The Gerber Job File

Including fabrication documentation in Gerber

Rev. 2019.02

This specification was developed by Karel Tavernier.

Having a question or remark about the spec? Please contact us at gerber@ucamco.com

1 Preface

Since decades, Gerber is the standard for describing the 2D images – copper layers, solder mask, drills – in PCB fabrication data. Attributes added with Gerber X2 provide a standard to 'add intelligence to the image' by defining the layer structure – which file is which layer – identify via and SMDs component, pin numbers and reference descriptions.

But PCB fabrication data is not just about images - it includes finish, overall thickness, materials, solder mask color. They are essential for the quoting, planning, engineering, CAM and fabrication of the bare and assembled board. It is often transferred informally, in drawings and texts, and handled manually, wasting time and risking errors.

This specification defines how to transfer this information in a machine-readable manner as part of Gerber fabrication data.

Please send questions or suggestions about this specification to gerber@ucamco.com

2 Specification

2.1 Overview

PCB characteristics pertaining to the job as a whole, such as the finish or overall thickness, are stored in a separate file, the Gerber job file. One job file per PCB. It contains the following sections:

1. **Header**: information about the file itself, such as its creation date.
2. **General specs**: overall board characteristics such as finish.
3. **Material stackup:** specification of the stackup and characteristics of the materials used.
4. **Design rules**: the design rules used when laying out the PCB.
5. **Files attributes:** the polarity and function (e.g. top copper layer) of all Gerber files.

The Gerber job file is a separate file, not part of the image files. It contains only PCB characteristics and no image data. Applications that process a Gerber job file do not need to be image-savvy.

The image format is not changed. Thus, compatibility is achieved. Legacy applications can happily ignore the job file and process the images as before. Existing image input/output code does not need to be changed when implementing the Gerber job file.

The Gerber job file is compatible with both X1 and X2.

Partial implementations are allowed. Better a partial job file than no job file at all.

The job file can be extended with custom information.

The job file is simple and human readable.

The job file intentionally does *not* specify default values for the PCB characteristics. If a characteristic is not present on the job file, it is not defined *in* the job file and must be specified by other means, for instance fabricator defaults or a side agreement. This restriction allows partial implementations, where certain parameters are absent because they are not yet implemented. And anyhow, a mere data transfer format must not pretend to specify what defaults for the PCB industry should be.

The job file intentionally defines the *technical* PCB parameters only, *not* commercial conditions such as delivery times, pricing and quantities. The reason is simple. The same PCB can be ordered at different times, with different commercial conditions, or it can be ordered in a panel. The physical PCB however is the same, and its definition must not be touched. Commercial conditions are best in a separate file.

The job file intentionally does *not* cover assembly arrays. Layout systems are generally designed to generate a single PCB rather than arrays. This makes sense: assembly panels are typically not decided by the designer but by the assembler. The same PCB may be assembled in different panels at different times, of a panel may contain multiple PCB's. Therefore the assembly panel definition must be separate from the single PCB definition. However, the job file contains information that is essential to the assembler, such as the job size.

2.2 Syntax

The job file follows the JSON syntax.

For an introduction to JSON see http://json.org/ . For an online JSON syntax checker see http://jsoneditoronline.org/ .

In addition, names follow the restriction of the main Gerber spec. Wherever possible, we re-use the attribute values from the image spec as JSON names or values. Names starting with a "$" are reserved for custom extensions.

The unit for length is mm.

The standard file extension is ".gbrjob". An example name: `Ctlr54382r4.gbrjob`

Partial implementations are allowed. One can omit entire sections, or e.g. define only a few board characteristics in the general specs section. Missing parameters are not defined, there are no defaults.

2.3 Header

This object has values that provide information about the Gerber job file itself rather than about the PCB it describes, e.g. the creation date of the file.

Name/value	Usage

"GenerationSoftware": {
"Vendor": "**<string>**",
"Application": "**<string>**",
"Version": "**<string>**
}
	Identifies the software generating this file. See the file attribute .GenerationSoftware for the values and their interpretation.

