

The operation codes are controlled by the graphics state, see 2.2.5.

## 2.2.4 Stroking

A *draw object* is created by stroking a straight line segment with a solid circle or solid rectangle standard aperture. If stroked with a circle aperture the draw has round endings and its thickness is equal to the diameter of the circle. The effect of stroking a line segment with a rectangle aperture is illustrated below:



*Linear interpolation using rectangle aperture: example 1*

If the rectangle aperture is aligned with the draw the result is a draw with a straight line ending:



*Linear interpolation using rectangle aperture: example 2*



**Note:** The rectangle is *not* automatically rotated to align with the draw.

An *arc object* is created by stroking an arc segment with a solid circle standard aperture. The arc has round endings and its thickness is equal to the diameter of the circle. An arc segment cannot be stroked with a rectangle.

The only apertures allowed for stroking are the solid circle and the solid rectangle *standard* apertures (line segments only for the rectangle). Other standard apertures or special apertures, whatever their final shape, cannot be used for stroking.

Zero size apertures can be used for stroking. They create graphic objects without image, which can be used to transfer non-image information, e.g. reference points.



**Note:** Zero-length draws and arcs are allowed. The resulting image is the same as the flashed aperture. However, the graphic object is a draw or arc, not a flash.



**Note:** Any valid aperture can be flashed.



## 2.2.5 Graphics State

A Gerber file defines a graphics state after each statement.

The operation codes are controlled by the graphics state, see 2.2.3.

Except for the *current point*, all graphics state variables are set by codes or parameters. They are modal, which means that their value does not change until explicitly set by a code or parameter.

The value range of the current point is the points in the plane. The current point is not set explicitly by a code or a parameter, but implicitly by the coordinate data blocks. After a coordinate data block is processed the current point is set to the coordinate in that block.

The table below lists the graphics state variables. The column 'Fixed or changeable' indicates whether a variable value is fixed during the processing of a file or whether it can be changed. The column 'Value at the beginning of a file' describes the default value at the beginning of each file; if it is undefined it must be set before it is first needed.

Graphics state variable	Value range	Fixed or changeable	At the beginning of the file
Coordinate format	See FS parameter	Fixed	Undefined
Unit	Inch or mm See MO parameter.	Fixed	Undefined
Image polarity	POS, NEG See IP parameter	Fixed	Positive
Step & Repeat	See SR parameter	Changeable	1,1,-
Level polarity	Dark, Clear See LP parameter.	Changeable	Dark
Region mode	On/Off. See 4.4.	Changeable	No
Current aperture	Standard or macro aperture. See AD and AM parameters.	Changeable	Undefined
Quadrant mode	Single-, Multi-Quadrant See G74, G75	Changeable	Undefined
Interpolation mode	See G01, G02, G03	Changeable	Undefined
Current point	Point in plane	Changeable	(0,0)

Graphics state variables



**Note:** It is safer and more robust not to rely on the defaults but to set the mode explicitly.

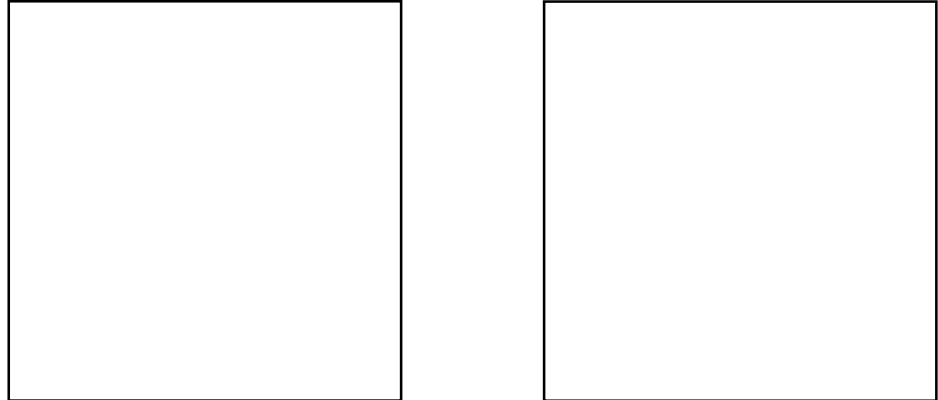
The graphics state determines the effect of a coordinate data block. If a state variable is required but undefined when a coordinate data block is processed the Gerber file is *invalid*. If a graphics state variable is not needed then it can remain undefined. For example, if the interpolation mode is G01 (linear interpolation) then the quadrant mode may remain undefined because it is not required for interpolating. However if the interpolation mode is switched to G02 or G03 (circular interpolation) the quadrant mode becomes required and thus must be defined.

## 2.3 Annotated Example Files

These annotated samples illustrate the use of the elements of the Gerber file format. If you are not familiar with the Gerber file format they can give you a feel for it which will make it easier to read the specification.

### 2.3.1 Example 1

Example 1 is a single-level image with two square boxes.



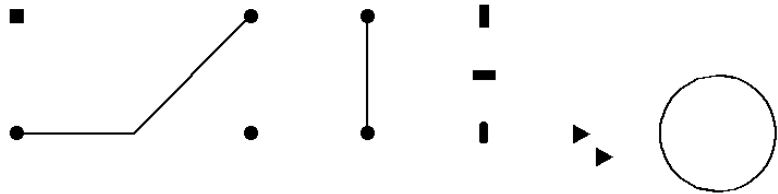
*Example 1: two square boxes*

G04 Ucamco ex. 1: Two square boxes*	A comment
%INBoxes*%	Image name set to 'Boxes', information only
%FSLAX26Y26*%	Coordinate format specification: Leading zeroes omitted Absolute coordinates 2 digits in the integer part 6 digits in the fractional part
%MOIN*%	Unit set to inch
%LPD%	Start a new level with dark polarity
%ADD10C,0.010*%	Define aperture with D-code 10 as a 0.01 inch circle
D10*	Select aperture with D-code 10 as current aperture
X0Y0D02*	Move to (0, 0)
G01X5000000Y0D01*	Draw to (5, 0) with D10
G01Y5000000D01*	Draw to (5, 5) with D10
G01X0D01*	Draw to (0, 5) with D10
G01Y0D01*	Draw to (0, 0) with D10
X6000000D02*	Move to (6, 0)
G01X11000000D01*	Draw to (11, 0) with D10

End of file

Example 2 illustrates the use of levels and various apertures.

Example 2 illustrates the use of levels and various apertures.



### EXAMPLE 2. VARIOUS SHAPES

Aperture macro 'TARGET125'

6,0,0,0.125,.01,0.01,3,0.00 3,0.150,0*%	Moiré primitive
%AMTHERMAL80*	Aperture macro 'THERMAL80'
7,0,0,0.080,0.055,0.0125,45 *%	Thermal primitive
%ADD10C,0.01*%	Aperture definition: D10 is a circle with diameter 0.01 inch
%ADD11C,0.06*%	Aperture definition: D11 is a circle with diameter 0.06 inch
%ADD12R,0.06X0.06*%	Aperture definition: D12 is a rectangle with size 0.06 x 0.06 inch
%ADD13R,0.04X0.100*%	Aperture definition: D13 is a rectangle with size 0.04 x 0.1 inch
%ADD14R,0.100X0.04*%	Aperture definition: D14 is a rectangle with size 0.1 x 0.04 inch
%ADD15O,0.04X0.100*%	Aperture definition: D15 is an obround with size 0.04 x 0.1 inch
%ADD16P,0.100X3*%	Aperture definition: D16 is a polygon with 3 vertices and circumscribed circle with diameter 0.1 inch
%ADD17P,0.100X3*%	Aperture definition: D17 is a polygon with 3 vertices and circumscribed circle with diameter 0.1 inch
%ADD18TARGET125*%	Aperture definition: D18 is the special aperture called 'TARGET125' defined earlier
%ADD19THERMAL80*%	Aperture definition: D19 is the special aperture called 'THERMAL80' defined earlier
%LPD*%	Start a new level with dark polarityk
%LNXTEST1*%	Level name is 'XTEST1'
%SRX1Y1I0J0*%	Set 'Step and Repeat' to 1 for both X and Y. This is the default value for 'Step and Repeat'. The statement is not required but makes the intention clear.
D10*	Select aperture with D-code 10
X0Y250D02*	Move current point to (0, 0.25) inch
G01X0Y0D01*	Linear interpolation (draw)
G01X250Y0D01*	Linear interpolation (draw)
X1000Y1000D02*	Move current point
G01X1500D01*	Linear interpolation (draw)
G01X2000Y1500D01*	Linear interpolation (draw)

X2500D02*	Move current point. Since the X and Y coordinates are modal, Y is not repeated
G01Y1000D01*	Linear interpolation. The X coordinate is not repeated and thus its previous value of 2.5 inch is used
D11*	Select aperture with D-code 11
X1000Y1000D03*	Flash D11 at (1.0, 1.0). Y is modal.
X2000D03*	Flash D11 at (2.0, 1.0). Y is modal.
X2500D03*	Flash D11 at (2.5, 1.0). Y is modal.
Y1500D03*	Flash D11 at (2.5, 1.5). X is modal.
X2000D03*	Flash D11 at (2.0, 1.5). Y is modal.
D12*	Select aperture with D-code 12
X1000Y1500D03*	Move to (1.0, 1.5) and flash
D13*	Select new aperture with D-code 13
X3000Y1500D03*	Move to (3.0, 1.5) and flash
D14*	Select new aperture with D-code 14
Y1250D03*	Move to (3.0, 1.25) and flash
D15*	Select new aperture with D-code 15
Y1000D03*	Move to (3.0, 1.0) and flash
D10*	Select new aperture with D-code 10
X3750Y1000D02*	Move current point. This sets the start point for the following arc interpolation
G75*	Set multi quadrant mode
G03X3750Y1000I250J0D01*	Interpolate a complete circle
D16*	Select new aperture with D-code 16
X3400Y1000D03*	Flash D16
D17*	Select new aperture with D-code 17
X3500Y900D03*	Flash D17
D10*	Select new aperture with D-code 10
G36*	Start a region
X500Y2000D02*	Move current point to (0.5, 2.0)
G01Y3750D01*	Linear interpolation (draw)
G01X3750D01*	Linear interpolation (draw)

G01Y2000D01*	Linear interpolation (draw)
G01X500D01*	Linear interpolation (draw)
G37*	Create the region by filling the contour
D18*	Select new aperture with D-code 18
X0Y3875D03*	Flash D18
X3875Y3875D03*	Flash D18
%LNXTTEST2*LPC*%	Statement that contains two parameters: Level name is 'XTEST2' Level polarity is clear
G36*	Start a region
X1000Y2500D02*	Move current point to (1.0, 2.5)
G01Y3000D01*	Linear interpolation (draw)
G74*	Set single quadrant mode
G02X1250Y3250I250J0D01*	Clockwise arc with radius 0.25
G01X3000D01*	Linear interpolation (draw)
G75*	Set multi quadrant mode
G02X3000Y2500I0J-375D01*	Clockwise arc with radius 0.375
G01X1000D01*	Linear interpolation (draw)
G37*	Create the region by filling the contour
%LPD*%	Start a new level with dark polarity
%LNXTTEST3*%	Level name is 'XTEST3'
D10*	Select new aperture with D-code 10
X1500Y2875D02*	Move current point
G01X2000D01*	Linear interpolation (draw)
D11*	Select aperture with D-code 11
X1500Y2875D03*	Flash D11
X2000D03*	Flash D11
D19*	Select aperture with D-code 19
X2875Y2875D03*	Flash D19
M02*	End of file



## 3 Syntax

### 3.1 Character Set

A Gerber file is expressed in 7-bit ASCII characters codes 32 to 126 (i.e. the printable characters in ANSI X3.4-1986) plus characters codes 10 (LF, Line Feed) and 13 (CR, Carriage Return). No other characters are valid. Gerber files are therefore printable and human readable.

CR and LF can be used as line separators. Line separator style can be Windows (CRLF), Unix (LF), Macintosh (CR) or a mix. Line separators are only allowed between data blocks and within aperture macro definitions (see AM parameter).

The characters '\*' and '%' are reserved as resp. the end-of-block character and the parameter delimiter.

The character code 32 (SP, Space) can only be used in comments.



**Tip:** It is recommended to add line separators between data blocks to improve readability. They have no effect on the image.

### 3.2 Names

Names are used to identify macros, variables, images and levels. Names are made up of all valid characters *except* SP, CR, LF, '%', '\*', and ';'. Names *cannot* begin with a digit, a "+" or a "-". Names *cannot* contain be longer than 255 characters.

### 3.3 Data Blocks

Data blocks are building blocks for a Gerber file. Each data block ends with the mandatory end-of-block character asterisk '\*'. Each data block can contain one or more parameters, codes or coordinates.



**Example:**

```
X0Y0D02*
```

```
G01X50000Y0D01*
```

Data blocks are lower level syntactical elements of a Gerber file. The data blocks can be semantically interconnected so they form a group that represents a higher level element called a statement.







**Example:**

G02X0Y100I-400J100D01\*

In the example above the statement consists of a single data block that represents G02 and D01 function code together with a coordinate and offset in X and Y.



**Example:**

%AMDONUTFIX\*1,1,0.100,0,0\*1,0,0.080,0,0\*%

In this example the parameter statement contains an AM parameter built of three data blocks. There is no limitation on the number of statements that a Gerber file can contain.

## 3.5 Data Types

A Gerber file can contain the following data types:

- ❑ Function codes. See 3.5.1.
- ❑ Coordinate data. See 3.5.2.
- ❑ Parameters. See 3.5.3.

### 3.5.1 Function Codes

Function codes are either

- ❑ Operation codes operating on coordinate data, i.e. D01, D02 or D03.
- ❑ Codes that set a graphics state variable.



**Example:**

G74\*

If a code is located in the same data block as coordinate data, the graphics state is first changed before the operation code operates on the coordinate data. In other words the code applies to the

In the example below there are two data blocks. In the first block the function code 'G01' is followed by coordinate data. The G01 function code means 'start linear interpolation' and the coordinate data means the starting point (300, 200) for the interpolation. In the second data block the next interpolation point (1100, 200) is specified.



**Example:**

G01X300Y200D02\*

G01X1100Y200D01\*

Function codes are described in detail in chapter 0.

### 3.5.2 Coordinate Data Blocks

A coordinate data block consists of coordinate data followed by a function code D01, D02 or D03. The function code operates on the coordinate data.

A coordinate data block can be expressed using the formula:

**<Coordinate data>: [X<Number>][Y<Number>][I<Number>][J<Number>](D01|D02|D03)**

Syntax	Comments
X, Y	Characters used to indicate X, Y coordinates of a point
I, J	Characters used to indicate an offset in the X, Y direction
<Number>	When used with X or Y – decimal digits (possibly with a sign) defining X or Y coordinate of a point.  When used with I or J – decimal digits (possibly with a sign) defining an offset in the X or Y direction.
D01 D02 D03	Function codes that determine the effect of the coordinate preceding it. Their meaning is explained below.

The FS and MO parameters specify how to interpret the digits following the X, Y, I, J characters.

Coordinate data define points in the plane using a right-handed ortho-normal coordinate system. The plane is infinite, but implementations can have size limitations.

Coordinates are modal. If an X is omitted the X coordinate of the current point is used. The same applies to Y.

Offsets are *not* modal. If I or J is omitted the default is zero (0).

Each coordinate data block must end with a one and only one instance of an operation code (D01, D02 or D03). The operation code operates on the coordinate data.



#### Examples of coordinate data blocks

X200Y200D02*	point (+200, +200) and offset (0,0) operated upon by D02
Y-300D0*	point (+200, -300) and offset (0,0) operated upon by D03
I300J100D01*	point (+200, -300) and offset (+300, +100) operated upon by D01
X200Y200I50J50D01*	point (+200, +200) and offset (+50, +50) operated upon by D01
X+100I-50D01*	point (+100, +200) and offset (-50, 0) operated upon by D01

Note that X and Y are modal in a coordinate data block. Consequently, in a data block “D0n\*” (n = 1,2, or 3) without explicit X and Y, the X and Y values are supplied by modal operation.

In the example below D03 operates on the last point, supplied by the modal action of X and Y.



#### Example

D03\*

### 3.5.3 Parameters

Parameters define characteristics of the file. Most parameters affect how coordinate data is processed.



**Note:** Originally parameters were called Mass Parameters.

Parameters operating on the entire image must be placed at the beginning of the file. Parameters generating a new level are placed at the appropriate place in the file.

Parameters consist of a two-character parameter code followed by parameter data. The parameter code indicates which parameter is used. The parameter data depends on the parameter code.

Parameters are enclosed into a pair of delimiter '%' characters. Usually a parameter consists of a single data block ending with a '\*'. The AM parameter can include more than one data block. The '%' must immediately follow the '\*' of the last data block without intervening line separators. This is an exception to the general rule that a data block can be followed by a line separator.



#### Examples:

```
%FSLAX23Y23*%
```

```
%AMDONUTFIX*1,1,0.100,0,0*1,0,0.080,0,0*%
```

Parameters may be provided single or grouped between '%' delimiters, up to a maximum of 4096 characters between these delimiters.



#### Example:

```
%SFA1.0B1.0*ASAXBY*%
```

Line separators are permitted between parameters to improve readability. For a set of parameters the syntax is:

```
%<Parameter>{{<Line separator>}<Parameter>}%
```



#### Example:

```
%SFA1.0B1.0*
```

```
ASAXBY*%
```



**Tip:** For readability it is recommended to have one parameter per line.

Use an explicit decimal point with all numerical values associated with a parameter. If the decimal point is omitted, an integer value is assumed.

