XNC

The PCB CAD to CAM Exchange NC format

A subset of IPC-NC-349, aka Excellon

Rev 2019.02.16

Please send your comments to gerber@ucamco.com

This subset was defined by Denis Morin, Karel Tavernier, Jean-Pierre Charras and Marius Matioc. This specification was written by Denis Morin and Karel Tavernier.

Contents

1 Preface

Among other PCB fabrication data, CAD needs to send to CAM the location and size of drill holes and rout slots.

The best and simplest solution is to transfer drill/rout information in the ubiquitous Gerber format. Gerber supports the primitives and the attributes needed for drill information - see the Gerber format specification. When drill information is transferred in Gerber there are very rarely problems.

It is sometimes objected that one cannot send a Gerber file to a drilling machine. While this is true, it is irrelevant. Drill/rout files from CAD are never directly sent to a drilling machine, but are read in the fabricator's CAM system, where they are panelized, compensated for distortions in lamination, etc. When this is done, CAM outputs drill files output exactly as the fabricator's drilling machines need them, with feeds and speeds, nibbling, etc. For CAD, the question is not which format is best for feeding drilling machines. The question is which format is best for input into CAM. This is undoubtedly Gerber.

However, for historic reasons drill information is transferred in what is called an NC or Excellon file, loosely based on the IPC-NC-349 specification from August 1985. Many of these NC files are of deplorable quality. The reason for the deplorable state of NC files is that, for CAD CAM data exchange, the IPC-NC-349 specification is deeply problematic:

- The IPC specification is expensive, therefore only few have access to a copy. The company Excellon published a specification based on IPC-NC-349, and this was widely used. (Hence the name Excellon files.) However, Excellon no longer publishes its specification, and one has to rely on "black copies", probably published in violation of the copyright. Ironically, the PCB industry develops NC files based on non-existent, illegal or unaffordable specifications. All too often the specifications are abandoned altogether, and the software is reverse engineered based on the chaotic NC files that circulate. Confusion breeds more confusion.
- The 349 format is an input format for NC drill machines, not a CAD/CAM exchange format. Most commands are superfluous to CAD/CAM. Nobody implements it completely, it is much too complicated. However, it is not clear which subset is to be used. Consequently, dialects abound. Some intentionally write incomplete NC files for fear of picking the wrong subset and relegate essential information to informal sidecar files.
- The 349 format is overcomplicated, with overlapping methods to express the same data. These embellishments increase the confusion further.
- The Excellon speciation not only contained the IPC-NC-349 command, but also historic Excellon proprietary commands. This added another slate of useless complications.
- The 349 specification is often very obscure. Consequently, different interpretations abound.
- The 349 specification promotes fixed point coordinate data, with an implied decimal point. In itself there is nothing wrong with fixed point data, but 349 provides no way to specify where the decimal point is! Without knowing where the decimal point is the file is meaningless. This is the fatal flaw. To get around this glaring hole in the spec, developers had to invent their own way to try to specify where the decimal point is. More dialects and confusion.

NC files are slowly being replaced by Gerber but will be around for a long time. To address the chaos with NC files while they are still around the software organizations whose logo's are on the front page of this document came together in an informal consortium to develop a simple and clear specification for NC files: the CAD/CAM Exchange format. A tight and unequivocal specification gives the industry a reference to work towards, so that NC files over time converge to a common standard rather than diverging more and more.

The CAD/CAM Exchange NC format is a minimal subset of IPC-NC-349 able to exchange CAD/CAM drill information. Great care was taken to make the specification compact and unequivocal. Files written according to this specification are simple, complete, clear, easy to interpret and human readable.

As the XNC spec is a strict subset of IPC-NC-349, legacy NC input processors will read XNC files without a hitch.

Developers of NC output software can now work from a simple specification rather than wading through confusing documents and reverse engineering from dialectical, incomplete and confusing NC files. Developers of NC input software continue to face a harder task: each one will have to decide which fraction of the dialectical, incomplete and confusing NC files they will support and reverse engineer the interpretation of such files.

2 Overview

2.1 Purpose and scope

The Exchange NC format describes drill and rout holes in PCB fabrication data, for exchange between PCB CAD and CAM. An XNC file is complete: it does not need sidecar files or additional parameters. It is human readable and easy to interpret.