"CreationDate": "**<string>**"	Creation date of the file. See the .CreationDate file attribute for allowed values and their interpretation.
"Comment": "**<string>**"	Comment about the file.

 **Example:**

```
"Header": {
  "GenerationSoftware": {
    "Vendor": "Ucamco",
    "Application": "UcamX",
    "Version": "2017.12"
  },
  "CreationDate": "2018-01-20T15:59:51+01:00",
  "Comment": "Example"
}
```

2.4 General Specs

This object contains characteristics that apply to the whole board, such as finish. Characteristics which apply to a specific element, such as the legend color, are stored in other objects where the subject such as legend is identified. The object and all characteristics below are optional.

Name/value	Usage

"ProjectId": {
"Name": "**<string>**",
"GUID": "**<string>**",
"Revision": "**<string>**"
}
	See the file attribute .ProjectId for values and their interpretation.

"Owner": "**<string>**"	Reference of the design owner, as used by himself.
"Part": "Single"	See the file attribute .Part. As the Gerber job file is about a single PCB the value must be "Single".

"Size": {
"X": **<decimal>**,
"Y": **<decimal>**,
"Tol+": **<decimal>**,
"Tol-": **<decimal>**
}
	Board size, being the size of the axis-aligned enclosing rectangle of the board outline. *It is strongly recommended to include this information in any job file.*
	Tolerances are optional.

"LayerNumber": **<integer>**	Number of copper layers. *It is strongly recommended to include this information in any job file.*
"BoardThickness": **<decimal>**	The overall thickness of the base material and all conductive materials deposited thereon.
"IPC-2221-Type": **<integer>**	The board type according to IPC-2221. The integer can take the values from 1 to 6, corresponding to the six primary board types: • Type 1 - Single-sided • Type 2 - Double-sided • Type 3 – Multilayer, TH components only • Type 4 – Multilayer, with TH, blind and/or buried vias. • Type 5 - Multilayer metal-core board, TH components only • Type 6 - Multilayer metal-core
"IPC-600-Class": "**(1|2|3|NA)**"	
"Standard": <**string**>	MIL, JSS, IPC6012B, PCA600, …

"ImpedanceControlled": (**true**|**false**)	
"UL_Logo": (**true**|**false**)	Indicates whether an UL logo must be added. The exact PCB requirements this corresponds to are left open at this time.
"Fabricator_Logo": (**true**|**false**)	Indicates whether a fabrication logo must be added.
"Fabricator_Datecode": (**true**|**false**)	Indicates whether a fabrication date code must be added.
"ViaProtection": ["**<IPC-4761>**" **{,"<IPC-4761>"}**] where **<IPC-4761>=(Ia|Ib|IIa|IIb|IIIa|IIIb|IVa|IVb|V|VI|VII|None)**	This via protection types present, according to the IPC-4761 classification: Ia Tented - Single-sided Ib Tented - Double-sided IIa Tented and Covered – Single-sided IIb Tented and Covered – Double-sided IIIa Plugged – Single-sided IIIb Plugged – Double-sided IVa Plugged and Covered – Single-sided IVb Plugged and Covered – Double-sided V Filled (fully plugged) VI Filled and Covered VIII Filled and Capped None No protection Which vias are of which type is determined by .AperFunction in the Gerber drill files.
"HolePlatingThickness": **<decimal>**	The plating thickness in the holes.
"HalogenFree": (**true**|**false**)	
"Press-fit": (**true**|**false**)	Presence of press-fit holes. These are holes for connector pins that will not be soldered but pressed in the holes. They require tighter tolerances.
"HeatSinkPaste": (**true**|**false**)	Indicates the presence of heat sink paste.
"EdgePlating": (**true**|**false**)	
"Castellated": (**true**|**false**)	