The syntax for an individual parameter is:

**%Parameter code<required modifiers>[optional modifiers]\*%**

Syntax	Comments
Parameter code	2-character code (AD, AM, FS, etc...)
<required modifiers>	Must be entered to complete definition
<optional modifiers>	May be required depending on the required modifiers



## 4 Function Codes

Code	Description	Comments
D01	Plot operation code	<p>If region mode is off D01 creates a draw or arc using the current aperture. Only specific apertures can be used; see 2.2.4.</p> <p>When region mode is on D01 creates a contour segment. The current aperture is not used.</p> <p>After the object is created the current point is moved to the coordinate</p>
D02	Move operation code	D02 does not create a graphics object but move the current point to the coordinate.
D03	Flash operation code	<p>With region mode is off D03 flashes the current aperture. D03 is not allowed in region mode.</p> <p>After the flash is created the current point is moved to the coordinate</p>
D10 and higher	Set the current aperture'	Sets the current aperture to a number defined by an AD parameter.
G01	Set the interpolation mode to linear	A modifier of the plot operation code D01. See 4.2 and 4.3.
G02	Set the interpolation mode to 'Clockwise circular interpolation'	
G03	Set the interpolation mode to 'Counterclockwise circular interpolation'	
G04	Ignore data block	Used for comments.
G36	Set region mode on.	Used to create regions. See 4.4.
G37	Set region mode off.	
G74	Set quadrant mode to 'Single quadrant'	A modifier of the circular interpolation mode. See dedicated section for more details.
G75	Set quadrant mode to 'Multi quadrant'	
M02	Indicates the end of the file	Every file must end in a M02. It can only occur once, at the end of the file. No data is allowed after M02.

*Current codes*

## 4.1 Syntax

This section provides a general description of the syntax for current function codes. Detailed information for each specific code can be found in subsequent sections.

A function code is identified by a code letter followed by a code number. The code letter can be G, D or M. (These letters are historic, originating from the original EIA RS-274-D specification.)

Most of the function codes manipulate the graphics state. The G04 function code is used for comments and the M02 function code indicates the end of the file.

The syntax for G04 is the following:

**<Comment>: G(4|04)<Comment string>\***

Section 3.1 gives the syntax rules for the comment string.



### Example:

```
G04 This is a comment*
```

The syntax for M02:

**<End of file marker>: M02\***

The data block with the M02 function code must be the last data block in the file.

Other function codes set graphics state variables.

The function codes G01, G02, G03, D01, D02 and D03 can be put together with coordinate data in the same data block. In this case, the graphics state is modified before it operates on the coordinate data, even if the function code trails the coordinate data.



### Example:

```
G01X100Y100D01*  
X200Y200D01*
```

G01 sets the interpolation mode to linear and this used to process the coordinate data X100Y100 from the same data block as well as the coordinate data X200Y200 from the next data block.

The syntax for G01, G02, G02, D01 and D02 is the following:

**<Interpolation>: [G(1|01|2|02|3|03)][<Coordinate data>D(1|01|2|02)]\***

The following data blocks are syntactically valid:

```
G01*  
X100Y100D01*  
G01X500Y500D01*  
X300Y300D01*  
G01X100Y100D01*
```

A valid data block must contain at least one of the parts.



**Warning:** It is recommended always to specify G01/G02/G03 together with D01 in each coordinate block. The block then contains all the information needed to generate the graphic object. It makes the file easier to read for man and machine.



**Example:**

G01X100Y200D01\*

It is allowed to specify D02 (move) with any of G01/G02/G03. However the G01/G02/G03 are then ignored. The recommended syntax for the D02 function code is the following:

**<Move current point>: [X<Coordinate>][Y<Coordinate>]D02\***

Syntax	Comments
X<Coordinate>	Defines the X coordinate of the new current point. If missing then the previous X coordinate is used.
Y<Coordinate>	Defines the Y coordinate of the new current point. If missing then the previous Y coordinate is used.
D02	Move operation code.



**Example:**

X200Y100D02\*

The syntax for the D03 function code ('Flash' mode) is:

**<Flash current aperture>: [X<Coordinate>][Y<Coordinate>]D03\***

Syntax	Comments
X<Coordinate>	Defines the X coordinate of the flash point. If missing then the previous X coordinate is used.
Y<Coordinate>	Defines the Y coordinate of the flash point. If missing then the previous Y coordinate is used.
D03	Flash operation code



**Example:**

X100Y100D03\*

Each of the remaining function codes is put into a separate data block:



**<Function code usage>: (G36|G37|G74|G75|D10|D11|...)\***



**Example:**

```
G36*
X200Y1000D02*
G01X1200D01*
G01Y200D01*
G01X200D01*
G01Y600D01*
G01X500D01*
G75*
G03X500Y600I300J0D01*
G74*
G01X200D01*
G01Y1000D01*
G37*
```

## 4.2 Linear Interpolation

Linear interpolation generates a straight line from the current point to the point with X, Y coordinates specified by the data block. The current point is then set to the X, Y coordinates specified by the data block. The resulting graphic object is called a 'draw'.

### 4.2.1 Data Block Format

The syntax for the linear interpolation code is:

**<Linear interpolation>: G(01|1)[X<Coordinate>][Y<Coordinate>][D(01|02)]\***

Syntax	Comments
G(01 1)	G01 or G1 – Sets interpolation mode to 'Linear interpolation'
X<Coordinate>	Defines the X coordinate of the draw end point. If missing then the previous X coordinate is used.
Y<Coordinate>	Defines the Y coordinate of the draw end point. If missing then the previous Y coordinate is used.
D(01 02)	Plot/Move operation code



**Example:**

```
G01X0Y250D01*
```

## 4.3 Circular Interpolation

### 4.3.1 Overview

Circular interpolation generates a circular arc from the current point to the point with X, Y coordinates specified by the data block; the center of the arc is specified by the offsets I and J. The current point is then set to the X, Y coordinates specified by the data block.

There are two orientations:

- Clockwise, set by G02
- Counterclockwise, set by G03

The orientation is defined around the center of the arc, moving from begin to end.

There are two quadrant modes:

- Single quadrant mode (G74)
- Multi quadrant mode (G75)

Quadrant mode	Comments
Single quadrant (G74)	<p>In single quadrant mode the arc is not allowed to extend over more than 90°. The following relation must hold:</p> <p><math>0^\circ \leq A \leq 90^\circ</math>, where A is the arc angle</p> <p>If the start point of the arc is equal to the end point, the arc has length zero, i.e. it covers 0°. A data block is required for each quadrant. A minimum of four coordinate data blocks is required for a full circle.</p>
Multi quadrant (G75)	<p>In multi quadrant mode the arc is allowed to extend over more than 90°. To avoid ambiguity between 0° and 360° arcs the following relation must hold:</p> <p><math>0^\circ &lt; A \leq 360^\circ</math>, where A is the arc angle</p> <p>If the start point of the arc is equal to the end point, the arc is a full circle of 360°.</p>

#### *Quadrant modes*

The codes G74 and G75 allow switching between the two quadrant modes. A data block containing G75 turns on multi quadrant mode. Every block following it will be interpreted as multi quadrant, until cancelled by a G74. A data block containing G74 code turns on single quadrant mode.



**Warning:** A Gerber file containing arcs but without a preceding G74 or G75 code is invalid.

### 4.3.2 Definition of arcs

In a circular arc the center must be at exactly the same distance - radius - from the start point and the end point. The two radii must be exactly equal. The definition of the arc is obvious when the radii are equal.

However, as Gerber file has a finite resolution the radii cannot be equal, except in special cases. Furthermore the software generating the Gerber file unavoidably adds rounding errors of its own. The two radii are different for almost all arcs, unavoidably so. We will call the difference between the radii the *arc deviation*.

This raises the question which is the image represented by a “circular arc” with unequal radii. It is defined *as a continuous and monotonous curve going from the start point to the end point, approximating the circle with the given center and a radius equal to the average of the two radii within a few arc deviations*. For the special case of zero arc deviation this reduces to a mathematically perfect circular arc.

This definition has a fuzziness of the order magnitude of the arc deviation. The writer of the file accepts any interpretation within the fuzziness above as valid. If the writer requires a more precise interpretation of the arc he needs to write arcs with lower deviation, by using a high coordinate resolution and rounding carefully.

Note that self-intersecting contours are not allowed, see 4.4. If any of the valid arc interpretations turns the contour in a self-intersecting one, the file is invalid, with unpredictable results.

Most real-life issues come from using a low coordinate resolution. Using high coordinate resolution is an obvious first step to minimize the arc deviation and potential problems.

### 4.3.3 Numerical instability in multi quadrant (G75) arcs

In G75 mode small changes in the position of center point, start point and end point can swap the large arc with the small one, dramatically changing the image.

This most frequently occurs with very small arcs. Start point and end point are close together. If the end point is slightly moved it can end on top of the start point. Under G75, if the start point of the arc is equal to the end point, the arc is a full circle of 360°, see 4.3.1. A small change in the position of the end point has changed the very small arc to a full circle.

Under G75 rounding must be done carefully. Using high resolution is an obvious prerequisite.

The Gerber writer must also consider that the reader unavoidably has rounding errors. Perfectly exact numerical calculation cannot be assumed. It is the responsibility of the writer to avoid unstable arcs.

Under G74 arcs are always less than 90° and this numerical instability does not exist. G74 is intrinsically stable. Another option is not to use very small arcs, e.g. by replacing them with draws - the error is very small and draws are stable.

### 4.3.4 Single Quadrant Mode

Single quadrant mode is set by a G74 code.



**Example:**

G74\*

#### 4.3.4.1 Data Block Format

The syntax in single quadrant mode is:

**<Circular interpolation>: G(02|2|03|3)[X<Coordinate>][Y<Coordinate>]  
[I<Distance>][J<Distance>][D(01|02)]\***

Syntax	Comments
G(02 2 03 3)	Sets the interpolation mode': G02 or G2 – 'Clockwise circular interpolation' G03 or G3 – 'Counterclockwise circular interpolation'
X<Coordinate>	Defines the X coordinate of the arc end point. If missing then the previous X coordinate is used.
Y<Coordinate>	Defines the Y coordinate of the arc end point. If missing then the previous Y coordinate is used.
I<Distance>	The distance between the arc start point and the center parallel to the X axis. The value is always positive. A sign is not allowed. The sign of the offset to the center is determined implicitly. If missing then a 0 distance is used.
J<Distance>	The distance between the arc start point and the center parallel to the X axis. The value is always positive. A sign is not allowed. The sign of the offset to the center is determined implicitly. If missing then a 0 distance is used.
D(01 02)	Plot/Move operation code



**Note:** Because the sign in offsets is omitted, there are four candidates for the center: (<Current X> +/- <X distance>, <Current Y> +/- <Y distance>). The center is the candidate that results in an arc with the specified orientation and not greater than 90°.



**Warning:** If the center is not precisely positioned, there may be none or more than one candidate fits. In that case the arc is invalid. The creator of the file accepts any interpretation.



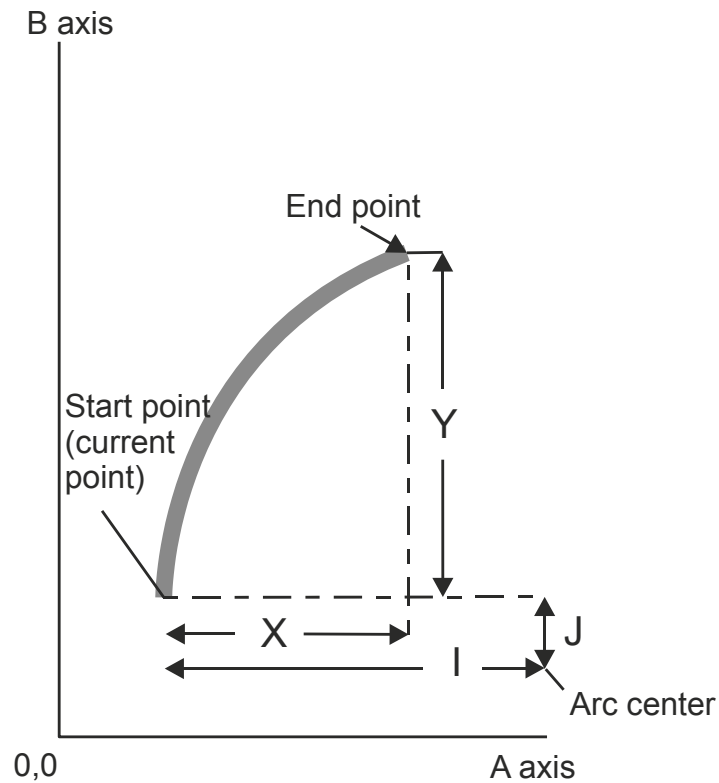
**Example:**

G74\*

G03X700Y1000I400J0D01\*

#### 4.3.4.2 Image

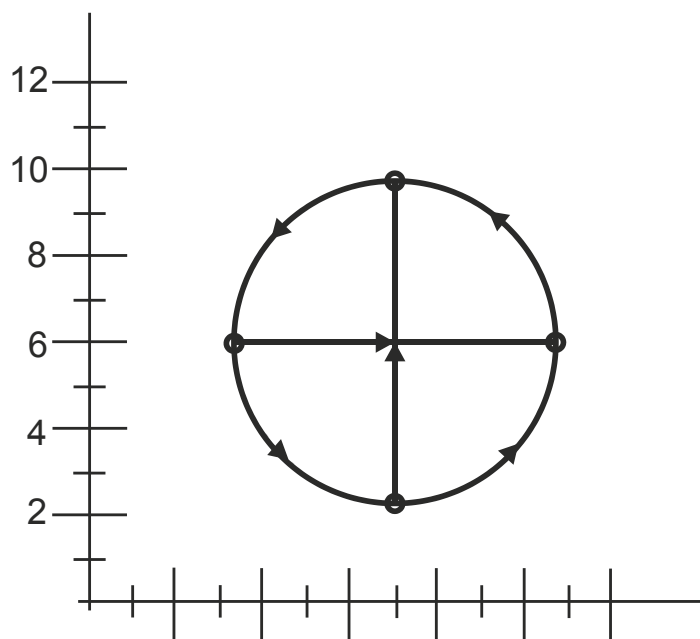
The coordinates of an arc endpoint and the center distances are interpreted according to the coordinate format specified by the FS parameter and the unit specified by the MO parameter. The following image illustrates how arcs are interpolated.



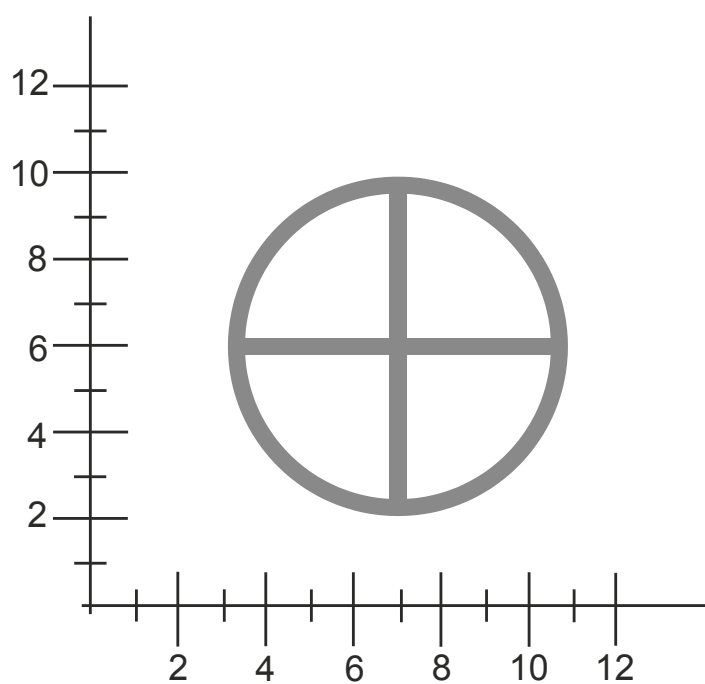
*Single quadrant mode*

#### 4.3.4.3 Example

Syntax	Comments
G74*	Single quadrant mode
D10*	Use aperture D10
X1100Y600D02*	Start from (11, 6)
G03X700Y1000I400J0D01*	Quarter arc (radius 4) to (7, 10)
G03X300Y600I0J400D01*	Quarter arc (radius 4) to (3, 6)
G03X700Y200I400J0D01*	Quarter arc (radius 4) to (7, 2)
G03X1100Y600I0J400D01*	Quarter arc (radius 4) to (11, 6)
X300D02*	Start from (3 ,6)
G01X1100D01*	Draw to (11, 6)
X700Y200D02*	Start from (7, 2)
G01Y1000D01*	Draw to (7, 10)



*Single quadrant mode example: arcs and draws*



*Single quadrant mode example: resulting image*

### 4.3.5 Multi Quadrant Mode

The multi quadrant mode is set by a G75 code.