This specification is mainly intended for developers of NC output files. Files complying with this specification will be read by legacy NC input software. However, for the developers of NC input processers the situation is not so simple: a lot of dialectical and incomplete files circulate. Developers of input software must decide how many dialects they will support and reverse engineer their interpretation.

Note that technically, it is best to use the Gerber format for drill and rout files. Gerber is the main format in fabrication data, and can express drill/route files perfectly - the XNC format is an unnecessary variation. However, for historic reasons the XNC format is also often used.

2.2 Syntax

An XNC file is expressed in the 7-bit ASCII codes 32 to 126 (i.e. the printable characters in ANSI X3.4-1986) plus codes 10 (LF, Line Feed) and 13 (CR, Carriage Return). No other characters are allowed. XNC files are therefore printable and human readable. XNC files are case-sensitive. Code 32 (SP, Space) is only allowed in comments.

Commands are higher level semantic elements of an XNC file. Each line in the file is a single command: one line, one command. Commands must be in upper case.

An XNC file consists of a stream of commands. The number of commands is not limited. The commands create a stream of objects describing the drill and rout holes. An XNC file can be processed in a single pass.

The standard file extension is .xnc or .XNC

Mac OS X UTI:

```
<key>UTExportedTypeDeclarations</key>
<array>
  <dict>
    <key>UTTypeIdentifier</key>
    <string>com.ucamco.drill</string>
    <key>UTTypeReferenceURL</key>
    <string>http://www.ucamco.com/gerber</string>
    <key>UTTypeDescription</key>
    <string>NC drill file</string>
    <key>UTTypeConformsTo</key>
    <array>
        <string>public.plain-text</string>
        <string>public.image</string>
    </array>
    <key>UTTypeTagSpecification</key>
    <dict>
```

```
        <key>public.filename-extension</key>
        <array>
            <string>xnc</string>
        </array>
    </dict>
  </dict>
</array>
```

2.3 Tools

A tool is a circle with a given diameter, identified by a tool number.

 Example, tool number 1 with a diameter of 1.2:

```
T01C1.2
```

A tool is replicated at given coordinates to create a drill hole or stroked over a given path to create a rout slot. The tool diameter is the end (or finished) diameter, *after* plating. The name tool is historic, and actually a misnomer: to create a via hole of 0.5 mm, one will not use a drill tool of 0.5 mm as plating must be considered.

2.4 Processing state

During the processing of an XNC file a processing state is maintained. The processing state affects how the commands work.

Processing state parameter	Value range	Constant or variable
Unit	METRIC or INCH. See 3.3.	Constant
Current point	Point in the plane	Variable
Selected tool	Used to generate drilled or routed holes. See 3.8	Variable
Drill/rout mode	Drill or rout. See 3.6 and 3.7	Variable

2.5 Hole objects

An XNC file creates a stream of hole objects. A hole object represents a drill hole/slot/rout. It has a shape and a position.

There are three types of hole objects:

- A drill hole. Circular.
- A straight-line rout segment. It has a constant thickness and circular line endings.
- An arc is circular rout segment. It has a constant thickness and round line endings.

2.6 Coordinates

Coordinates is a decimal number representing the coordinates in the file unit.

Signs in coordinates are allowed; the '+' sign is optional. Coordinates must have at least one character. Zero therefore must be encoded as "0".

When XNC files are part of fabrication data, they must have the *same* coordinate system – origin and axes - as the Gerber copper layers, in other words they must align perfectly with the Gerber files; no offset, rotation, mirroring. They must be of sufficient resolution to avoid rounding problems.

2.7 Command overview

stands for a number.

$ stands for a string.

Command Code	Command name	Section
;$	Comment	3.1
M48	Start of header	3.2
METRIC|INCH	Set unit	3.3
T#C#	Tool declaration	0
%	End of header	3.5
G05	Set drill mode	3.6
G00X#Y#	Set rout mode	3.7
T#	Select tool	3.8
X#Y#	Drill hit	3.9
M15	Tool down	3.10
M16	Tool up	3.11
G01X#Y#	Linear rout	3.12
G02X#Y#A#	Circular clockwise rout	3.13
G03X#Y#A#	Circular counter-clockwise rout	3.14
M30	End of file	3.15