"EdgeConnector": (**true**|**false**)	
"EdgeConnectorBevelled": (**true**|**false**)	
"HardGoldArea": **<decimal>**	Area of hard gold in mm squared. Where that gold is can be defined with a gold mask file.
"RoHS": (**true**|**false**)	
"Finish": "(**HAL SnPb|HAL lead-free| Immersion tin|Immersion nickel| Immersion silver|Immersion gold (ENIG)|ENEPIG|Hard gold|OSP| HT_OSP|None|<string>)**"	
"Foil": "**(Electro-Deposited |Rolled|<string>))**"	Copper foil type
"Substrates": ["**<substrate>**" {,"**<substrate>**"}] where **<substrate> = (FR4|Polyimide|Polyolefin|Al|PTFE| Teflon|Ceramic|<string>)**	The substrate(s) used in the PCB. The order of the substrates in this list is arbitrary and is not related to the physical order in the PCB.
"Material_Tg": **<decimal>**	Minimal glass transition temperature, in degrees Celcius.
"ITAR": (**true**|**false**)	The job is ITAR controlled.
"ElectricalTest": (**true**|**false**)	
"Notes": "**<string>**"	A free field with informal information.

Example:

```
"GeneralSpecs": {
  "Part": "Single"
  "Size": {
    "X": 160,
    "Y": 50.8
  },
  "LayerNumber": 4,
  "BoardThickness": 1.6,
  "IPC-600-Class": 2,
  "Finish": "Immersion Gold (ENIG)"
}
```

2.5 Material Stackup

The material stackup array defines the material layers and their attributes. For most standard designs the stackup is obvious and full description not needed and usually not included. However even in simple design one may want to define the solder mask and legend colors, or even the copper weights.

An array element is a layer in the material stackup. The order of the elements is the order of the material layers. The type of material is identified by the "Type:" pair and is mandatory for each array element. Other optional pairs allow to specify properties of each material. Common sense indicates which properties are appropriate for which material.

If a material stackup is included, it must be complete - all layers in the stackup must be present. It is allowed to add a material without any attributes to indicate the presence of that material.

Material Stackup Pairs	
Name/value	**Name/value**
"Type": "**(Copper|Dielectric|Legend|SolderMask|CoverLay|PeelableMask|Carbon|<string>)**"	
"Thickness": **<decimal>**	*Base* thickness. This excludes any extra thickness added by plating.
"Color": "**(Red|Yellow|Green|Blue|White|Black|R<integer>G<integer>B<integer>)[,(Gloss|Semi-matte|Matte)]**" **(**<integer> ranges from 0 to 255)	
"DielectricConstant": **<decimal>**	Dielectric constant.
"LossTangent": **<decimal>**	Loss tangent.
"Conductivity": **<decimal>**	Conductivity in S/m.
"Tg": **<decimal>**	Minimal glass transition temperature.
"Material": "<**substrate**>"	Material used as substrate.
"Note": "**<string>**"	A note on the layer or material.

"Substacks:" [**<integer>{,<integer>}**]	This array is only applicable for flex-rigid boards. It uses the concept of substacks that is used by Polar Instruments in its handling of flex-rigid boards. There is a master stackup, with all material present; it is substack 1. Different parts of the flex-rigid use different substacks, each identified by a positive integer. The array substacks contains the indices of the substacks in which the material is present, in ascending order.

An example shows best how it is done, it is not rocket science. Here is the material stackup of a four-layer board, with solder mask on both sizes and legend on top:

Example:

```
"MaterialStackup": [
  {
    "Type":      "Legend",
    "Color":     "White"
  },
  {
    "Type":       "SolderMask",
    "Thickness": 0.025,
    "Color":      "Green"
  },
  {
    "Type":       "Copper",
    "Thickness": 0.03556
  },
  {
    "Type":       "Dielectric",
    "Thickness": 0.34
  },
  {
    "Type":       "Copper",
    "Thickness": 0.01500
  },
  {
    "Type":       "Dielectric",
    "Thickness": 0.34,
    "DielectricConstant": 4.7
  },
  {
    "Type":       "Copper",
```

```
      "Thickness": 0.01500
    },
    {
      "Type":      "Dielectric",
      "Thickness": 0.34
    },
    {
      "Type":      "Copper",
      "Thickness": 0.03556
    },
    {
      "Type":      "SolderMask",
      "Thickness": 0.02500,
      "Color":     "Green"
    }
  ]
```

Note that this describes a CAD stackup – also called virtual stackup by Polar Instruments. A layer represents a type of material, such as copper or dielectric. A two or three prepreg layers can be used to fabricate one dielectric layer, and a core can be used for two copper layers separated by a dielectric. In other words, in fabrication the virtual or CAD stackup described here is converted into a real of fabrication stackup.