**Example:**

G75\*

#### 4.3.5.1 Data Block Format

The syntax in multi quadrant mode is:

**<Circular interpolation>: G(02|2|03|3)[X<Coordinate>][Y<Coordinate>]  
[I<Offset>][J<Offset>][D(01|02)]\***

Syntax	Comments
G(02 2 03 3)	Sets the interpolation mode: G02 or G2 – 'Clockwise circular interpolation' G03 or G3 – 'Counterclockwise circular interpolation'
X<Coordinate>	Defines the X coordinate of the arc end point. If missing then the previous X coordinate is used.
Y<Coordinate>	Defines the Y coordinate of the arc end point. If missing then the previous Y coordinate is used.
I<Offset>	Defines the offset or signed distance between the arc start point and the center measured parallel to the X axis. If missing then a 0 offset is used.
J<Offset>	Defines the offset or signed distance between the arc start point and the center measured parallel to the X axis. If missing then a 0 offset is used.
D(01 02)	Operation code



**Note:** In multi quadrant mode the offsets in I and J are signed. If no sign is present, the offset is positive.



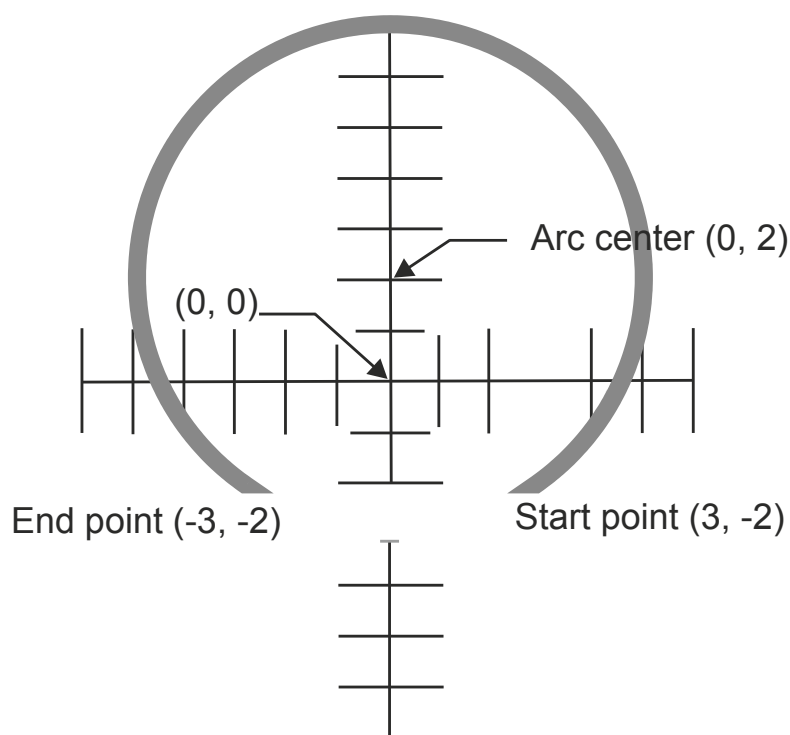
**Example:**

G75\*

G03X-300Y-200I-300J400D01\*

### 4.3.6 Example

Syntax	Comments
X300Y-200D02*	Move to (3, -2)
G75*	Set multi quadrant mode
G03X-300Y-200I-300J400D01*	Arc counterclockwise to (-3,-2); offsets from the start point to the center point are -3 for X and 4 for Y, i.e. the center point is (0, 2)
G74*G01*	Back to linear interpolation mode and G74



*Multi quadrant mode example: resulting image*



### 4.3.7 Using G74 or G75 can result in completely different image

An arc statement can define a completely different image under G74 and G75. The two sample files below differ only in G74/G75, but they define a dramatically different image.

Syntax	Comments
D10*	Use aperture D10
G01X0Y600D02*	Start from (0, 6)
G74	Single quadrant mode
G02X0Y600I500J0D01*	Arc to (0, 6) with radius 5

The resulting image is small dot, an instance of the aperture at position (0, 6)

Syntax	Comments
D10*	Use aperture D10
G01X0Y600D02*	Start from (0, 6)
G75	Multi quadrant mode
G02X0Y600I500J0D01*	Arc to (0, 6) with center (5,6)

The image is a full circle.



**Warning:** It is essential to always specify G74 or G75 if arcs are used.

## 4.4 Regions (G36/G37)

### 4.4.1 Overview

A region is a graphic object with an arbitrary shape defined by its closed contour.

A contour is a sequence of connected draw or arc segments. A pair of segments is said to connect only if they are defined consecutively, with the second segment starting where the first one ends. Thus the order in which the segments of a contour are defined is significant. Nonconsecutive segments that meet or intersect fortuitously are not considered to connect. A contour is closed: the end point of the last segment of a contour must be equal to the start point of the first segment.

The G36 code set region mode on and the G37 code sets it off.

With region mode on, the operation codes D01 and D02 are used to create contours. D02 sets the start point of a contour and D01 creates contour segments. The segments are not graphics objects in themselves; they do not darken or clear the image area directly. Many segments can go into the specification of a single contour. As soon as a D02 moves the current point a contour is considered finished. A new contour is started when a D01 is performed.


When region mode is turned off the region objects are created by filling all the contours individually. The filled area is the union of the areas filled for the individual contours. Different contours can overlap.


Self-intersecting contours are not allowed because their interpretation is not obvious. Contour edges can coincide, allowing cut-ins to create holes in solid regions.


Moves, draws or arcs with zero length in contours are ignored. (It is recommended to avoid such constructs; they have no use and can confuse the reader.)


In region mode no parameters are allowed, nor any other D codes than D01 and D02.


The current aperture has no effect on a region.

 **Note:** Cut-ins are complex constructions and should be used only for the simplest cases. They are not recommended to create planes with in it. It is recommended to create holes with alternating dark and clear polarity levels.

 **Note:** In the 1960's and 1970s, the era of vector plotters, a region had to be created by painting (aka filling or stroking) it with draws. This produces the correct image, but the file size explodes and painted regions cannot be handled in CAM. Painting must no longer be used.

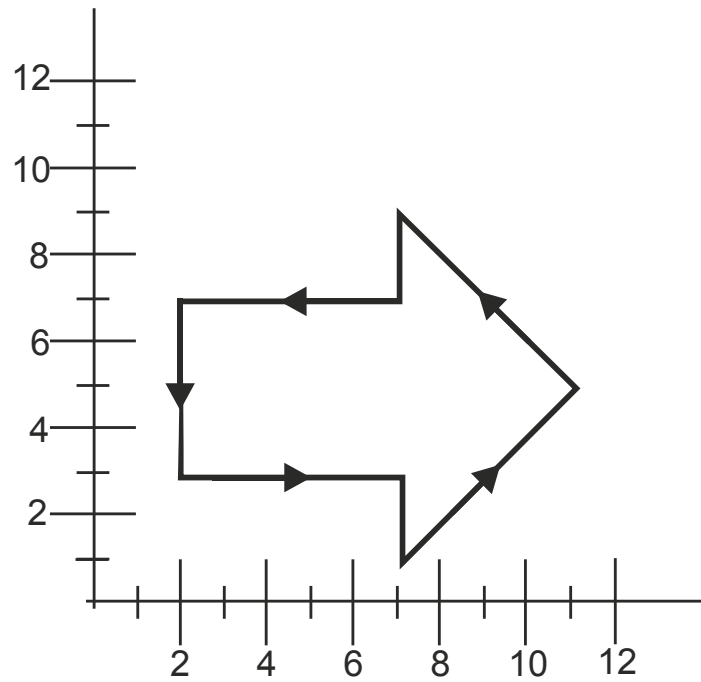
 **Note:** In previous versions of this document “contour fill” was called “polygon fill”.

 **Warning:** Care must be taken that rounding does not turn a proper contour into a self-intersecting one, leading to unpredictable results. The Gerber writer must also consider that the reader unavoidably has rounding errors. Perfectly exact numerical calculation cannot be assumed. It is the responsibility of the writer to avoid marginal contours that become self-intersecting under normal rounding. Most real-life problems come from using low file coordinate resolution, so using the highest resolution is a recommended.

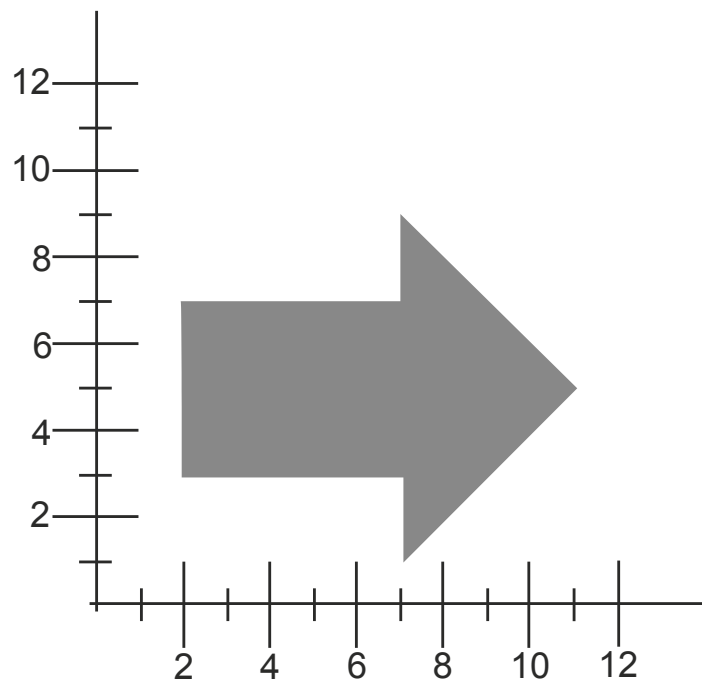
 **Warning:** An arc can be validly interpreted by any curve in a range, see 4.3. If any of these curves results in a self-intersecting contour the file is invalid and the result is unpredictable.

#### 4.4.2 Simple contour example

Syntax	Comments
G36*	Start a region
X200Y300D02*	Move the current point to (2, 3)
G01X700D01*	Line segment to (7, 3)
G01Y100D01*	Line segment to (7, 1)
G01X1100Y500D01*	Line segment to (11, 5)
G01X700Y900D01*	Line segment to (7, 9)
G01Y700D01*	Line segment to (7, 7)
G01X200D01*	Line segment to (2, 7)
G01Y300D01*	Line segment to (2, 3)
G37*	Create the region by filling the contour



*Simple contour example: draws*



*Simple contour example: resulting image*

### 4.4.3 Using levels to create holes

The recommended way to create holes in regions is by using levels with alternating dark and clear polarity, as illustrated in the following example. The file has four levels. The first level has dark polarity and contains the big square region. The second level has clear polarity and contains a circular disk; the disk is cleared from the image and creates a round hole in the big square. The third level has dark polarity and contains a small square that is darkened on the image inside the hole. The fourth level has clear polarity and contains a small disk; the disk erases parts of the big and the small squares.

The file uses absolute notation with the leading zeros omitted. The units are millimeters.



#### Example:

G04 This file illustrates how to use levels to create holes\*

%FSLAX27Y27\*%

%MOMM\*%

G04 First level: big square - dark polarity\*

%LPD\*%

%LNBIGSQUARE\*%

G36\*

X2500000Y2500000D02\*

G01X1750000D01\*

G01Y1750000D01\*

G01X2500000D01\*

G01Y2500000D01\*

G37\*

G04 Second level: big circle - clear polarity\*

%LPC\*%

%LNBIGCIRCLE\*%

G36\*

G75\*

X5000000Y10000000D02\*

G03X5000000Y10000000I5000000J0D01\*

G37\*

G04 Third level: small square - dark polarity\*

%LPD\*%

%LNSMALLSQUARE\*%

G36\*

X7500000Y7500000D02\*

G01X1250000D01\*

G01Y1250000D01\*

G01X7500000D01\*

G01Y7500000D01\*

G37\*

G04 Fourth level: small circle - clear polarity\*

%LPC\*%

%LNSMALLCIRCLE\*%

G36\*

G75\*

X11500000Y10000000D02\*

G03X11500000Y10000000I2500000J0D01\*

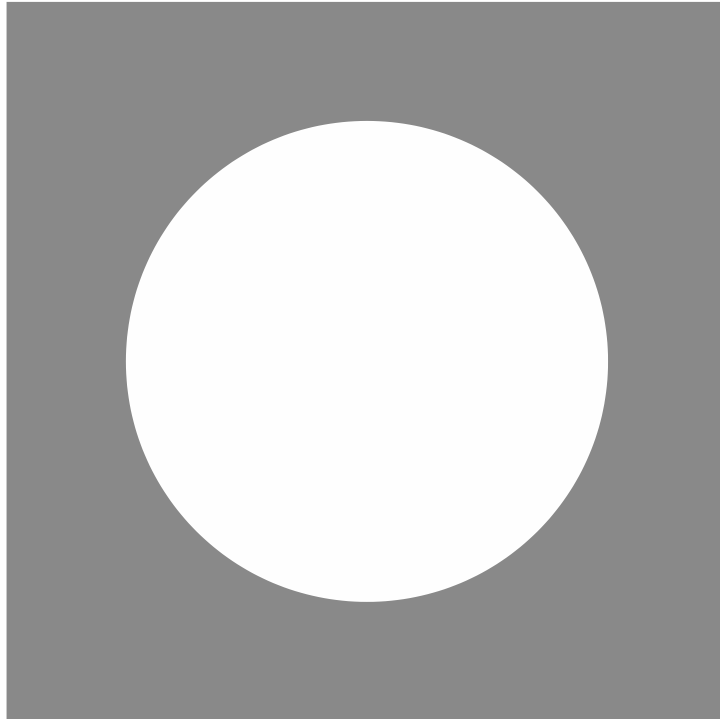
G37\*

M02\*

Below there are pictures which show the resulting image after adding each level.



*Resulting image: first level only*



*Resulting image: first and second levels*



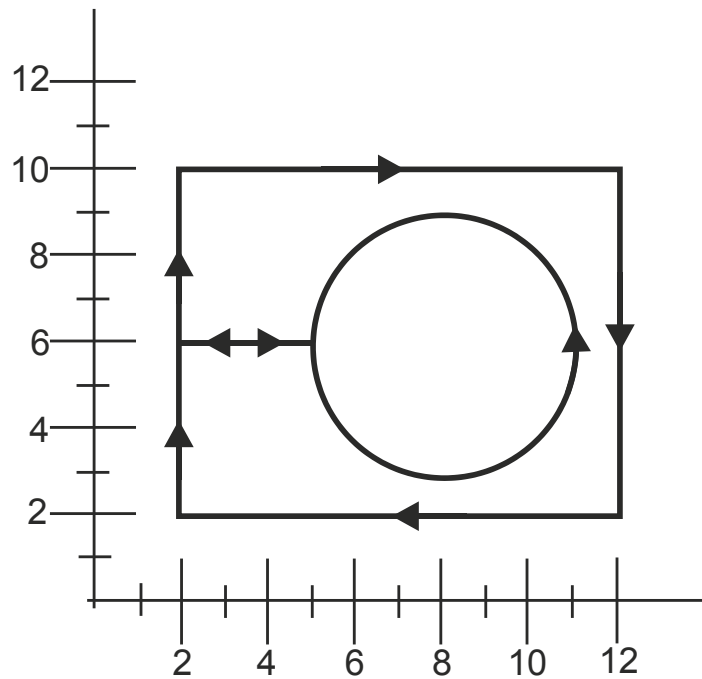
*Resulting image: first, second and third levels*



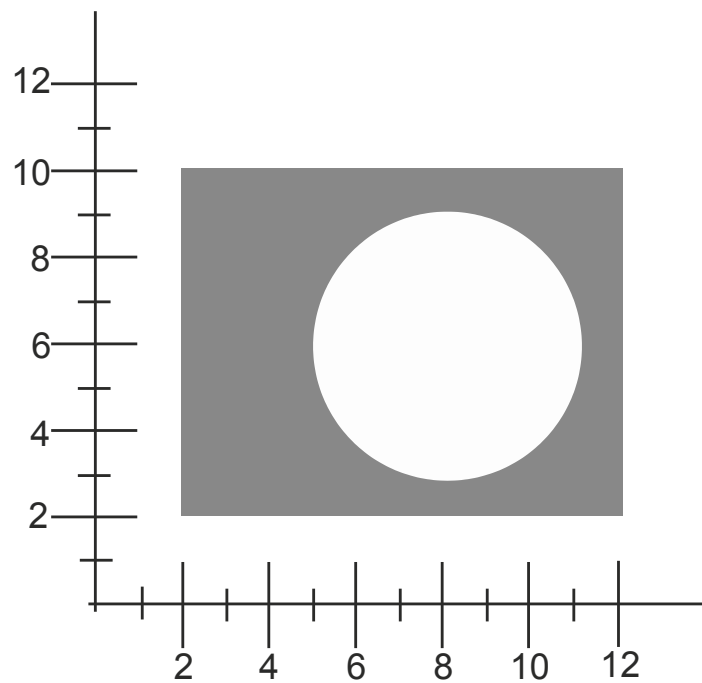
*Resulting image: all four levels*

#### 4.4.4 Cut-in example 1

Syntax	Comments
G36*	Initiate a region
X200Y1000D02*	Move the current point to (2,10)
G01X1200D01*	Line segment to (12,10)
G01Y200D01*	Line segment to (12, 2)
G01X200D01*	Line segment to (2, 2)
G01Y600D01*	Line segment to (2, 6)
G01X500D01*	Line segment to (5, 6)
G75*	Multi quadrant mode
G03X500Y600I300J0D01*	Full circle counterclockwise (radius is 300)
G74*	Single quadrant mode
G01X200D01*	Line segment to (2, 6)
G01Y1000D01*	Line segment to (2, 10)
G37*	Create the region by filling the contour



*Cut-in example 1: arcs and draws*



*Cut-in example 1: resulting image*

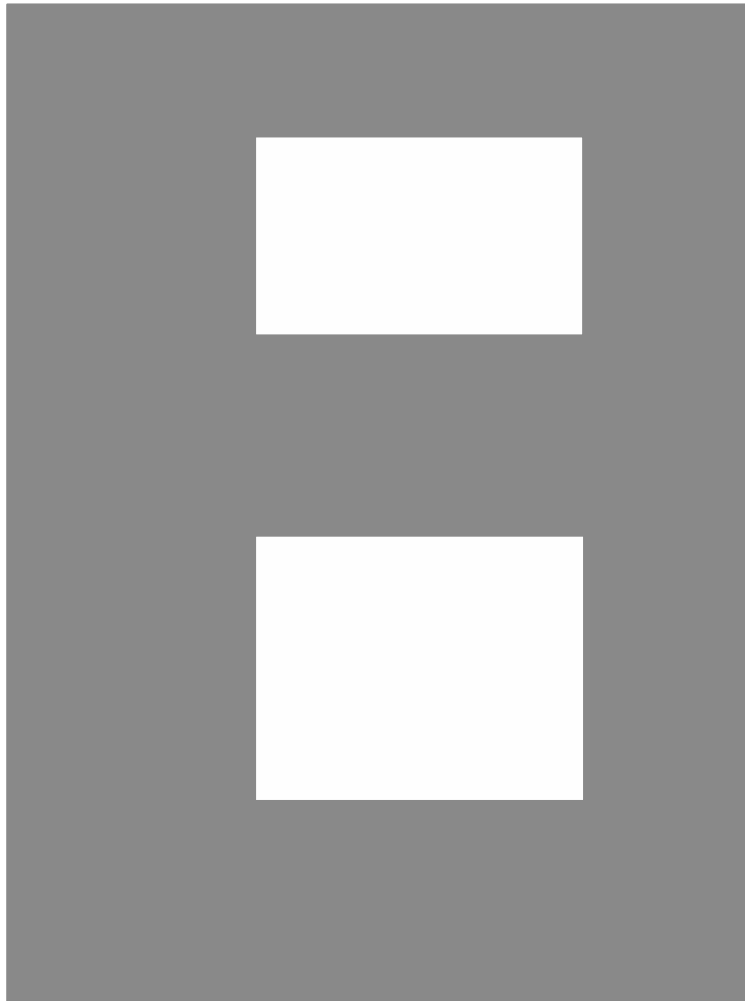


#### 4.4.5 Cut-in example 2 – good practice

Contours with cut-ins are susceptible to rounding problems: when the vertices move due to the rounding the contour may become self-intersecting. This may lead to unpredictable results. Construct cut-ins defensively, as in example 1 below, where draws which are on top of one another have the *same* end vertices. When the vertices move under rounding, the draws will remain exactly on top of one another, and no self-intersections are created.

```
G36*  
X1220000Y2570000D02*  
G01Y2720000D01*  
G01X1310000D01*  
G01Y2570000D01*  
G01X1250000D01*  
G01Y2600000D01*  
G01X1290000D01*  
G01Y2640000D01*  
G01X1250000D01*  
G01Y2670000D01*  
G01X1290000D01*  
G01Y2700000D01*  
G01X1250000D01*  
G01Y2670000D01*  
G01Y2640000D01*  
G01Y2600000D01*  
G01Y2570000D01*  
G01X1220000D01*  
G37*
```





*Cut-in example 2: resulting image*

#### 4.4.6 Cut-in example 3 – poor practice

Example below creates the same image as example 2, but with a less robust construction. The number of draws has been reduced by eliminating vertices between collinear draws. When the vertices move slightly due to rounding, the draws that were on top of one another may become intersecting, with unpredictable results. When a Gerber file moves from system to system, numerical rounding must be expected. Therefore this construction is bad practice.