2.8 Example XNC file

File	Meaning	Effect	
M48	Beginning of header.	Set file unit and tool sizes	Header
METRIC	Unit is METRIC (mm).		
T01C0.6	Tool sizes in mm. Tool 1 is 0.6 mm, tool 2 is 0.7mm, etc.		
T02C0.7			
T03C0.8			
T04C1.0			
%	End of header.		
G05	Set drill/rout mode to drill	Drill mode	Body
T01	Select tool T01	Create 2 holes	
X9.01Y3.3375	A hole with T01 size at XY		
X9.01Y4.3125	Another hole at another XY		
T02	Select tool T02	Create 4 holes	
X8.01Y4.8	Holes with T02 size at these coordinates.		
X8.01Y2.85			
X6.54Y2.85			
X6.45Y4.8			
T03	Set current tool to T03	Create a slot	
G00X8.01Y3.825	Set rout mode and move T03 to XY		
M15	Tool down		
G01X6.54Y3.825	Linear rout to the coordinate		
M16	Tool up		
T04	Select tool T04	Rout	
G00X5.0Y2.6	Set rout mode and move T03 to XY		
M15	Tool down		
G03X6.0Y1.6A1.0	Circular CCW rout with radius A.		
G01X11.0Y1.6	Linear rout		
G01X11.0Y5.0	Linear rout		
G03X10.0Y6.0A1.0	Circular CCW rout		
G01X5.0Y6.0	Linear rout		
G01X5.0Y2.6	Linear rout		
M16	Tool up		
G05	Set drill/rout mode to drill.	Drill mode	
T03	Select tool T03	Create 1 hole	
X8.0Y8.0	A hole of size T03		
M30	End of file		

The corresponding drill/rout image is:

2.9 Convention for syntax rules

The syntax is expressed in Backus-Naur form:

- Syntax rules are written with bold font,
  <Elements set> = {<Elements>}
- Optional items enclosed in square brackets,
  [<Optional element>]
- Items repeating zero or more times are enclosed in braces,
  <Elements set> = <Element>{<Element>}
- Alternative choices are separated by the '|' character,
  <Option A>|<Option B>
- Grouped items are enclosed in regular parentheses,
  (A|B)(C|D)<NL> = New line

Examples of XNC file content are written with mono-spaced font, e.g. `X0Y0`

2.10 Overall file Backus-Naur form

The Backus-Naur form below describes the structure of the complete file using the commands as primitives. The Backus-Naur form of the individual commands in terms of the character set is given in section 3.

```
<XNC file>      = <header><body><end of file>
<header>        = <start of header><set unit><tool table><end of
header>
<tool table>    = {<tool declaration>}
<body>          = {(<drill section>|<rout section>)}
<drill section> = <set drill mode>{<select tool>{<drill hit>}}
<rout section>  = <set rout mode>{<select tool><tool down>{rout}<tool
up>}
<rout>          = (<linear rout>|<CW rout>|<CCW rout>)
```

To avoid cluttering this specification it excludes comments, which can be inserted anywhere, before the header, in the header and in the body.

3 Commands

3.1 Comment (;)

This command is used for human readable comments.

The syntax is:

```
<comment> = ;<Comment content><NL>
```

The `<Comment content>` must be strings and do not have the (;) character. Strings can be maximally 255 characters long. Comments are allowed anywhere in the file.

 Example:

```
; This is a comment
```

3.2 Start of header (M48)

M48 defines the beginning of the file header. This command must appear on the first line of the header.

The syntax is:

```
<start of header> = M48<NL>
```

3.3 Set unit

This command sets the unit of the XNC file. The unit is valid for the whole XNC file, both for coordinates and tool diameters. The possible values are INCH or METRIC. The metric unit is mm. It is recommended to use metric.

The set unit command must be used once and only once, in the header, before the first use of coordinates.

The syntax is:

```
<set unit> = (METRIC|INCH)<NL>
```

Syntax	Comments
`METRIC|INCH`	Unit used in the whole XNC file

3.4 Tool declaration (T#/C#)

The tool numbers and tool diameters must be specified in the header, after the unit definition.