If we only need to specify the colors the material stackup is as in the example below.

Example:

```
"MaterialStackup": [
  { "Type": "Legend",     "Color": "White" },
  { "Type": "SolderMask", "Color": "Green" },
  { "Type": "Copper"                       },
  { "Type": "Dielectric"                   },
  { "Type": "Copper"                       },
  { "Type": "Dielectric"                   },
  { "Type": "Copper"                       },
  { "Type": "Dielectric"                   },
  { "Type": "Copper"                       },
  { "Type": "SolderMask", "Color": "Green" }
]
```

One may remark that this requires plenty of dummy lines to represent just three colors. While true, this is not really a problem. The very generic stackup array allows to define the colors. It makes no sense to introduce a redundant ad-hoc structure dedicated to the colors, just to save a few dummy lines.

2.6 Design Rules

The design rules array contains the intended design rules used by CAD to lay out the copper layers. Thus the intended design rules are known without needing to perform complex geometric analysis of the copper layers. This is useful information for planning and quoting. It helps CAM to detect and handle possible layout problems.

The copper layers are grouped in classes with the same design rules. Typically, design rules are different for inner and outer layer, and then the classes are inner and outer. Each design rule class is an entry in the array. The entry consists of a definition of the layers in the class and their design rules.

Design rules name/value pairs.	
"Layers": **"(Inner|Outer|L<integer>)"**	Design rule group. All layers in the group share the same design rules. The precedence is "Ln", "Inner" or "Outer". There is no inheritance, all applicable design rule values must be set for each class.
"PadToPad": **<decimal>**	
"PadToTrack": **<decimal>**	
"PadToRegion": **<decimal>**	Regions are also known as copper pours.
"TrackToTrack": **<decimal>**	
"TrackToRegion": **<decimal>**	
"RegionToRegion": **<decimal>**	
"MinLineWidth": **<decimal>**	
"MinRing": **<decimal>**	
"MinClearanceToProfile": **<decimal>**	
"Notes": "<**string**>"	Informal information.

An example with the typical inner/outer design rules shows how it works.

Example:

```
"DesignRules": [
  {
    "Layers": "Outer",
    "PadToPad":       0.15,
    "PadToTrack":     0.15,
    "TrackToTrack":   0.15,
    "MinLineWidth":   0.2,
    "TrackToRegion":  0.381,
    "RegionToRegion": 0.381
  },
  {
    "Layers": "Inner",
    "PadToPad":       0.15,
    "PadToTrack":     0.15,
    "TrackToTrack":   0.15,
    "MinLineWidth":   0.254,
    "TrackToRegion":  0.508,
    "RegionToRegion": 0.508
  }
]
```

2.7 Files Attributes

The files attributes array assigns attributes to each Gerber file in the archive. The syntax is very general and can assign any attribute to the files.

The array now defines the function and polarity of each Gerber file in the archive, both for Gerber X1 and X2 files. This information can also be expressed as file attributes in the Gerber X2 files. By centralizing it in the job file it is available without opening each Gerber image file.

The file path serves as a unique index to identify an array element; attributes associated with that element then specify function and polarity. The order of the files in the array is free; for human readability it is suggested to sort them from top to bottom, as in the example.

An example shows best how it is done, it is not rocket science. The see the file attributes .FileFunction and .FilePolarity for interpretation and values.