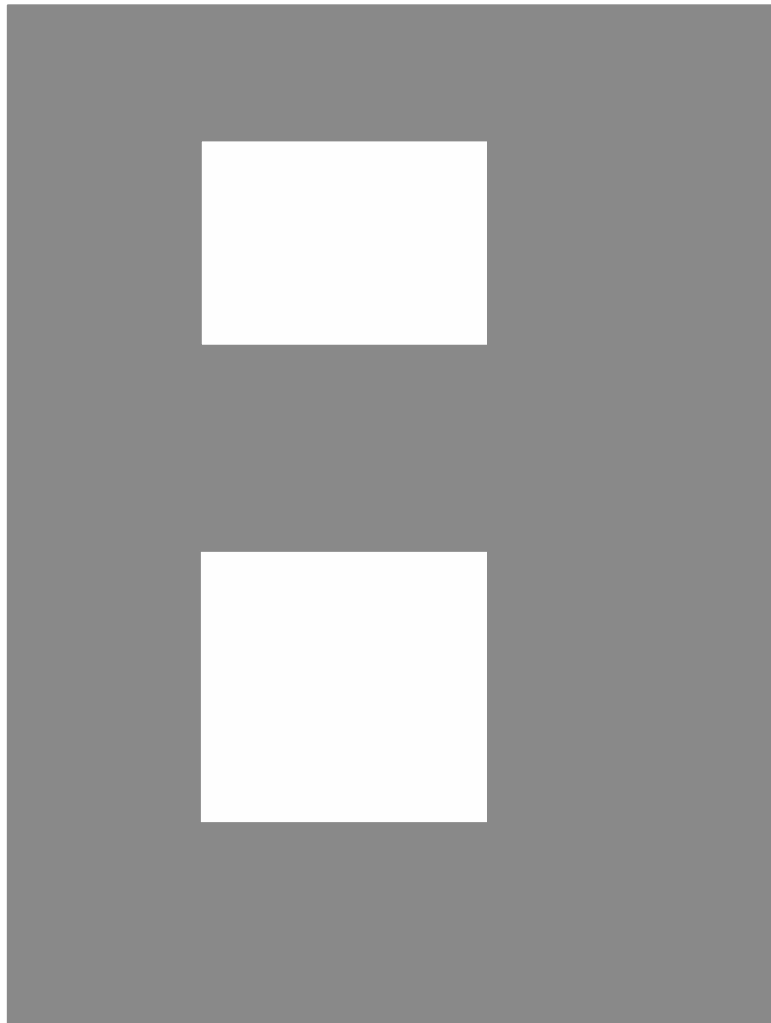
```
G36*  
X1110000Y2570000D02*  
G01Y2600000D01*  
G01X1140000D01*  
G01Y2640000D01*  
G01X1110000D01*  
G01Y2670000D01*  
G01X1140000D01*  
G01Y2700000D01*  
G01X1110000D01*  
G01Y2570000D01*  
G01X1090000D01*  
G01Y2720000D01*  
G01X1170000D01*  
G01Y2570000D01*  
G01X1110000D01*  
G37*
```

This results in the following contour:



*Cut-in example 3: poor practice*

This contour represents the following image:



*Cut-in example 2: resulting image*



## 5 Parameters

Parameters define characteristics of the file. The table below lists the current parameters. They are explained in detail in this chapter.

Parameter	Name	Description	Comments
FS	Format Specification	Sets the 'Coordinate format' graphics state variable	These parameters are only to be used once, at the beginning of the file, before object generation starts
MO	Mode (inch or mm units)	Sets the 'Unit' graphics state variable	
IP	Image Polarity	Sets the 'Image polarity' graphics state variable	
IN	Image Name	Sets the name of the file image	
AD	Aperture Definition	Assigns a D code number to an aperture definition	These parameters can be used multiple times
AM	Aperture Macro	Defines special apertures which can be referenced from the AD parameter	
SR	Step and Repeat	Sets the 'Step and Repeat' graphics state variable	
LP	Level Polarity	Starts a new level and sets the 'Level polarity' graphics state variable	
LN	Level Name	Sets the name of the current level	

*Current parameters*



### Example:

```
G04 Beginning of the file*
%FSLAX25Y25*%
%MOIN*%
%SRX1Y1I0.00000J0.00000*%
%ADD10C,0.000070*%
%LPD*%
%LNarc*%
...
M02*
```

### 5.1 FS – Format Specification

The FS parameter specifies the format of the coordinate data. It must be used only once at the beginning of a file. It must be specified before the first use of coordinate data.





**Note:** It is recommended to put the FS parameter at the very first line, maybe after some general comments.

The FS parameter specifies the following format characteristics:

- ☐ Number of integer and decimal places in coordinate data (coordinate format)
- ☐ Zero omission (leading or trailing zeroes omitted)
- ☐ Absolute or incremental coordinate notation



**Warning:** Explicit decimal points in coordinates are not allowed.

### 5.1.1 Coordinate Format

The coordinate format specifies the number of integer and decimal places in the coordinate data. For example, the “23” format specifies 2 integer and 3 decimal places. A maximum of 7 integer and 7 decimal places can be specified (nnnnnnn.nnnnnnn). The same format must be defined for X and Y. Signs are allowed. The ‘+’ sign is optional.



**Note:** In previous versions of the specification the implementation limit on integer and decimal places was 6. However, some applications started to generate 7 decimal places because they needed the accuracy. We adapted the specification to technology requirements and raised the limit to 7. However, there are probably still a number of Gerber readers in use that can only handle 6. Therefore it is recommended to use 7 decimal places only if the extra accuracy is needed.



**Warning:** It is strongly recommended to use 6 decimal places at least. A lower number of decimal places lose vital precision. The option to use a lower number of decimal places is a simplistic compression method introduced in the 1950’s, when saving a few bytes was of paramount importance and computers were too feeble for proper compression algorithms. Nowadays the few bytes saved are irrelevant. Modern compression methods far outperform this simplistic method, without any loss of accuracy. If the extra digits are not significant, they will be compressed away. The benefits of a small number of decimal digits are long gone. The disadvantages remain. It is a source of endless confusion. Always use 6 digits at least.

The resolution of a Gerber file is the distance expressed by the least significant digit of coordinate data. Thus the resolution is the size of grid steps of the coordinates.

The unit in which the coordinates are expressed is set by the %MO parameter. See 5.2.

### 5.1.2 Zero Omission

Zero omission allows reducing the size of data by omitting *either* leading or trailing zeroes from the coordinate values.

With *leading zero omission* some or all leading zeroes can be omitted but all trailing zeroes are required. To interpret the coordinate string, it is first padded with zeroes in front until its length fits the coordinate format. For example, with the “23” coordinate format, “015” is padded to “00015” and therefore represents 0.015.

With *trailing zero omission* some or all trailing zeroes can be omitted but all leading zeroes are required. To interpret the coordinate string, it is first padded with zeroes at the back until its length fits the coordinate format. For example, with the “23” coordinate format, “15” is padded to “15000” and therefore represents 15.000.


Leading zero omission is easier to read.

If the coordinate data in the file does not omit zeroes it is conventional to specify leading zero omission.

### 5.1.3 Absolute or Incremental Notation

Coordinate values can be expressed either as absolute coordinates (absolute notation) or as incremental distances from a previous coordinate position (incremental notation).

**Incremental notation is deprecated.**

 **Warning:** It is strongly recommended to use absolute notation only. With incremental notation rounding errors can accumulate, losing vital precision. With incremental notation Gerber files are no longer human readable, losing a big advantage. Incremental notation is a simplistic compression technology introduced in the 1950's, when saving a few bytes was of paramount importance and computers were too feeble for proper compression algorithms. Nowadays the few bytes saved are irrelevant. Modern compression methods far outperform this simplistic method, without any loss of accuracy. If the size of the archives is important for you use a strong compression algorithm rather than sacrificing precision and clarity. The advantage of incremental notation is long gone. Its disadvantages remain. Incremental notation is a source of endless confusion. Always use absolute notation.

### 5.1.4 Data Block Format

The syntax for the FS parameter is:

**<FS parameter>: FS(LIT)(All)X<Format>Y<Format>\***

Syntax	Comments
FS	FS for Format Specification
L T	Specifies zero omission mode: L – omit leading zeroes T – omit trailing zeroes
A I	Specifies coordinate values notation: A – absolute notation I – incremental notation
X<Format>Y<Format>	Specifies the format of X and Y coordinate data. The format of X and Y coordinates must be the same! <Format> must be expressed as a number NM where N - number of integer positions in coordinate data (0 <= N <= 7) M - number of decimal positions in coordinate data (0 <= M <= 7)

### 5.1.5 Examples

Syntax	Comments
%FSLAX25Y25*%	Coordinate data has leading zeros omitted. Coordinates are expressed using absolute notation with 2 integer and 5 decimal positions for both axes.

## 5.2 MO – Mode

The MO parameter sets the units used for coordinate data and for sizes parameters or modifiers indicating sizes or coordinates. The units can be either inches or millimeters. This parameter must be used only once, at the beginning of the file.



**Note:** the FS parameter is used to set the format (i.e. number of integer and decimal positions) of the coordinate data.

### 5.2.1 Data Block Format

The syntax for the MO parameter is:

**<MO parameter>: MO(IN|MM)\***

Syntax	Comments
MO	MO for Mode
IN MM	Units of the dimension data: IN – inches MM – millimeters

### 5.2.2 Examples

Syntax	Comments
%MOIN*%	Dimensions are expressed in inches
%MOMM*%	Dimensions are expressed in millimeters

## 5.3 IP – Image Polarity

The IP parameter sets the positive or negative polarity for the entire image. It can only be used once, at the beginning of the file.

### 5.3.1 Positive image polarity


Under *positive* image polarity, the image is generated as specified elsewhere in this document. (In other words, the image generation has been assuming positive image polarity.)

### 5.3.2 Negative image polarity

Under negative image polarity, image generation is different. Its purpose is to create a negative image, clear areas in a dark background. The entire image plane in the background is initially dark instead of clear. The effect of dark and clear polarity is toggled. The entire image is simply reversed, dark becomes white and vice versa.

Note that the first object encountered must have dark polarity, and therefore clears the dark background. It is not allowed to have a clear polarity first object.

Consequently, the first is object always clears the background. It determines the effect of the polarity of all subsequent objects.

 **Note:** Plane layers in PCB's are typically solid copper areas with holes in it, called anti-pads and thermals. For historic reasons, such layers are sometimes transferred as negative images: the copper area is clear and the anti-pads are dark. %IPNEG is a convenient way to create such images. It also clearly specifies the layer is transferred in negative. However, today there is no need to transfer layers in negative. Plane layers are better described positive, using regions (G36/G37) with clear polarity levels (%LPC) to make holes.)

### 5.3.3 Data Block Format

The syntax for the IP parameter is:

**<IP parameter>: IP(POSINEG)\***

Syntax	Comments
IP	IP for Image Polarity
POS	Image has positive polarity
NEG	Image has negative polarity

### 5.3.4 Examples

Syntax	Comments
%IPPOS*%	Image has positive polarity
%IPNEG*%	Image has negative polarity

## 5.4 IN - Image Name

The IN parameter assigns a name to the entire image contained in the Gerber file. The name must comply with the syntax rules described in section 3.2.

This parameter can only be used once, at the beginning of the file.

### 5.4.1 Data Block Format

The syntax for the IN parameter is:

**<IN parameter>: IN<Name>\***

Syntax	Comments
IN	IN for Image Name
<Name>	Image name

## 5.4.2 Examples

Syntax	Comments
%INPANEL_1*%	Image name is 'PANEL_1'

## 5.5 AD - Aperture Definition

The AD parameter assigns a D-code number to an aperture (shape) and sets the aperture as current aperture.

The AD parameter must precede the first use of the assigned aperture. It is recommended to put all AD parameters in the beginning of the file.


A D-code number remains assigned to an aperture until it is re-assigned by a new AD parameter. At any point in the file, the last assignment is valid.

An aperture has a *flash point*. When an aperture is flashed at a given coordinate data, the aperture is positioned in such a way that the flash point coincided with the coordinate data.

There are two kinds of apertures: *standard apertures* and *special apertures*.

The Gerber file format contains a number of standard apertures. The AD parameter assigns a D-code to a standard aperture and defines its parameters, typically sizes. The flash point of a standard aperture is the geometric center of its shape.

Other apertures, called special apertures or macro apertures, can be defined with the AM (Aperture Macro) parameter. These aperture macros are identified by their name. See section 5.6. The AD parameter is also used to assign a D-code to a special aperture and define its parameters. The flash point of a special aperture is the origin of the coordinates used in the AM parameter.

 **Note:** Zero size apertures are valid. An object created with a zero size aperture is a valid object, but it does not affect the image. It can be used to define meta information, e.g. an outline or a reference point.

### 5.5.1 Syntax Rules

The AD parameter starts with 'AD', followed by 'D' and D-code number, then the aperture type and then optionally modifiers.

The allowed range of D-code is from 10 up to 2147483647 (max int32). The D-codes 1 to 9 are reserved and *cannot* be used for apertures.



**Warning:** In older versions of the specification the maximum D-code was 999. Gerber readers may be severely limited in the maximum D code they support. It is therefore recommended to use low D-codes when possible.



**Example:**

%ADD10C, .025\*%



**Note:** For readability it is recommended to enclose each AD parameter into a separate pair of '%' characters.

## 5.5.2 Data Block Format

The syntax for the AD parameter is the following:

<AD parameter>: ADD<D-code number><Aperture type>[,<Modifiers set>]\*

<Modifiers set>: <Modifier>{X<Modifier>}

Syntax	Comments
ADD	AD for Aperture Definition and D for D-code
<D-code number>	The D-code number being defined ( $\geq 10$ )
<Aperture type>[,<Modifiers set>]	The aperture type optionally followed by modifiers

The **<Aperture type>** can be in one of two available formats:

- For a standard aperture: one of the standard aperture codes (C,R,O or P)
- For a special aperture: an aperture macro name previously defined by an AM parameter

The required number of modifiers in <Modifiers set> depends on the <Aperture type>. Modifiers are separated by the 'X' character. All sizes must be  $\geq 0$ . Units follow the MO parameter. The numbers follow standard notation, optionally including a decimal point; they do not follow the FS parameter.

Standard apertures may be solid or open. Open means there is a hole in the aperture. Holes in apertures have no effect on the image, in other words they are transparent.

The syntax of a hole is common for all standard apertures:

**<Hole>:    <X-axis hole size>[<Y-axis hole size>]**

If only the **<X-axis hole size>** modifier is specified the hole is round, and the modifier specifies the diameter. If both X and Y is specified the hole is rectangular and the modifiers specify the X and Y size. If both parameters are omitted the aperture is solid.

The hole must fit within the aperture. It is centered on the aperture.

□

## 5.5.3 Aperture Definition with Standard Apertures

The standard apertures are described below.