The syntax is:

```
<Tool declaration> = T<tool number>C<tool diameter><NL>
<tool number> = <digit>[<digit>]
```

Syntax	Comments
`T`	Code for tool declaration
`<tool number>`	Coded with 2 digits from 01 to 99. It must be unique.
`C`	Code for diameter parameter
`<tool diameter>`	A positive decimal in the unit of the XNC file. It states the end diameter on the board - for plated holes this is the diameter *after* plating. Tool diameter is actually a misnomer.

3.5 End of header (%)

The character % defines the end of the header and the start of the body.

The syntax is:

```
<end of header> = %<NL>
```

3.6 Set drill mode (G05)

G05 sets the drill/rout mode to drill. In drill mode only round holes can be created.

The syntax is:

```
<set drill mode> = G05<NL>
```

 Example:

```
G05
```

3.7 Set rout mode (G00)

G00 sets the drill/rout mode to rout, and moves the current point to the coordinate.

The syntax is:

```
<set rout mode> = G00X<coordinate>Y<coordinate><NL>
```

 Example:

```
G00X8.01Y38.25
```

3.8 Select tool (T#)

Before creating any hole objects one of the tools in the tool table must be selected. Selecting an undefined tool is invalid.

The syntax is:

```
<select tool> = T<tool number>
```

Syntax	Comments
`T`	Code for tool selection
`<tool number>`	Number of the selected tool. Coded with 1 or 2 digits from 01 to 99.

Example:

```
T13
```

3.9 Drill hit

The drill hit command consists of a coordinate only. It creates a drill hole of the size of the selected tool at that coordinate. Drill hit commands can only be used in drill mode.

The syntax is:

```
<drill hit> = X<coordinate>Y<coordinate><NL>
```

Example:

Code	Comment
`X0.901Y3.3375`	X and Y coordinate
`X0.901Y4.3125`	

3.10 Tool down (M15)

This command starts a rout path. It can only be used in rout mode.

The syntax is:

```
<Tool down> = M15<NL>
```

Example:

```
M15
```

3.11 Tool up (M16)

This command ends a rout path. It can only be used in rout mode.

The syntax is:

```
<Tool up> = M16<NL>
```

 Example:

```
M16
```

3.12 Linear rout (G01)

Draws a linear rout with the current tool, from the current point to the coordinates in the command. After the rout is performed the current point is set to the values in the command.

The syntax is:

```
<Linear rout> = G01X<coordinate>Y<coordinate><NL>
```

 Example of a linear rout command:

```
G01X6.54Y8.54
```

 Example of a linear rout sequence

First the tool must be selected. Then the rout command G00 is specified followed by the X and Y coordinate of the start point of the rout. A M15 command set the tool down. A G01 command with the rout end point executes the linear rout. The M16 tool up command prepares for another rout.

Code	Comment
T01	Tool selection T01
G00X8.01Y3.825	G00 followed by X and Y coordinate of the rout start point
M15	Tool down command
G01X6.54Y3.825	G01 followed by the coordinate of the draw end point. This will be the start point of the next draw
G01X5.0Y4.5Y3.825	G01 followed by the coordinate of the draw end point
M16	Tool up command

Linear rout put the status in rout mode. Making a hole after the linear move needs to have the drill mode specified (See 3.6).

3.13 Circular clockwise rout (G02)

Circular clockwise rout with the current tool, from the current point to the coordinate, with the radius in the command, spanning an angle ≤ 180º. After the rout is performed the current point and current radius is set to the values in the command.

The syntax is:

<CW rout> = G02X<coordinate>Y<coordinate>A<radius><NL>

 Example of a command:

```
G02X6.54Y8.5004A2.5
```

 Example of a route circular rout sequence:

First a tool must be selected. G00 sets rout mode and moves the current point to its X and Y coordinate. A M15 command set the start of rout. G02 specifies the circular clockwise rout end point and its radius. M16 is the tool up command.

Code	Comment
T01	Tool selection T01
G00X5.05Y2.6	G00 followed by X and Y coordinate of the arc start point
M15	Tool down
G03X6.0Y1.6A1.0	G03 followed by the coordinate of the arc end point and radius
M16	Tool up command

3.14 Circular counter-clockwise rout (G03)

Circular counter-clockwise rout with the current tool, from the current point to the coordinate, with the radius in the command, spanning an angle ≤ 180º. After the rout is performed the current point and current radius is set to the values in the command.