 **Example:**

```
"FilesAttributes": [
  {
    "Path":         "AZ2375EM_Top_Silk.gbr",
    "FileFunction": "Legend,Top",
    "FilePolarity": "Positive"
  },
  {
    "Path":         "AZ2375EM_Top_SolderMask.gbr",
    "FileFunction": "Soldermask,Top",
    "FilePolarity": "Negative"
  },
  {
    "Path":         "AZ2375EM_Top_Copper.gbr",
    "FileFunction": "Copper,L1,Top",
    "FilePolarity": "Positive"
  },
  {
    "Path":         "AZ2375EM_Bot_Copper.gbr",
    "FileFunction": "Copper,L2,Bot",
    "FilePolarity": "Positive"
  },
  {
    "Path":         "AZ2375EM_Bot_SolderMask.gbr",
    "FileFunction": "Soldermask,Bot",
    "FilePolarity": "Negative"
  },
  {
    "Path":         "AZ2375EM_Bot_Drill_Plated.gbr",
    "FileFunction": "Plated,1,4,PTH",
    "FilePolarity": "Positive"
  },
```

```
    {
      "Path":         "AZ2375EM_Drawing.gbr",
      "FileFunction": "FabricationDrawing",
      "FilePolarity": "Positive"
    }
  ]
```

This section is an exception in the job file as it does not contain information about the PCB itself, but about how the PCB is represented in the archive.

3 Examples

3.1 Minimal CAD Job File

Below is the minimal job file that CAD must include in the fabrication data. CAD systems 'know' board size and layer count. CAD systems must output a simple job file with board size and layer count (and ideally any other parameters they might know). This is the basis for pricing and allowing assemblers to define their panels automatically without having to process Gerber image files. Other applications can read and extend this initial job file with more information.

```
{
  "Header": {
    "GenerationSoftware": {
      "Vendor":      "Ucamco",
      "Application": "UcamX",
      "Version":     "2017.12"
    },
    "CreationDate":  "2018-01-20T15:59:51+01:00",
    "Comment": "Example"
  },
  "GeneralSpecs": {
    "Part": "Single",
    "Size": {
      "X": 160.0,
      "Y":  50.8
    },
    "LayerNumber":    4,
    "BoardThickness": 1.6
   }
}
```

3.2 Basic Job File

The job file contains the basic board parameters needed for a quote next to the images.

```
{
  "Header": {
    "GenerationSoftware": {
      "Vendor":      "Ucamco",
      "Application": "UcamX",
      "Version":     "2017.12"
    },
    "CreationDate":  "2018-01-20T15:59:51+01:00",
    "Comment":       "Example"
  },
  "GeneralSpecs": {
    "Part":  "Single",
    "Owner": "Galactic Corporation",
    "ProjectId": {
      "Name":     "HFDF Controller nRF52832",
      "GUID":     "f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
      "Revision": "Rev. 2b"
     },
    "Size": {
     "X": 160.0,
     "Y":  50.8
    },
    "LayerNumber":     4,
    "BoardThickness":  1.6,
    "ROHS":            true,
    "B_IPC-600-Class": 2,
    "Finish":          "Immersion gold (ENIG)"
  },
  "MaterialStackup": [
    { "Type": "Legend",      "Color":     "White" },
    { "Type": "SolderMask",  "Color":     "Green" },
    { "Type": "Copper",      "Thickness": 0.03556 },
    { "Type": "Dielectric"                         },
    { "Type": "Copper",      "Thickness": 0.01500 },
    { "Type": "Dielectric"                         },
    { "Type": "Copper",      "Thickness": 0.01500 },
    { "Type": "Dielectric"                         },
    { "Type": "Copper",      "Thickness": 0.03556 },
    { "Type": "SolderMask",  "Color":     "Green" }
  ]
}
```

4 Revisions

Rev 2018.04

This is the initial version of the Gerber Job File specification.

This specification was developed by Karel Tavernier. In January 2017 an initial draft was published for public review by the Gerber user communed. Based on the received input the draft went through seven revisions. The review process was closed in April 2018 and resulted in this specification. The following people contributed to this draft: Jean-Pierre Charras, Alan Shrives, Bruce McKibben, Wim De Greve, Ken Caluwaert, Aurelio Bantigue, Luc Samyn, Scott McClusky, Richard Attrill, Paul Wells-Edwards, Remco Poelstra, Rik Breemeersch.

Rev 2019.02

Removed superfluous design rule group **All**.

"Foil" values contained **<f>,** replaced by **<string>.**

"Type" values contained **Other <string>,** replaced by **<string>**

5 Copyright

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

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

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

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