### 5.5.3.1 Circle

The syntax of the circle standard aperture:

**C,<Diameter>[X<Hole>]**

Syntax	Comments
C	Indicates that this is a circle aperture
<Diameter>	Circle diameter, ≥0
<Hole>	Optional hole If no hole is specified the aperture is solid



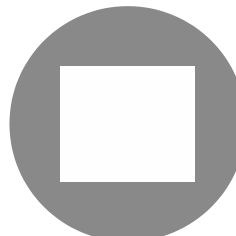
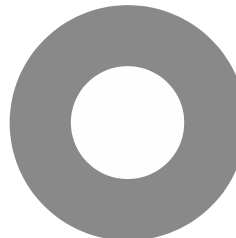
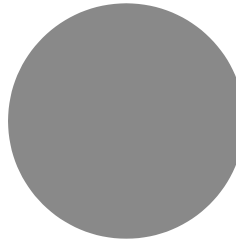
#### Examples:

These statements define the apertures below

```
%ADD10C,0.5*%
```

```
%ADD10C,0.5X0.25*%
```

```
%ADD10C,0.5X0.29X0.29*%
```



*Circles with different holes*

### 5.5.3.2 Rectangle

The syntax of the rectangle or square standard aperture:

**R,<X size>X<Y size>[X<Hole>]**

Syntax	Comments
R	Indicates that this is a rectangle or square aperture
<X size> <Y size>	X and Y sizes of the rectangle sides If the <X size> equals the <Y size>, the aperture is square
<Hole>	Optional hole If no hole is specified the aperture is solid



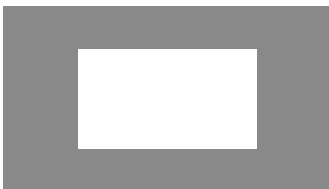
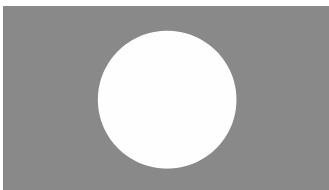
#### Examples:

These statements define the apertures below

```
%ADD22R,0.044X0.025*%
```

```
%ADD22R,0.044X0.025X0.019*%
```

```
%ADD22R,0.044X0.025X0.024X0.013*%
```



*Rectangles with different holes*



### 5.5.3.3 Obround

Obround (oval) is a shape consisting of two semicircles connected by parallel lines tangent to their endpoints. The syntax of the obround standard aperture:

**O,<X size>X<Y size>[X<Hole>]**

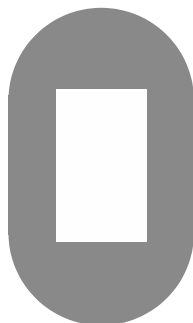
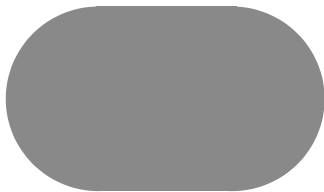
Syntax	Comments
O	Indicates that this is an obround aperture
<X size> <Y size>	X and Y sizes of the obround sides The smallest side is terminated by half a circle. If the <X size> is larger than <Y size>, the shape is horizontal. If the <X size> is smaller than <Y size>, the shape is vertical. If the <X size> is equal to <Y size>, the shape is a circle
<Hole>	Optional hole. If no hole is specified, the aperture is solid



#### Example:

These statements define the apertures below

```
%ADD22O,0.046X0.026*%
%ADD22O,0.046X0.026X0.019*%
%ADD22O,0.026X0.046X0.013X0.022*%
```



#### 5.5.3.4 Regular polygon

The syntax of the polygon standard aperture:

**P,<Outer diameter>X<Number of vertices>[X<Degrees of rotation>[X<Hole>]]**

Syntax	Comments
P	Indicates that this is a polygon aperture
<Outer diameter>	Diameter of the circumscribed circle, i.e. the circle through the polygon vertices
<Number of vertices>	Number of polygon vertices, ranging from 3 to 12
<Degrees of rotation>	Specifies rotation of the aperture around its center. Without rotation one vertex is on the positive X-axis through the center. Rotation angle is expressed in decimal degrees; positive value for counterclockwise rotation, negative value for clockwise rotation.
<Hole>	Optional hole The hole modifiers can be specified only after a rotation angle; set an angle of zero if no rotation is required If no hole is specified the aperture is solid



**Note:** The orientation of the hole is not affected by the rotation angle modifier.



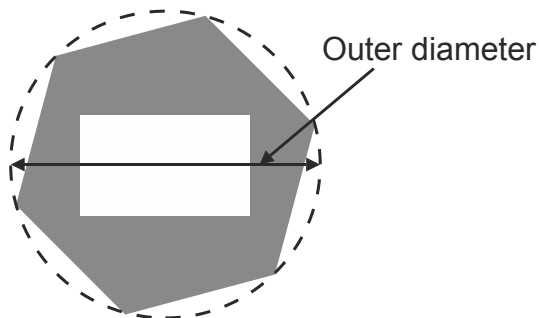
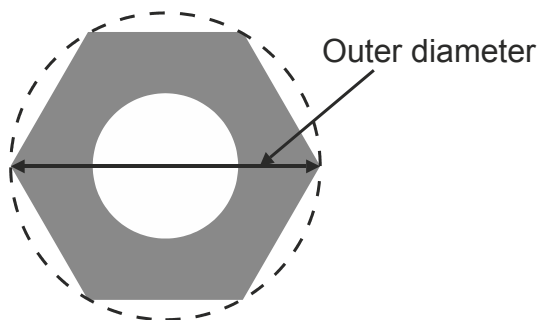
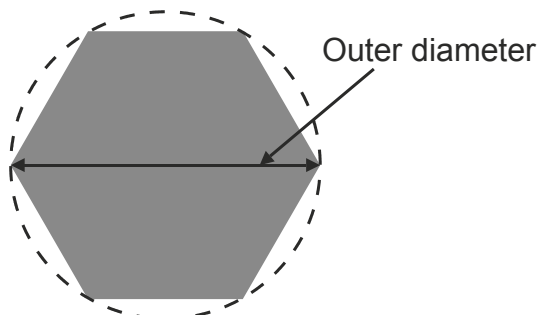
### Examples:

These statements define the apertures below

```
%ADD17P, .040X6*%
```

```
%ADD17P, .040X6X0.0X0.019*%
```

```
%ADD17P, .040X6X15.0X0.023 X0.013*%
```



*Polygons with different holes*

## 5.5.4 Examples

Syntax	Comments
%ADD10C,.025*%	D-code 10 is a solid circle with diameter 0.025
%ADD22R,.050X.050X.027*%	D-code 22 is a square with sides of 0.05 and with a 0.027 diameter round hole
%ADD57O,.030X.040X.015*%	D-code 57 is an obround with sizes 0.03 x 0.04 with 0.015 diameter round hole
%ADD30P,.016X6*%	D-code 30 is a solid polygon with 0.016 outer diameter and 6 vertices
%ADD15CIRC*%	D-code 15 is a special aperture described by aperture macro CIRC defined previously by an aperture macro (AM) parameter

## 5.6 AM - Aperture Macro

The AM parameter defines special apertures consisting of building blocks called primitives.

A special aperture macros defined by the AM parameter can be referenced from AD parameter definitions in the same way as the standard apertures. (One could view the standard apertures as pre-defined macro apertures.) A special aperture must be defined before a D-code number can be assigned to it.

Special apertures offer two advantages compared to standard apertures:

- Multiple shapes called primitives can be combined in a single aperture to create non-standard aperture shapes
- Aperture macro modifiers can be variable; the actual values can be defined as:
  - Values provided by an AD parameter referencing the aperture macro
  - Arithmetic expressions calculating the actual value based on other variables

The AM parameter can be used multiple times in a file. It must be put at the beginning of a file or level.

### 5.6.1 Data Block Format

The syntax for the AM parameter is:

**<AM parameter>:           AM<Aperture macro name>\*<Macro content>**  
**<Macro content>:    {{{<Variable definition>}\*}{<Primitive>}\*}}**  
**<Variable definition>:       \$K=<Arithmetic expression>**  
**<Primitive>:           <Primitive code>,<Modifier>{,<Modifier>}|<Comment>**  
**<Modifier>:            \$MI< Arithmetic expression>**  
**<Comment>:            0 <Text>**

Syntax	Comments
AM	AM for Aperture Macro


<Aperture macro name>	Name of the aperture macro. See 3.2 for the syntax rules.
<Macro content>	Macro content describes primitives included into the aperture macro. Can also contain definitions of new variables.
<Variable definition>	Definition of a variable.
\$K=<Arithmetic expression>	Definition of the variable \$K. An arithmetic expression may use arithmetic operators (described later), constant numbers and also variables \$X where the definition of \$X precedes \$K.
<Primitive>	Individual primitive that includes primitive code and modifiers which are primitive parameters (e.g. diameter for a circle). The modifiers depend on the primitive. All primitives are described later in this document.
<Primitive code>	A code specifying the primitive (e.g. polygon).
<Modifier>	Modifier can be a decimal number (e.g. 0.050), a variable (e.g. \$1) or an arithmetic expression based on numbers and variables. The actual value for a variable is either provided by an AD parameter or defined within the AM by some previous <Variable definition>.
<Comment>	Comment does not affect the image.
<Text>	Single-line text string

Modifiers can be classified as in the following table:

Modifier type	Comments
Exposure modifier	The exposure modifier that can have the following values: 0 = exposure is 'off' 1 = exposure is 'on'
Rotation modifier	Modifier that specifies the rotation angle: positive value means counterclockwise rotation, negative - clockwise.
Geometry modifier	Modifier that specifies e.g. a diameter, X and Y positions, width, etc.

Coordinates and sizes are expressed in the unit set by the MO parameter.

Exposure is set with the exposure modifier. Exposure can be 'on' or 'off'. Exposure 'on' creates a solid part of the macro aperture. Exposure 'off' clears or erases the underlying solid part to create a hole in it. Note that the clearing action only affects the macro aperture itself and is not passed through to affect the image that the macro aperture is used on. Exposure off only creates a hole in the macro aperture, and a hole has no effect on the image.

 **Warning:** A hole is transparent. One sees the objects below it. This is not the same as clear level polarity, where all objects below are cleared or erased.

The rotation angle is expressed in degrees, by a decimal number; a positive value means counterclockwise rotation, a negative value means clockwise rotation. The pivot of the rotation of a primitive is always the origin, i.e. the point (0,0). To rotate a macro composed of several

primitives it is then sufficient to rotate all primitives with the same angle. Note that for the polygon, thermal and moiré rotation is only allowed if their center is placed on the origin.

## 5.6.2 Primitives

### 5.6.2.1 Comment

The comment primitive has no image meaning. It is used to include human-readable comments into the AM parameter. The comment primitive starts with the '0' code (zero) followed by a space and then a single-line text string. The text string follows the syntax rules for comments as described in section 3.1.



#### Example:

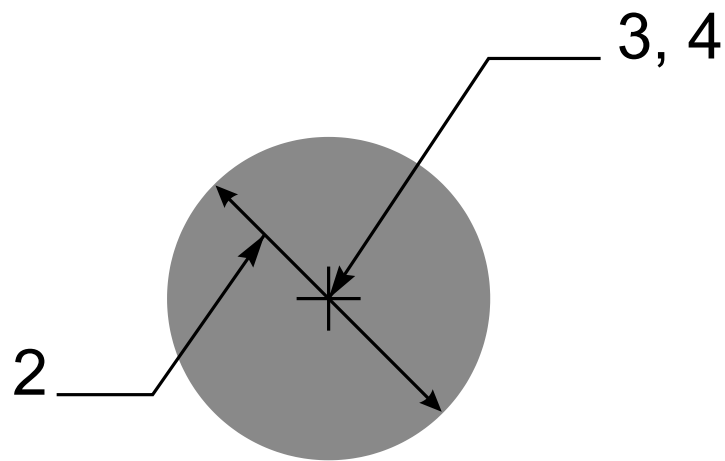
```
%AMRECTROUNDCORNERS*
0 Rectangle with rounded corners. *
0 Offsets $4 and $5 are interpreted as the *
0 offset of the flash origin from the pad center. *
0 First create horizontal rectangle. *
21,1,$1,$2-$3-$3,0-$4,0-$5,0*
0 From now on, use width and height half-sizes. *
$9=$1/2*
$8=$2/2*
0 Add top and bottom rectangles. *
22,1,$1-$3-$3,$3,0-$9+$3-$4,$8-$3-$5,0*
22,1,$1-$3-$3,$3,0-$9+$3-$4,0-$8-$5,0*
0 Add circles at the corners. *
1,1,$3+$3,0-$4+$9-$3,0-$5+$8-$3*
1,1,$3+$3,0-$4-$9+$3,0-$5+$8-$3*
1,1,$3+$3,0-$4-$9+$3,0-$5-$8+$3*
1,1,$3+$3,0-$4+$9-$3,0-$5-$8+$3*%
```

In the example above all the lines starting with 0 are comments and do not affect the image.

### 5.6.2.2 Circle

A circle primitive is defined by its center point and diameter. The **<Primitive code>** of a circle primitive is '1'.

Modifier number	Description
1	Exposure off/on (0/1)
2	Diameter
3	X coordinate of center position
4	Y coordinate of center position



*Circle primitive*



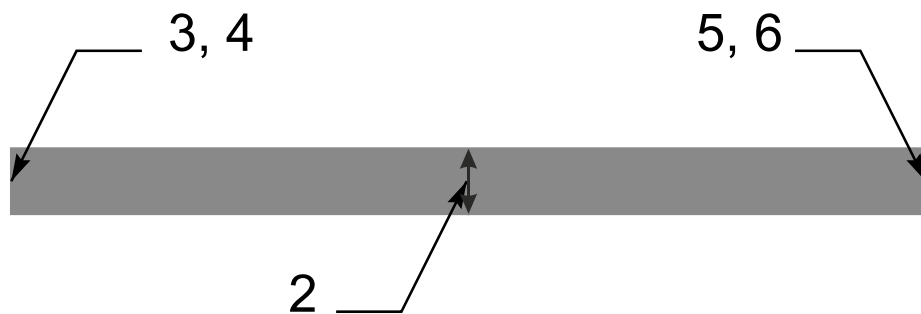
#### Example:

```
%AMCIRCLE*  
1,1,1.5,0,0*%
```

### **Line (vector)**

A line (vector) primitive is a rectangle defined by its line width, start and end points. The line ends are rectangular. The **<Primitive code>** of a line (vector) primitive is '2' or '20'.

Modifier number	Description
1	Exposure off/on (0/1)
2	Line width
3	X coordinate of start point
4	Y coordinate of start point
5	X coordinate of end point
6	Y coordinate of end point
7	Rotation angle around the <i>origin</i> (rotation is <i>not</i> around the center of the object)



### *Line (vector) primitive*



#### **Example:**

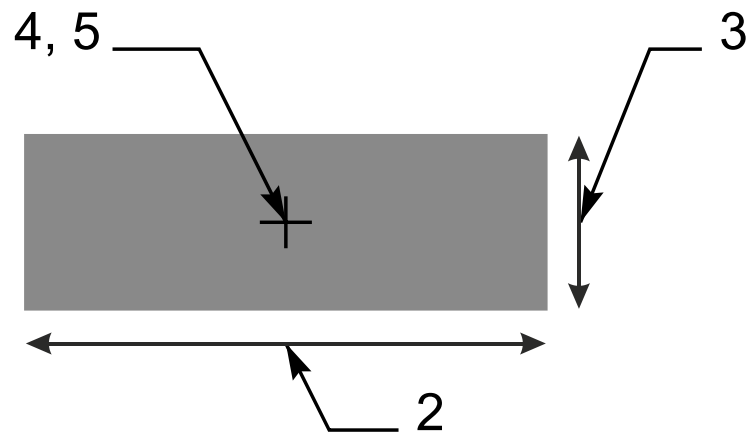
```
%AMLIN*20,1,0.9,0,0.45,12,0.45,0*%
```



### 5.6.2.3 Line (center)

A line (center) primitive is a rectangle defined by its width, height, and center point. The **<Primitive code>** of a line (center) primitive is '21'.

Modifier number	Description
1	Exposure off/on (0/1))
2	Rectangle width
3	Rectangle height
4	X coordinate of center point
5	Y coordinate of center point
6	Rotation angle around the <i>origin</i> (rotation is <i>not</i> around the center of the object)



*Line (center) primitive*



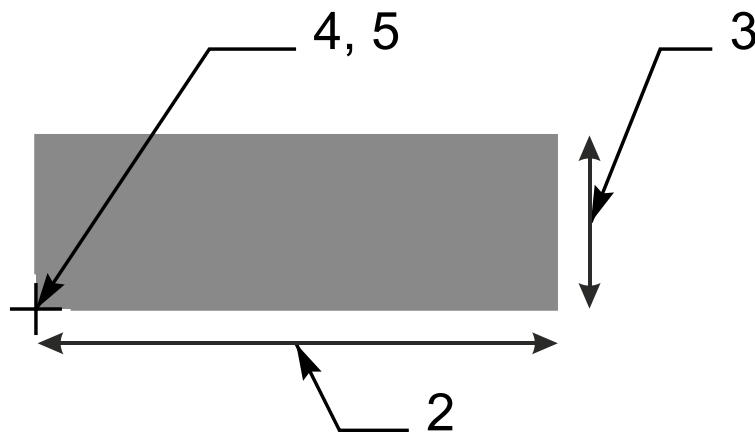
**Example:**

```
%AMRECTANGLE*21,1,6.8,1.2,3.4,0.6,0*%
```

#### 5.6.2.4 Line (lower left)

A line (lower left) primitive is a rectangle defined by its width, height, and the lower left point. The **<Primitive code>** of a line (lower left) primitive is '22'.

Modifier number	Description
1	Exposure off/on (0/1))
2	Rectangle width
3	Rectangle height
4	X coordinate of lower left point
5	Y coordinate of lower left point
6	Rotation angle around the <i>origin</i> (rotation is <i>not</i> around the center of the object)



*Line (lower left) primitive*



**Example:**

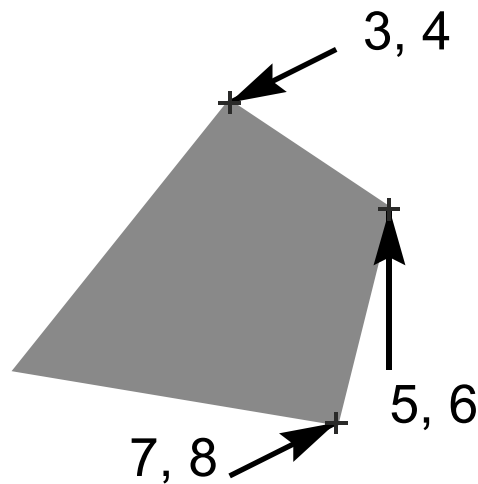
```
%AMLINE2*22,1,6.8,1.2,0,0,0*%
```

#### 5.6.2.5 Outline

An outline primitive is an area enclosed by an n-points polygon, defined by its start point and n subsequent points. The outline must be closed, i.e. the last point must be equal to the start point. The minimum number of subsequent points is 2. Self-intersecting outlines are not allowed. The **<Primitive code>** of an outline primitive is '4'.