The syntax is:

<CCW rout> = G03X<coordinate>Y<coordinate>A<radius><NL>

 Example:

```
G03X6.54Y8.5004A0.25
```

3.15 End of file (M30)

The M30 command indicates the end of the file.

The syntax is:

<end of file> = M30<NL>

 Example:

```
M30
```

The last command in an XNC file must be the M30 command. No data is allowed after a M30.

4 Attributes

4.1 Syntax and Semantics

Attributes add meta-information to an XNC file. These are akin to labels with additional information about the file or features within. Examples of such meta-information are:

- The span of the drill holes, e.g. they go from layer 1 to layer 2.
- Whether a drill hole is plated or not.
- Whether a drill hole is a via, a component hole, a mechanical hole.

This meta-information is essential in PCB fabrication data. The fabricator needs to know this information to fabricate the board. The attribute syntax provides a flexible and standardized way to add meta-information to XNC files, independent of the specific semantics or application.

Attributes do not affect the size and location of the holes. An XNC reader that ignores the attributes will generate the correct holes. Attributes are defined via *standardized comments*, this is comments starting with the string " #@*!* ". XNC readers must ignore all comments when generating the holes. Using standardized comments guarantees compatibility with legacy applications that do not support attributes: they will ignore the comments and provide the correct holes albeit without the meta-information. The content of standardized comments must follow this specification – this makes them machine-readable.

Example:

```
; #@! TF.FileFunction,Plated,1,4,PTH
```

This attribute indicates that the XNC file contains plated holes going from layer 1 to 4, which are plated through holes.

XNC attributes follow the scheme of the Gerber attributes. See the Gerber format specification for more information about Gerber attributes.

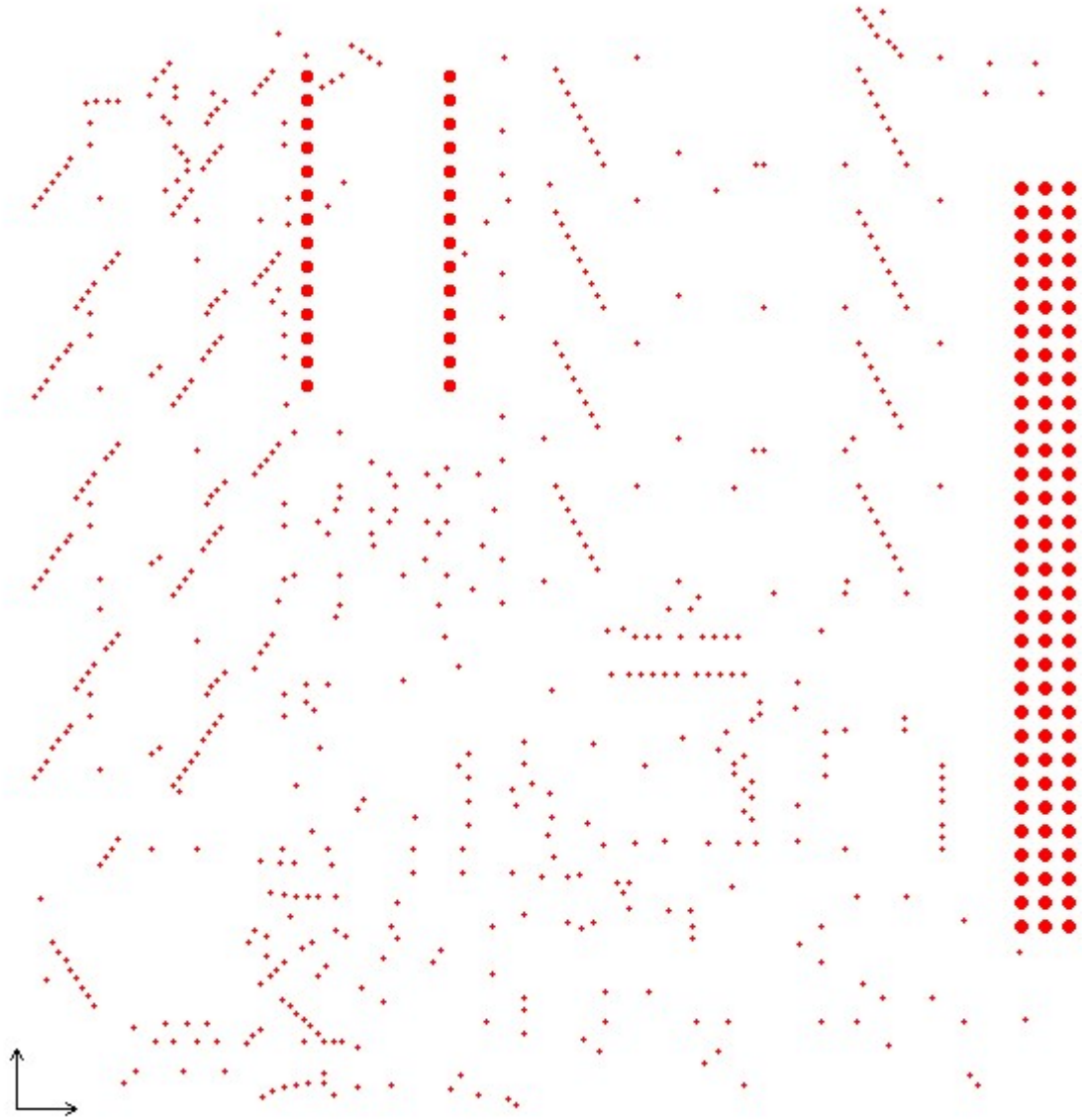
XNC attributes can be attached to the complete file, to tools or to individual holes. The translation from Gerber is obvious, Gerber file attributes become XNC file attributes, Gerber aperture attributes become XNC tool attributes and Gerber graphics object attributes become XNC hole object attributes. (As the Gerber format is far richer as the XNC format no all Gerber attributes are useful for XNC files.) The content of the XNC standardized comments follows the syntax of the Gerber attributes. XNC files are then compatible with Gerber X2, and allows to include XNC files in X2 data sets seamlessly, without loss of information. Furthermore, re-using existing attributes simplifies the tasks of software developers.