Modifier number	Description
1	Exposure off/on (0/1)
2	The number of subsequent points n

3, 4	X and Y coordinates of the start point
5, 6	X and Y coordinates of subsequent point number 1
...	X and Y coordinates of further subsequent points
3+2n, 4+2n	X and Y coordinates of subsequent point number n. Must be equal to coordinates of start point
5+2n	Rotation angle around the <i>origin</i> (rotation is <i>not</i> around the center of the object)



*Outline primitive*

The X and Y coordinates are not modal, both the X and the Y coordinate must be specified for all points.



**Note:** Older versions of the specification defined the maximum of 50 for the number of subsequent points n. This has proven to be too restrictive, and the limit is now increased to 4000.



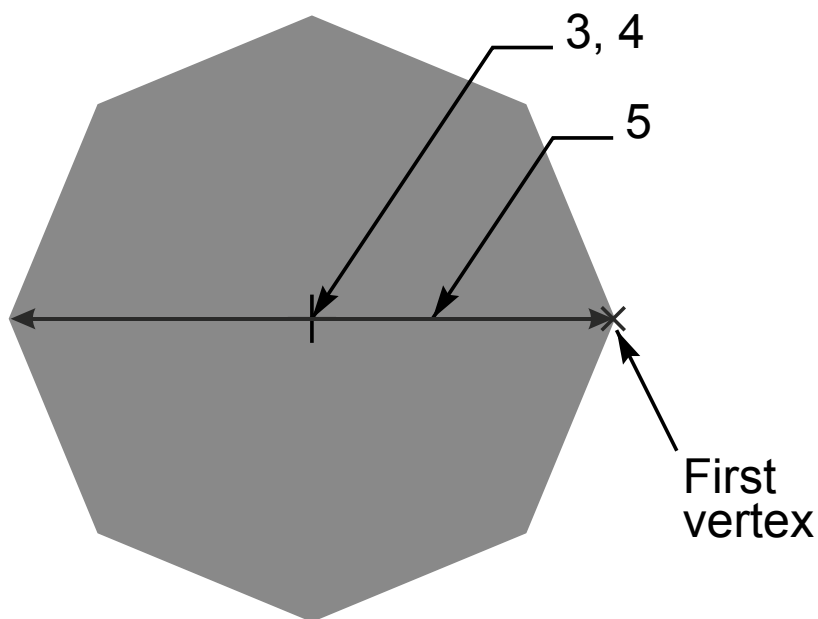
**Example:**

```
%AMOUTLINE*
4,1,4,
0.1,0.1,
0.5,0.1,
0.5,0.5,
0.1,0.5,
0.1,0.1,
0*%
```

### 5.6.2.6 Polygon

A polygon primitive is a regular polygon defined by the number of vertices  $n$ , the center point and the diameter of the circumscribed circle. The **<Primitive code>** of a polygon primitive is '5'.

Modifier number	Description
1	Exposure off/on (0/1)
2	Number of vertices $n$ , $3 \leq n \leq 12$
3	X coordinate of center point
4	Y coordinate of center point
5	Diameter of the circumscribed circle
6	Rotation angle around the <i>origin</i> (rotation is <i>not</i> around the center of the object). Rotation is only allowed if the center point is on the origin.



*Polygon primitive*

Without rotation, the first vertex is on the positive X-axis through the center point.



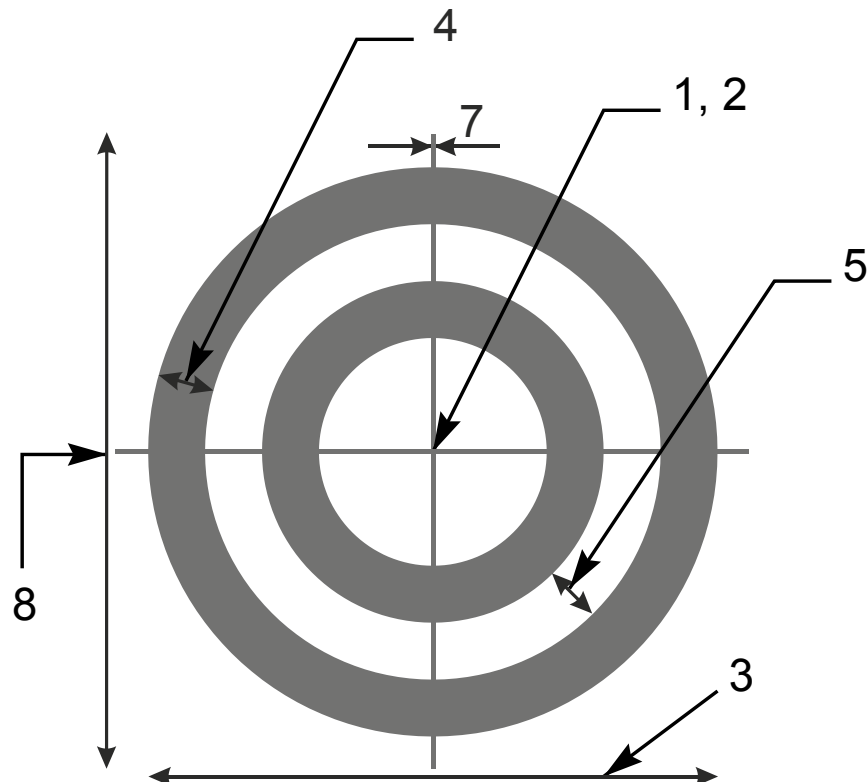
**Example:**

```
%AMPOLYGON*
5,1,8,0,0,8,0*%
```

### 5.6.2.7 Moiré

The moiré primitive is defined as a cross hair centered on concentric rings (annuli). The **<Primitive code>** of a moiré primitive is '6'. Exposure is always on.

Modifier number	Description
1	X coordinate of center point
2	Y coordinate of center point
3	Outer diameter of outer concentric ring
4	Ring thickness
5	Gap between rings
6	Maximum number of rings
7	Cross hair thickness
8	Cross hair length
9	Rotation angle around the <i>origin</i> (rotation is <i>not</i> around the center of the object). Rotation is only allowed if the center point is on the origin.



*Moiré primitive*

The outer diameter of the outer ring is specified by modifier 3. The ring has the thickness defined by modifier 4. Moving further towards the center there is a gap defined by modifier 5, and then the second ring etc. The maximum number of rings is defined by modifier 6. The

number of rings can be less if the center is reached. If there is not enough space for the last ring it becomes a full disc centered on the origin.



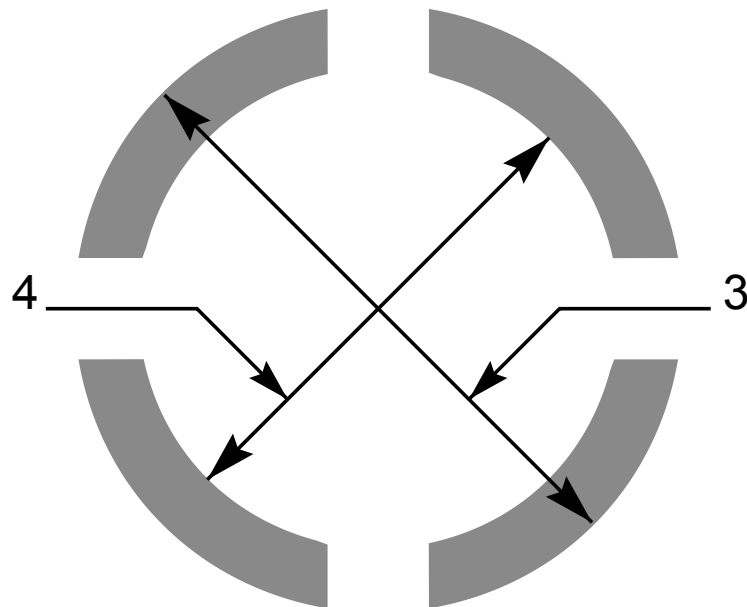
**Example:**

```
%AMMOIRE*  
6,0,0,5,0.5,0.5,2,0.1,6,0*%
```

### 5.6.2.8 Thermal

The thermal primitive is defined as a ring (annulus) interrupted by four gaps. The **<Primitive code>** of a thermal primitive is '7'. Exposure is always on.

Modifier number	Description
1	X coordinate of center point
2	Y coordinate of center point
3	Outer diameter
4	Inner diameter
5	Gap thickness
6	Rotation angle around the <i>origin</i> (rotation is <i>not</i> around the center of the object). Rotation is only allowed if the center point is on the origin.



*Thermal primitive*

Without rotation, the gaps are on the X and Y axes through the center.



**Note:** Modifier 5 must be smaller than modifier 3. The inner circle disappears when modifier 5 is greater than modifier 4.

### 5.6.3 Parameter Contents

An aperture macro definition must contain an aperture macro name that later can be referenced from an AD parameter. An aperture macro definition also contains one or more of the aperture primitives described above. Each primitive (besides the special comment primitive) includes modifiers setting exposure, position, size, rotation etc. Primitive modifiers can use variables. To indicate variables the special character '\$' is used. The values for such variables must either be provided by an AD parameter or calculated using an arithmetic expression over other variables.

### 5.6.4 Syntax Rules

Each AM definition must be enclosed into a separate pair of '%' characters.



**Warning:** An AM definition cannot be grouped. This is different from the other parameters.

As an AM definition can be quite long, it can contain line separators to enhance readability. Line separators are ignored within an AM definition.

An AM parameter can contain the following types of data blocks:

- ☐ AM declaration
- ☐ Primitive with modifiers
- ☐ Variable definition by an arithmetic expression
- ☐ Comment (special comment primitive)

Each data block must end with the end-of-block '\*' character. Within a primitive data block each modifier must be separated by a comma. A modifier can be one of the following:

- ☐ A decimal number, such as 0, 1, 2, or 9.05
- ☐ A variable, such as \$1 or \$VAR
- ☐ An arithmetic expression including numbers and other variables.

#### 5.6.4.1 Variable values from an AD Parameter

An AM parameter can use variables, whose actual values can be provided by an AD parameter. Such variables are identified by '\$n' where n indicates the serial number of the variable value in the list provided by an AD parameter. Thus \$1 means the first value in the list, \$2 the second, and so on.



**Example:**

```
%AMDONUTVAR*1, 1, $1, $2, $3*1, 0, $4, $2, $3*%
```

Here the variables \$1, \$2, \$3 and \$4 are used as modifier values. The corresponding AD parameter should look like:

```
%ADD34DONUTVAR, 0.100X0X0X0.080*%
```

In this case the value of variable \$1 becomes 0.100, \$2 and \$3 become 0 and \$4 becomes 0.080. These values are used as values of corresponding modifiers in the DONUTVAR macro.

#### 5.6.4.2 Arithmetic expressions

A modifier value can also be defined as an arithmetic expression that includes basic arithmetic operators such as 'add' or 'multiply', constant numbers (with or without decimal point) and other variables. The following arithmetic operators can be used:

Operator	Function
+	Add
-	Subtract
x (lowercase)	Multiply
/	Divide

### *Arithmetic operators*

The result of the divide operation is decimal; it is not rounded or truncated to an integer.

The standard arithmetic precedence rules apply. Below operators are listed in order from lowest to highest priority. The brackets '(' and ')' can be used to overrule the standard precedence rules.

- Add and subtract: '+' and '-'
- Multiply and divide: 'x' and '/'
- Brackets: '(' and ')'

There is no unary minus operator. Negative values can be expressed by subtracting from zero, e.g. '0-\$1'.



#### **Example:**

```
%AMRECT*
21, 1, $1, $2-$3-$3, 0-$4, 0-$5, 0*%
```

Corresponding AD parameter could look like:

```
%ADD146RECT, 0.0807087X0.1023622X0.0118110X0.5000000X0.3000000*%
```

### **5.6.4.3 Definition of a new variable**

The AM parameter allows defining new variables based on previously defined variables. A new variable is defined as an arithmetic expression that follows the same rules as described in previous section. A variable definition also includes '=' sign with the meaning 'assign'.

For example, to define variable \$4 as a variable \$1 multiplied by 0.75 the following arithmetic expression can be used:

```
$4=$1x0.75
```



#### **Example:**

```
%AMDONUTCAL*
1, 1, $1, $2, $3*
$4=$1x0.75*
```



1,0,\$4,\$2,\$3\*%

Local variables with symbolic names, e.g. \$XSIZE, \$YSIZE, are allowed. Symbolic variable names are subject to the syntax rules described in section 3.2.



**Example:**

%AMREC2\*\$XSIZE=\$1\*\$YSIZE=\$2\*21,1,\$XSIZE,\$YSIZE,0,0,0\*%

Here local variables \$XSIZE and \$YSIZE are defined and initialized to \$1 and \$2 values.

The values for variables in an AM parameter are determined as follows:

- ❑ All variables used in AM parameter are initialized to 0
- ❑ If an AD parameter that references the aperture macro contains N modifiers then variables \$1,\$2, ..., \$N get the values of these modifiers
- ❑ The remaining variables get their values from definitions in the AM parameter; if some variable remains undefined then its value is still 0
- ❑ The values of variables \$1, \$2, ..., \$N can also be modified by definitions in AM, i.e. the values originating from an AD parameter can be redefined



**Example:**

%AMDONUTCAL\*1,1,\$1,\$2,\$3\*\$4=\$1x0.75\*1,0,\$4,\$2,\$3\*%

The variables \$1, \$2, \$3, \$4 are initially set to 0.

If the corresponding AD parameter contains only 2 modifiers then the value of \$3 will remain 0.

If the corresponding AD parameter contains 4 modifiers. e.g.

%ADD35DONUTCAL,0.020X0X0X0.03\*%

the variable values are calculated as follows: the AD parameter modifier values are first assigned so variable values \$1 = 0.02, \$2 = 0, \$3 = 0, \$4 = 0.03. The value of \$4 is modified by definition in AM parameter so it becomes \$4 = 0.02 x 0.75 = 0.015.

The variable definitions and primitives are handled from the left to the right in the order of AM parameter. This means a variable can be set to a value, used in a primitive, re-set to a new value, used in a next primitive etc.



**Example:**

%AMTARGET\*1,1,\$1,0,0\*\$1=\$1x0.8\*1,0,\$1,0,0\*\$1=\$1x0.8\*1,1,\$1,0,0\*\$1=\$1x0.8\*1,0,\$1,0,0\*\$1=\$1x0.8\*1,1,\$1,0,0\*\$1=\$1x0.8\*1,0,\$1,0,0\*%  
%ADD37TARGET,0.020\*%



Here the value of \$1 is changed by the expression '\$1=\$1x0.8' after each primitive so the value changes like the following: 0.020, 0.016, 0.0128, 0.01024, 0.008192, 0.0065536.



**Example:**

```
%AMREC1*$2=$1*$1=$2*21,1,$1,$2,0,0,0*%
%AMREC2*$1=$2*$2=$1*21,1,$1,$2,0,0,0*%
%ADD51REC1,0.02,0.01*%
%ADD52REC2,0.02,0.01*%
```

Aperture 51 is the square with side 0.02 and aperture 52 is the square with side 0.01, because the variable values in AM parameters are calculated as follows:

For the aperture 51 initially \$1 is 0.02 and \$2 is 0.01. After operation '\$2=\$1' the variable values become \$2 = 0.02 and \$1 = 0.02. After the next operation '\$1=\$2' they remain \$2 = 0.02 and \$1 = 0.02 because previous operation changed \$2 to 0.02. The resulting primitive has 0.02 width and height.

For the aperture 52 initially \$1 is 0.02 and \$2 is 0.01 (the same as for aperture 51). After operation '\$1=\$2' the variable values become \$1 = 0.01 and \$2 = 0.01. After the next operation '\$2=\$1' they remain \$1 = 0.01 and \$2 = 0.01 because previous operation changed \$1 to 0.01. The resulting primitive has 0.01 width and height.

Below are some more examples of using arithmetic expressions in AM parameter. Note that some of these examples probably do not represent a reasonable aperture macro – they just illustrate the syntax that can be used for defining new variables and modifier values.



**Example:**

```
%AMTEST*
1,1,$1,$2,$3*
$4=$1x0.75*
$5=($2+100)x1.75*
1,0,$4,$5,$3*%

%AMTEST*
$4=$1x0.75*
$5=100+$3*
1,1,$1,$2,$3*
1,0,$4,$2,$5*
$6=$4x0.5*
1,0,$6,$2,$5*%

%AMRECTROUND CORNERS*
21,1,$1,$2-$3-$3,0-$4,0-$5,0*
$9=$1/2*
$8=$2/2*
22,1,$1-$3-$3,$3,0-$9+$3-$4,$8-$3-$5,0*
22,1,$1-$3-$3,$3,0-$9+$3-$4,0-$8-$5,0*
1,1,$3+$3,0-$4+$9-$3,0-$5+$8-$3*
1,1,$3+$3,0-$4-$9+$3,0-$5+$8-$3*
1,1,$3+$3,0-$4-$9+$3,0-$5-$8+$3*
1,1,$3+$3,0-$4+$9-$3,0-$5-$8+$3*%
```

## 5.6.5 Examples

### 5.6.5.1 Fixed Modifier Values

The following AM parameter defines an aperture macro named 'DONUTFIX' consisting of two concentric circles with fixed diameter sizes:

```
%AMDONUTFIX*1,1,0.100,0,0*1,0,0.080,0,0*%
```

Syntax	Comments
AMDONUTFIX	Aperture macro name is 'DONUTFIX'
1,1,0.100,0,0	1 – Circle 1 – Exposure on 0.100 – Diameter 0 – X coordinate of the center 0 – Y coordinate of the center
1,0,0.080,0,0	1 – Circle 0 – Exposure off 0.080 – Diameter 0 – X coordinate of the center 0 – Y coordinate of the center

The AD parameter using this aperture macro can look like the following:

```
%ADD33DONUTFIX*%
```

### 5.6.5.2 Variable Modifier Values

The following AM parameter defines an aperture macro named 'DONUTVAR' consisting of two concentric circles with variable diameter sizes:

```
%AMDONUTVAR*1,1,$1,$2,$3*1,0,$4,$2,$3*%
```

Syntax	Comments
AMDONUTVAR	Aperture macro name is 'DONUTVAR'
1,1,\$1,\$2,\$3	1 – Circle 1 – Exposure on \$1 – Diameter is provided by AD parameter \$2 – X coordinate of the center is provided by AD parameter \$3 – Y coordinate of the center is provided by AD parameter

1,0,\$4,\$2,\$3	<p>1 – Circle</p> <p>0 – Exposure off</p> <p>\$4 – Diameter is provided by AD parameter</p> <p>\$2 – X coordinate of the center is provided by AD parameter (same as in first circle)</p> <p>\$3 – Y coordinate of the center is provided by AD parameter (same as in first circle)</p>
-----------------	---

The AD parameter using this aperture macro can look like the following:

```
%ADD34DONUTVAR,0.100X0X0X0.080*%
```

In this case the variable modifiers get the following values: \$1 = 0.100, \$2 = 0, \$3 = 0, \$4 = 0.080.

### 5.6.5.3 Definition of a New Variable

The following AM parameter defines an aperture macro named 'DONUTCAL' consisting of two concentric circles with the diameter of the second circle defined as a function of the diameter of the first:

```
%AMDONUTCAL*1,1,$1,$2,$3*$4=$1x0.75*1,0,$4,$2,$3*%
```

Syntax	Comments
AMDONUTCAL	Aperture macro name is 'DONUTCAL'
1,1,\$1,\$2,\$3	<p>1 – Circle</p> <p>1 – Exposure on</p> <p>\$1 – Diameter is provided by AD parameter</p> <p>\$2 – X coordinate of the center is provided by AD parameter</p> <p>\$3 – Y coordinate of the center is provided by AD parameter</p>
\$4=\$1x0.75	Defines variable \$4 to be used as the diameter of the inner circle; the diameter of this circle is 0.75 times the diameter of the outer circle
1,0,\$4,\$2,\$3	<p>1 – Circle</p> <p>0 – Exposure off</p> <p>\$4 – Diameter is calculated using the previous definition of this variable</p> <p>\$2 – X coordinate of the center is provided by AD parameter (same as in first circle)</p> <p>\$3 – Y coordinate of the center is provided by AD parameter (same as in first circle)</p>

The AD parameter using this aperture macro can look like the following:

```
%ADD35DONUTCAL,0.020X0X0*%
```

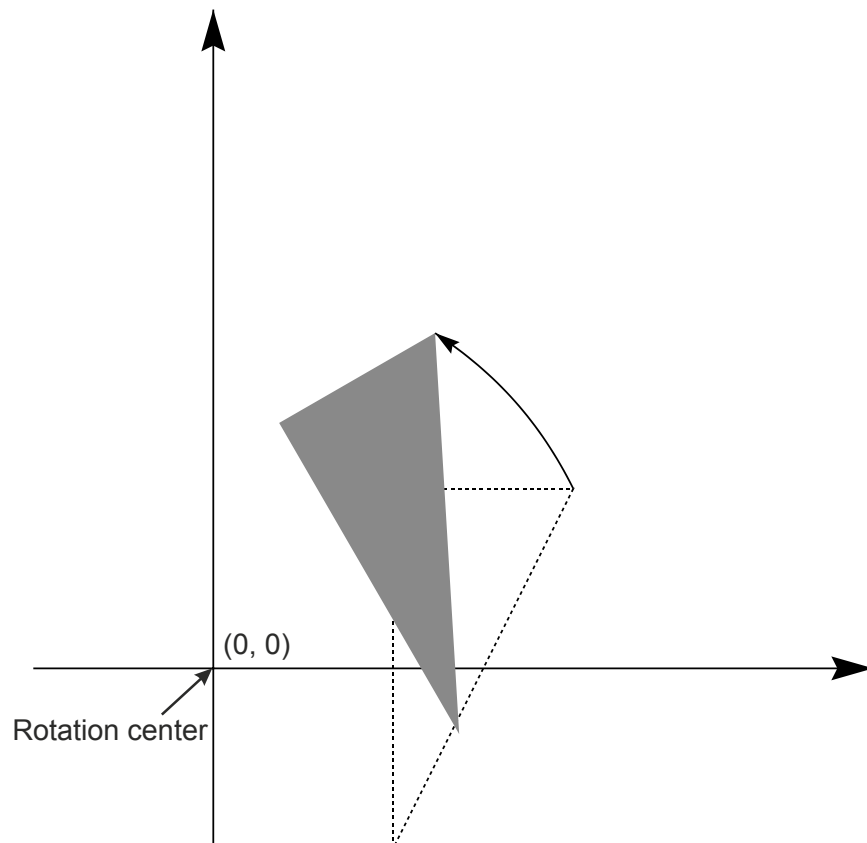
This defines a donut with outer circle diameter equal to 0.02 and inner circle diameter equal to 0.015.

#### 5.6.5.4 Rotation Modifier

The following AM parameter defines an aperture macro named 'TRIANGLE\_30'. The macro is a triangle rotated 30 degrees around the origin:

```
%AMTRIANGLE_30*4,1,3,1,-1,1,1,2,1,1,-1,30*%
```

Syntax	Comments
AMTRIANGLE_30	Aperture macro name is 'TRIANGLE_30'
4,1,3	4 – Outline 1 – Exposure on 3 – The outline has three subsequent points
1,-1	1 – X coordinate of the start point -1 – Y coordinate of the start point
1,1,2,1,1,-1	Coordinates (X, Y) of the subsequent points: (1,1), (2,1), (1,-1)
30	Rotation angle is 30 degrees counterclockwise



*Rotated triangle*

The AD parameter using this aperture macro can look like the following:

```
%ADD33AMTRIANGLE_30*%
```

## 5.7 SR – Step and Repeat

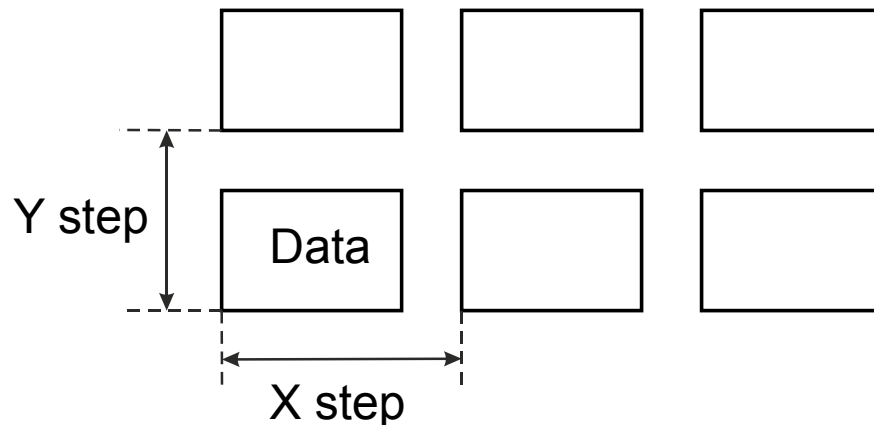
The SR parameter sets the graphics state variable 'Step & Repeat' to a number of repeats with a step distance along the X and Y axis.

Objects generated are collected in a data set. When another SR parameter or the end of the file is encountered, the data set is stepped and repeated (copied) in the image plane. The number of the repeats and the step distance is determined by the values in the 'Step & Repeat' graphics state variable.



**Example:**

%SRX3Y2I5.0J4.0\*%



A step & repeat data set can contain multiple levels with different polarities.

The step must be greater than the data size (the enclosing rectangle of all objects in the data). Consequently objects generated simultaneously under step & repeat never overlap. This is necessary to avoid ambiguities about the order of overlapping objects, which affects image generation.

The number of repeats and the steps can be different for the X and Y axes.

The number of repeats along an axis can be 1, which is equivalent to no repeat. The corresponding step is then irrelevant. It is recommended to set the step to 0.

The SR parameter can be used multiple times in a file. An SR parameter remains effective till superseded by a new SR parameter.

### 5.7.1 Data Block Format

The syntax for the SR parameter is:

**<SR parameter>: SR[X<Repeats>][Y<Repeats>][I<Step>][J<Step>]\***

Syntax	Comments
SR	SR for Step and Repeat
X<Repeats>	Defines the number of times the data is repeated along the X axis. If missing defaults to 1. If present must be a strictly positive integer.

Y<Repeats>	Defines the number of times the data is repeated along the Y axis. If missing defaults to 1. If present must be a strictly positive integer.
I<Step>	Defines the step distance along the X axis. It is mandatory if the number of repeats along X axis is >1. The step is a non-negative decimal number, expressed in the unit specified by %MO.
J<Step>	Defines the step distance along the Y axis. It is mandatory if the number of repeats along X axis is >1. The step is a non-negative decimal number, expressed in the unit specified by %MO.

## 5.7.2 Examples


Syntax	Comments
%SRX2Y3I2.0J3.0*%	Repeat the data 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*%	Repeat the image 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*%	Repeat the data 1 time along the X and Y axes, i.e. data is not repeated.
%SRX1Y1*%	Equivalent to the above.

## 5.8 LP – Level Polarity

The LP parameter starts a new level and sets its polarity to either dark or clear. The level polarity applies to all data following the LP parameter until superseded by another LP parameter.

This parameter can be used multiple times in a file.

Starting a new level resets the current point.

 **Warning:** *Level polarity* is not the same as *image polarity* (see the IP parameter for the image polarity description).

### 5.8.1 Data Block Format

The syntax for the LP parameter is:

**<LP parameter>: LP(CID)\***

Syntax	Comments
LP	LP for Level Polarity
C D	Polarity:

	C – clear polarity D – dark polarity
--	---

## 5.8.2 Examples

Syntax	Comments
%LPD*%	Start a new level with dark polarity
%LPC*%	Start a new level with clear polarity

## 5.9 LN – Level Name

The LN parameter assigns a name to the current level. It has *no* effect on the image. Its purpose is to make the file more easy to read for humans.

The name must comply with the syntax rules in section 3.2. This parameter can be used multiple times in a file.

### 5.9.1 Data Block Format

The syntax for the LN parameter is:

**<LN parameter>: LN<Name>\***

Syntax	Comments
LN	LN for Level Name
<Name>	Level name

## 5.9.2 Examples

Syntax	Comments
%LNVia_anti-pads*%	The name 'Via_anti-pads' is assigned to the current level.





## 6 Reported Semantic Errors

Some non-conforming Gerber implementations create invalid Gerber files, or – worse – valid Gerber files representing unintended images. The table below lists the most common errors encountered in Gerber files.

Symptom	Cause and Correct Usage
Full circles appear or disappear unexpectedly.	The file contains arcs but no G74 or G75. This is invalid. <b>A G74 or G75 is mandatory if arcs are used.</b> See 4.3.7.
Rotated aperture macros using primitive 21 look strange.	Some CAD systems incorrectly assume that primitive 21 rotates around its center while. This is wrong, it rotates around the origin. See 5.6.2.3.
Unexpected image after an aperture change or a D03.	Coordinates without explicit operation code (D01/D02/D03) are used. This is deprecated because it can result in confusion about which operation code is to use. <b>Always explicitly include the operation code (D01/D02/D03) with the coordinate data.</b> See 9.1.
Objects unexpectedly appear or disappear under holes in standard apertures.	Some CAD systems incorrectly assume the hole in an aperture <i>clears</i> (erases) the underlying objects. This is wrong, the hole is not part of the aperture, it is transparent and has no effect on the underlying image. See 5.5.2.
Objects unexpectedly appear or disappear under holes in macro apertures.	Some CAD systems incorrectly assume that exposure off in a macro definition <i>clears</i> (erases) the underlying objects under the aperture flash.. This is wrong, exposure off creates a hole in the resulting aperture, and that hole has no effect on the image. See 5.6.1.
Polygons are smaller than expected.	Some CAD systems incorrectly assume the parameter of a Regular Polygon specifies the inside diameter. This is wrong, it specifies the outside diameter. See 5.5.3.4.
A single Gerber file contains more than one image, separated by M00, M01 or M02.	This is invalid. A Gerber file can contain only one image. One file, one image. One image, one file.
Contour fill defined in a Level Polarity Clear (%LPC) erases a previously defined object at that location where this is not intended.	Contour fill as any other object in an LPC section indeed Clear does not mean transparent. See 2.2.2.
The M1 parameter is used to mirror a macro definition but the result is not as expected.	With the M1 parameter mirroring is not applied to aperture definitions. Do not use this deprecated parameter. Apply the transformation directly in the aperture definition and object coordinates.

### *Reported Semantic Errors*



## 7 Poor/Good Practice

Some Gerber files are syntactically correct but are needlessly cumbersome to work with or error-prone. The table below summarizes common poor practices and gives the corresponding good practice.

Poor Practice	Problems	Good Practice
Low resolution (numerical precision).	Poor registration of objects between PCB layers. Loss of accuracy. It can result in contours that self-intersect, to invalid arcs, zero-arcs, with unexpected results downstream. Note that software processing the file unavoidably adds further numerical rounding, aggravating the problem.	<b>Always use at least 6 decimal digits.</b> Do not sacrifice precision to save a few bytes.
Multi quadrant mode and rounding errors.	For a very small arc the start and end point can happen to move on top of one another due to rounding. Under G75 mode the small arc suddenly becomes a full circle. Under G74 it remains small.	<b>Use G74</b> single quadrant mode unless you are very careful with rounding on small arcs.
Imprecisely positioned center point of arcs	An imprecisely positioned center makes the arc fuzzy, or even open to very different interpretations. This can lead to unexpected results. By positioning the center imprecisely the creator of the file accepts any possible interpretation.	<b>Always position the center point precisely.</b>
Painted or stroked pads.	Painted pads produce the correct image but are very awkward and time consuming for CAM software, e.g. for DRC checks, electrical test and so on. Stroking was needed for vector photoplotters in the 1960's and 1970's, but these devices are as outdated as the mechanical typewriter.	<b>Always use flashed pads.</b> Define pads, including SMD pads, with the AD and AM parameters.
Painted or stroked areas.	Painted areas produce the correct image, but the files are needlessly large and the data is very confusing for CAM software. Stroking was needed for vector photoplotters in the 1960's and 1970's, but these devices are as outdated as the mechanical typewriter.	<b>Always use contours (G36/G37) to define areas.</b>
Standard Gerber or RS-274-D	Standard Gerber is deprecated. It was designed for a workflow that is as obsolete as the mechanical typewriter. It requires manual labor to process. It is not suitable for today's image exchange. Do not use it.	<b>Always use Extended Gerber.</b>

*Poor/good practices*



## 8 Glossary

**ABSOLUTE POSITION:** Position expressed in Cartesian coordinates from the origin (0, 0).

**APERTURE:** A shape that is used for stroking or flashing. (The name is historic; vector photoplotters exposed images on lithographic film by shining light through an opening, called aperture.)

**APERTURE MACRO:** A macro describing the geometry of a special aperture. This macro is defined by an AM parameter.

**ARC:** A graphic object created by stroking, with a circle aperture, a curve closely approximating a circular arc.

**CIRCULAR INTERPOLATION:** Creating an arc.

**CLEAR:** Clear (unmark, rub, erase, scratch) the shape of a graphic object on the image.

**CONTOUR:** A closed curve defining a region.

**CURRENT POINT:** An implicitly set point in the plane that is used as a begin point for arcs and draws.

**DRAW:** A graphic object created by stroking, with a circle or rectangle aperture, a line segment from the begin point to the end point of the draw.

**FILE IMAGE:** The entire image, including all levels.

**FLASH:** A graphic object with the shape of an aperture.

**GRAPHICS OBJECT:** A flash, draw, arc or region. Graphics objects can be dark or clear. The image is created by darkening or clearing a stream of graphic objects on the image area.

**INCREMENTAL POSITION:** Position expressed as a distance in X and Y from the current point.

**INFORMATION LAYER.** See level.

**STROKE:** To create a draw (linear interpolation) or an arc (circular interpolation).

**LEVEL:** A section of Gerber data. All objects in a section have the same polarity (dark or clear).

**LINEAR INTERPOLATION:** Creating a draw.

**DARKEN:** Darken (mark, expose, paint) the shape of a graphic object on the image.

**MULTI QUADRANT MODE:** A mode defining how circular interpolation is performed. In this mode the arc is allowed extend over more than 90°. If the start point of the arc is

equal to the end point the arc is a full circle of 360°.

**PARAMETERS:** Instructions that specify how the data should be processed.

**POLARITY:** When applied to the image, positive polarity means the image is positive black on white, and negative that it is negative. When applied to a level, dark means that the object exposes or marks the image area in dark and clear means that the object clears or erases everything underneath it.

**POLYGON FILL:** This is an old name for region fill. See region.

**REGION:** A graphic object with an arbitrary shape, defined by its contour.

**RESOLUTION:** The distance expressed by the least significant digit of coordinate data. Thus the resolution is the step size of the grid on which all coordinates are defined.

**SINGLE QUADRANT MODE:** A mode defining how circular interpolation is performed. In this mode the arc cannot extend over more than 90°. If the start point of the arc is equal to the end point, the arc has length zero, i.e. covers 0°.