Note that XNC (and Gerber) indicate plating with a file attribute. This require that drill files are split between plated and non-plated. This attribute does not support mixed files, with both plated and non-plated holes, are not supported. There is a good reason for that. A plating attribute indicates the presence of 'vertical' copper. The whole concept of X2 is that file structure - the presence and absence of material - is indicated by file attributes because they are simpler than object attributes: there is no need to specify the plating of each individual tool. (As a comparison, one could ask for all copper layers to be combined in a single file and distinguish the layers by indicating to which layer each object belongs, but this is obviously silly.) Mixed files are not supported because they make it needlessly complex to provide full information on the presence of the materials.

4.2 An example XNC file with attributes:

Code	Comment	
M48	Beginning of header.	Header
; #@! TF.FileFunction,Plated,1,4,PTH	The file contains plated holes from layer 1 to 4 which are the PTH holes.	
; #@! TF.CreationDate,2018-11-23T15:59:51+01:00	ISO 8601 creation date of the file	
; #@! TF.GenerationSoftware,Ucamco,UcamX,2017.04	Generated by Ucamco's UcamX version 2017.4	
METRIC	Unit is METRIC (mm).	
; #@! TA.AperFunction,ViaDrill	The next tool(s) are for via holes	
T01C0.300	Mechanical drill of 0.3 mm	
; #@! TA.AperFunction,ComponentDrill	The next tool(s) are for component holes	
T02C0.900	Component hole of 0.9 mm	
T03C1.000	Component hole of 1.0 mm	
%	End of header.	
G05	Set drill/rout mode to drill	Body
T01	Select tool T01	
X008.150Y012.050	A hole with T01 size at XY	
X007.550Y013.100	Another hole at another XY	
…		
X103.950Y112.700	Last hole with T01 size at XY	
T02	Holes with T02 size at these coordinates.	
X030.790Y078.415		
X030.790Y080.955		
…		
X046.030Y078.415		
T03	Holes with T03 size at these coordinates.	
X107.300Y043.490		
X107.300Y040.950		
…		
X109.840Y089.210		
M30	End of file	

The corresponding image is:



5 Revisions

Rev 2019.02.04 - Initial version.

This subset was defined by Denis Morin, Karel Tavernier, Jean-Pierre Charras and Marius Matioc. This specification was written by Denis Morin and Karel Tavernier.

Rev 2019.02.16

Corrected typo's and made text improvements pointed out by Nicholas Meeker.