**STEP AND REPEAT:** A method by which successive exposures of a single image are made to produce a multiple image.

**TRANSPARENT:** Part of an object that has no effect on the image, typically a hole. Any objects under the transparent part remain visible.

## 9 Miscellaneous Deprecated Elements

### 9.1 Coordinate Data Blocks without Operation Code

Previous versions of the specification allowed coordinate data *without explicit operation code* in some situations. In the absence of an explicit operation code, a deprecated *operation mode* operates on the coordinates.

A D01 sets the operation mode to plot. It remains in plot mode till any other D code is encountered. (In older terminology, D01 turns the light on, and D02 turns it off.) This allows omitting an explicit D01 after the first coordinate data block only in sequences of D01 data blocks.



**Example:**

```
D10*  
X700Y1000D01*  
X1200Y1000*  
X1200Y1300*  
D11*  
X1700Y2000D01*  
X2200Y2000*  
X2200Y2300*
```

This saves a few bytes. However, coordinate data blocks without explicit operation code are not intuitive and sometimes lead to errors. This risk far outweighs the meager benefit of saving a few bytes. These data blocks have therefore been deprecated. The risk of using them lies solely with the writer of the file.

The operation mode after any other D code than D01 or D02 is not defined.



**Warning:** Avoid writing coordinates without operation code like the plague.

### 9.2 Open Contours in Regions

Previous versions of the specification allowed leaving contours open in a region definition.

Before the region is created all open contours are closed by connecting the last point to the first with a straight draw. Closing the contour does *not* move the current point; the current remains at the last coordinate in the file.

Open contours can be misunderstood and are therefore deprecated. Contours must be explicitly closed.

Note that contours that become self-intersecting after the automatic closure are not allowed.





## 10 Deprecated Function Codes

The next table lists deprecated codes.

Code	Function	Comments
G54	Select aperture	This historic code optionally precedes an aperture selection D-code. It has no effect. It is superfluous and deprecated.
G55	Prepare for flash	This historic code optionally precedes D03 code. It has no effect. It is superfluous and deprecated.
G70	Set the 'Unit' to inch	These historic codes perform the same action as the MO parameter. They are superfluous and deprecated.
G71	Set the 'Unit' to mm	
G90	Set the 'Coordinate format' to 'Absolute notation'	These historic codes perform as subset of the function of the FS parameter. They are superfluous and deprecated.
G91	Set the 'Coordinate format' to 'Incremental notation'	
M00	Program stop	This historic code has the same effect as M02. It is superfluous and deprecated.
M01	Optional stop	This historic code has no effect. It is superfluous and deprecated.

### *Deprecated codes*

Gerber writers can no longer use deprecated codes. Gerber readers may implement them to support legacy applications and files. The codes G54, G70 and G71 are still found from time to time. The other codes are very rarely, if ever, present.



# 11 Deprecated Parameters

The table below lists the deprecated parameters. They are explained later in this chapter.

Parameter	Function	Description	Comments
AS	Axis Select	Sets the 'Axes correspondence' graphics state variable	These parameters must be used only once. Normally they are put at the beginning of the file.
MI	Mirror Image	Sets 'Image mirroring' graphics state variable	
OF	Offset	Sets 'Image offset' graphics state variable	
IR	Image Rotation	Sets 'Image rotation' graphics state variable	
SF	Scale Factor	Sets 'Scale factor' graphics state variable	

## *Deprecated parameters*

Gerber writers (creators of Gerber files) must not use deprecated parameters. Gerber readers may support deprecated parameters. There are few legacy files with these deprecated parameters and applications generating them. If they are present, it is nearly always, if not always, to confirm the default value; in other words they have no effect. It is probably not needed anymore to implement these parameters.

## 11.1 Deprecated Graphics State Variables

There are a number of deprecated graphics state variables.

Graphics state variable	Values range	Fixed	Value at the beginning of a file
Axes correspondence	AXBY, AYBX See AS parameter	Yes	AXBY
Image mirroring	See MI parameter	Yes	A0B0
Image offset	See OF parameter	Yes	A0B0
Image rotation	0°, 90°, 180°, 270° See IR parameter	Yes	0°
Scale factor	See SF parameter	Yes	A1B1

## *Deprecated graphics state variables*

These graphic state variables have corresponding deprecated parameters, listed below.

## 11.2 AS – Axis Select

The AS parameter sets the correspondence between the X, Y data axes and the A, B output device axes.

This parameter affects the entire image. It can only be used once, at the beginning of the file.

### 11.2.1 Data Block Format

The syntax for the AS parameter is:

**<AS parameter>: AS(AXBYIAYBX)\***

Syntax	Comments
AS	AS for Axis Select
AXBY	Assign output device axis A to data axis X, output device axis B to data axis Y
AYBX	Assign output device axis A to data axis Y, output device axis B to data axis X

### 11.2.2 Examples

Syntax	Comments
%ASAXBY*%	Assign output device axis A to data axis X and output device axis B to data axis Y
%ASAYBX*%	Assign output device axis A to data axis Y and output device axis B to data axis X

## 11.3 IR – Image Rotation

The IR parameter is used to rotate the entire image counterclockwise in increments of 90° around the image (0, 0) point. All image objects are rotated.

The IR parameter affects the entire image. It must be used only once at the beginning of the file.

### 11.3.1 Data Block Format

The syntax for the IR parameter is:

**<IR parameter>: IR(0|90|180|270)\***

Syntax	Comments
IR	IR for Image Rotation
0	Image rotation is 0° counterclockwise (no rotation)
90	Image rotation is 90° counterclockwise
180	Image rotation is 180° counterclockwise
270	Image rotation is 270° counterclockwise

### 11.3.2 Examples

Syntax	Comments
%IR0*%	No rotation
%IR90*%	Image rotation is 90° counterclockwise
%IR270*%	Image rotation is 270° counterclockwise

## 11.4 MI – Mirror Image

The MI parameter is used to turn axis mirroring on or off. When on, all A- and/or B-axis data is mirrored (that is, inverted or multiplied by -1). **MI does not mirror special apertures!**

This parameter affects the entire image. It can only be used once, at the beginning of the file.



**Note:** Mirroring A-axis data flips the image about the B axis and mirroring B-axis data flips the image about the A axis.



**Warning:** It is strongly recommended *not* to use the MI parameter. Avoid it like the plague. The exception for special apertures is confusing and leads to mistakes. If an image must be mirrored, write out the mirrored coordinates and apertures.

### 11.4.1 Data Block Format

The syntax for the MI parameter is:

**<MI parameter>: MI[A(0|1)][B(0|1)]\***

Syntax	Comments
MI	MI for Mirror image
A(0 1)	Controls mirroring of the A-axis data: A0 – disables mirroring A1 – enables mirroring (the image will be flipped over the B-axis) If the A part is missing then mirroring is disabled for the A-axis data

B(0 1)	<p>Controls mirroring of the B-axis data:</p> <p>B0 – disables mirroring</p> <p>B1 – enables mirroring (the image will be flipped over the A-axis)</p> <p>If the B part is missing then mirroring is disabled for the B-axis data</p>
--------	---

## 11.4.2 Examples

Syntax	Comments
%MIA0B0*%	No mirroring of A- or B-axis data
%MIA0B1*%	No mirroring of A-axis data. Mirror B-axis data
%MIB1*%	No mirroring of A-axis data. Mirror B-axis data

## 11.5 OF - Offset

The OF parameter moves the final image up to plus or minus 99999.99999 units from the imaging device (0,0) point. The image can be moved along the imaging device A or B axis, or both. The offset values used by OF parameter are absolute. If the A or B part is missing, the corresponding offset is 0. The offset values are expressed in units specified by MO parameter.

This parameter affects the entire image. It can only be used once, at the beginning of the file.

### 11.5.1 Data Block Format

The syntax for the OF parameter is:

**<OF parameter>: OF[A<Offset>][B<Offset>]\***

Syntax	Comments
OF	OF for Offset
A<Offset>	Defines the offset along the output device A axis
B<Offset>	Defines the offset along the output device B axis

The **<Offset>** value is a decimal number n preceded by the optional sign ('+' or '-') with the following limitation:

$0 \leq n \leq 99999.99999$

The decimal part of n consists of not more than 5 digits.

## 11.5.2 Examples

Syntax	Comments
%OFA0B0*%	No offset
%OFA1.0B-1.5*%	Defines the offset: 1 unit along the A axis, -1.5 units along the B axis
%OFB5.0*%	Defines the offset: 0 units (i.e. no offset) along the A axis, 5 units along the B axis

## 11.6 SF – Scale Factor

The SF parameter sets a scale factor for the output device A- and/or B-axis data. The factor values must be between 0.0001 and 999.99999. The scale factor can be different for A and B axes. If no scale factor is set for an axis the default value '1' is used for that axis.

All the coordinate data are multiplied by the specified factor value for the corresponding axis. Note that apertures are *not* scaled.

This parameter affects the entire image. It can only be used once, at the beginning of the file.

### 11.6.1 Data Block Format

The syntax for the SF parameter is:

**<SF parameter>: SF[A<Factor>][B<Factor>]\***

Syntax	Comments
SF	SF for Scale Factor
A<Factor>	Defines the scale factor for the A-axis data
B<Factor>	Defines the scale factor for the B-axis data

The **<Factor>** value is an unsigned decimal number n with the following limitation:

0.0001 <= n <= 999.99999

The decimal part of n consists of not more than 5 digits.

### 11.6.2 Examples

Syntax	Comments
%SFA1B1*%	Scale factor 1
%SFA.5B3*%	Defines the scale factor: 0.5 for the A-axis data, 3 for the B-axis data





## 12 Deprecated RS-274-D or Standard Gerber

The current Gerber file format, as specified in this document, is also known as RS-274X or Extended Gerber. There is also a historic format called Standard Gerber or RS-274-D format. It differs from the current Gerber file format (RS-274X), in that it:

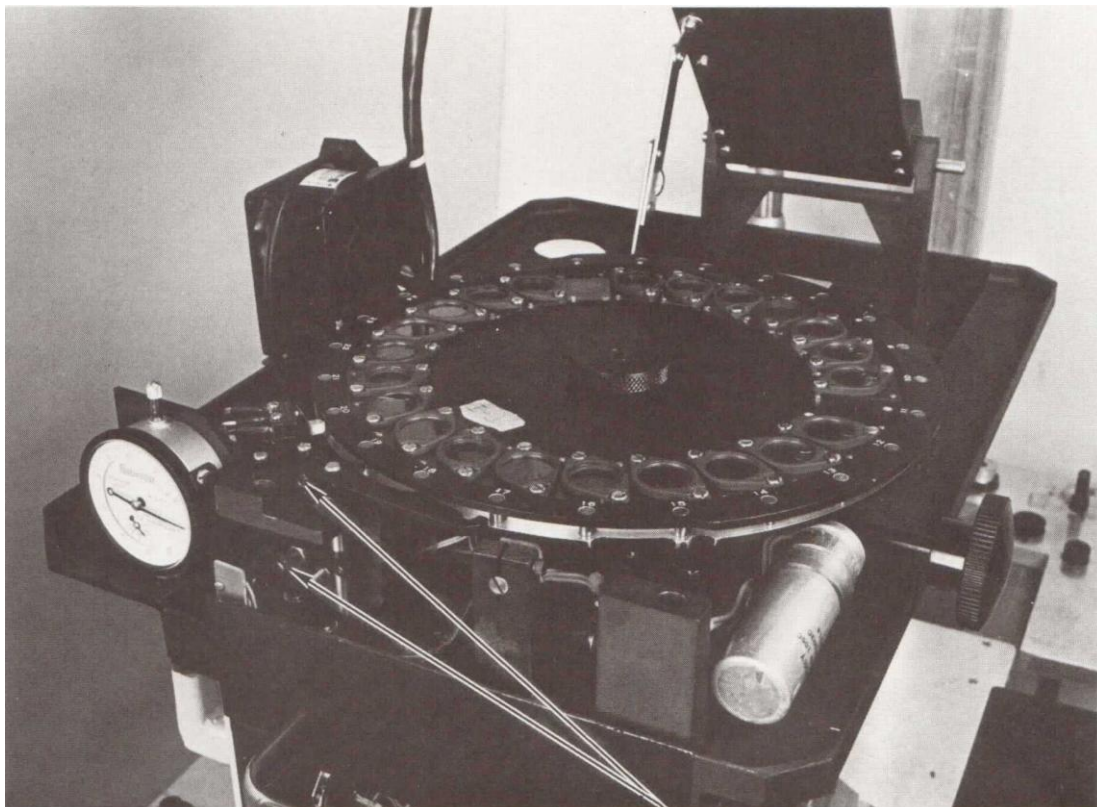
- does not support G36 and G37 codes
- supports the deprecated codes, and
- does not support parameters; therefore coordinate format and apertures cannot be defined

### 12.1 Standard Gerber must not be used

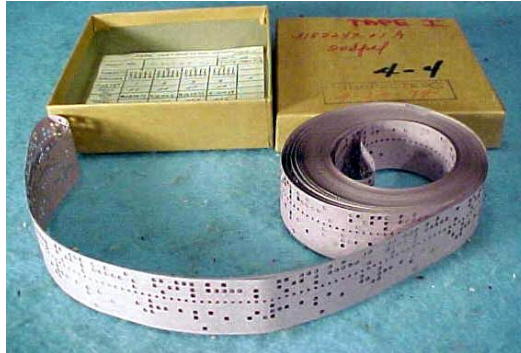
Standard Gerber is obsolete and deprecated. It is not an image description standard. It is not suited for automatic processing. It has many drawbacks over the current Gerber file format and not a single advantage. All users of the Gerber file format are strongly advised to use the current Gerber file format – Extended Gerber, or RS-274X. The reasons for this are given below.

### 12.2 Origin and purpose of Standard Gerber

In the 1960s and 1970s, images were produced on lithographic film by a vector photoplotter, a precision optical Numerical Control machine. Images were produced by beaming light from the plotter's light source onto the film through an aperture on a wheel like that shown in the photograph below. This wheel was rotated to select the appropriate aperture, or it could be substituted by another aperture wheel if additional aperture sizes were needed.



The data for the exposure process was contained in a Standard Gerber file, which was typically recorded onto magnetic or paper tape (see pictures), which was in turn mounted onto the vector photoplotter by the operator.



The operator consulted the accompanying notes, typed the coordinate format on a machine console, mounted the appropriate aperture wheel, changed apertures if necessary, and started the plotter. The Standard Gerber file then drove the plotter through the required movements, controlled the aperture wheel and exposure light, and produced the desired image.

Standard Gerber was so well suited to this task that it became the industry standard.

That was decades ago. Vector photoplotters have not been used since, so Standard Gerber has lost its *raison d'être*. While it deserves a place of honor in the Computer History Museum, Standard Gerber has no place at all in the 21st century's electronics industry.

And yet it is still used by some. This makes no sense at all.

## 12.3 Standard Gerber is a NC format

From the above, it is clear that Standard Gerber is an NC (Numerical Control) machine format, and not an image description format. It contains neither the coordinate format definition, so the meaning of coordinate data is undefined, nor aperture definitions, so the meaning of flashes and interpolations is undefined.

Thus if an image is to be defined using Standard Gerber, additional information is essential. This typically comes in the form of a so called “wheel file” consisting of notes in an informal text format, plus drawings that define the more complex apertures. The problem is that there is no standard for this extra information, creating enormous potential for error and misunderstandings. This puts the onus squarely on operators' shoulders to ensure that all of the information is assembled and checked on a workstation – manually and with the help of software tools – in order to be sure that all the necessary image data is present.

As if this were not enough, an additional issue is that Standard Gerber renders the informal description of complex apertures, SMD apertures and areas so difficult that designers give up, and opt instead to paint them. This in turn creates such chaos that there is a very real risk of losing valuable data in both CAD and CAM operations. Thus the CAM engineer must be extremely careful to recover, and piece together, the pads in the design.

All of which renders Standard Gerber totally unsuitable for current CAD to CAM data transfer. This format, from the days of paper tape, punched cards, teletypes and electrical typewriters, offers not one single advantage over Extended Gerber.

## 12.4 Standard Gerber is not a standard

So Standard Gerber, despite its name, is not an image definition standard, as it must be supported by a whole lot of extra non-standardized information in order to define an image. That's why Ucamco has defined the new Extended Gerber format. This, unlike its predecessor, IS a standard, as it standardizes the additional data needed, puts it in the file header, and adds some sorely needed extensions.

## 12.5 A fallacy

The following is sometimes said: *"The only difference between Standard Gerber and Extended Gerber is that in Extended Gerber the wheel file is embedded in the file. As software was developed to extract data automatically from the wheel files, this is no big deal."*

We beg to differ:

- It is not the only difference.
- This difference is a big deal.

Firstly, the other big difference is that Extended Gerber is a richer format that has all the constructs necessary for describing a PCB image efficiently. Pads are properly described as pads, ensuring that no data is lost.

Secondly, this difference is a really big deal. While it is true that a lot of effort was spent on automating the task of inputting the accompanying notes, only a fraction of all data sets can in fact be read in automatically because they are in a free format. While this freedom was perfectly adequate for the vector photoplotter operator of old, it flies in the face of standardization and automation, which consequently becomes a less reliable and higher maintenance process. And what happens if the notes arrive in another language – imagine, for example, automating the input of a wheel file in Japanese. Or of its supporting drawings, for which again, there are no format definitions. It becomes clear pretty quickly that it is not possible to fully and reliably automate the transfer of such informal data, so the operator must carefully check all results for errors. This is particularly important if we consider that a lack of standards can also mean lack of clarity about the intentions of the designer, and where responsibility lies in case of errors.

Compare this with the clarity of the formal, standardized aperture definitions in Extended Gerber: reading them in is straightforward, with no need to pore over the results for errors. And as there is a standard, it is clear what was intended, and who is responsible in case of a mistake. So yes, this difference is a big deal. It is the difference between using a published standard format and each individual using his own unspecified format. It is the difference between painstaking, minute manual work and inspection, and reliable, automatic data transfer.