Introduction

The purpose of this document is to define the format for the KiCad schematic files. This format supersedes the old file format defined in the file "library_file_format_EN.odt.

This file format is based on the Specttra DSN lexer[1] work graciously coded and contributed by Dick Hollenbeck. For those not familiar with the DSN format, it is very similar to the LISP programming language in that keywords are delimited by braces. It is also known as an S-expression format. A keyword may have zero or more tokens that behave like parameters for the keyword. Tokens may be defined as additional keywords creating a programming language like structure. The decision to change to this format is to provide a more robust file lexer, a human readable file format, better file error reporting, and improved extensibility. The other driving factor is to provide a more robust format for handling the various requirements of schematic designers.

Front Matter

This is a living document designed to be kept synchronized with the actual source code. If you modify the lexer, please update this specification as courtesy to other developers. There are many third party tools that are used to generate, manipulate, and parse component library files. Keeping this document current is vital to developers of these tools.

Formatting

The DSN lexer enforces strict formatting requirements. Files must be encoded in UTF8 (ASCII) format. All lexer keywords are comprised of lower case letters, numbers, and underscores. Tokens may be any alphanumeric characters delimited by a white space character. Each new delimited level of keywords is indented to make files easier to read and edit with a plain text editor. Comments are supported as separate lines indented at the same level as the current keyword delimiter and begin with the # character. In line comments are not allowed.

Using this document

This document is nothing more than several well commented schematic files. You should always be able to copy the text from the opening delimiter "(" to the closing delimiter ")" into a file and open it with the schematic capture program Eeschema. It may not actually draw a useful schematic or schematic element but it should always be syntactically correct. The comments for a specific keyword always appears above the keyword it describes. Keywords defined within brackets [] are optional. A simple syntax coloring scheme has been employed to make this document more readable. Keywords are highlighted in blue, comments are highlighted in turquoise, and strings are highlighted in red.

Logical coordinates.

The next version of Eeschema will be based on a coordinate system using 100nm as the base unit. This will allow for a 429.5 meter drawing space. Internally units will be integer values which guarantees all connectable items (pins, wires, junctions, etc.) will be connected properly. The file format units will be in millimeters which is the same as the board and footprint library file formats.

---

[1] See source files dsnlexer.cpp and dsnlexer.h in the KiCad source code for more information about how the DSN lexer works.

An example file showing syntax for all schematic elements.

```
# The schematic element is the root s-expression token used to define the schematic file.

# File header:
#   version: the file version time stamp
#   host: the version of Eeschema used to generate the file.  Third party generators should not use Eeschema's signature.
(kicad_schematic (version "20191124") (host "eeschema" "6.0.0-rc2-147-g34c6393b7")

 # Globally unique ID used across projects.  In version 6, this is only meaningful in the root schematic.  Future versions
 # may use this to verify that sheet file names are still valid.
 (uuid "UUID")

 #Schematic page settings.  See PAGE_INFO::Format documentation for the page output formatting.
(paper "PAGE_SIZE")

 # All instances (paths in the legacy file format) for this sheet.  Initially will only cover the page number but can be expanded
 # to cover other per instance sheet information.
 (instances
  # All instances of the symbol path information for a given project .
  (project "NAME" (uuid PROJECT_UUID)
   (path "PATH/INSTANCE" (page NUMBER) ...)
   ...
  ) # End of instance of this sheet for the project.
  ... # Sheet instances for other projects.  This will allow for safe use of sheets across projects.
 )

 # The complete list of symbols used within this schematic.  This will remove the need for the symbol cache library and all
 # of the headaches it has caused.  It also ensures the schematic file is 100% portable.  Symbols save here cannot use
 # inheritance so they can be edited in place at some point in the future.
(lib_symbols
  (symbol...)
  (symbol...)
  (symbol...)
)

 # User definable property for this schematic.  Properties can be drawn on the schematic by defining an effect element for
 #  the property.  If no effect  is defined, the property is not displayed.  Properties can be used for items such as schematic
 #  version, author, company name, address, etc.
 #
 # TODO: specify property precedence with project and hierarchical properties.
[(property "NAME" "VALUE"
  (at X Y [ANGLE])
  [(effects
    # If a font is not specified, the default property font defined in the schematic editor is used.
    [(font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD])]

    # By default, all property text are justified left horizontally and bottom vertically.  Valid horizontal justification
     # values are center, right, and  left.  Valid vertical justification values are center, top, bottom.
```

```
   [(justify HORIZONTAL_JUSTIFY VERTICAL_JUSTIFY)]

   # If a color is not specified, the default property color defined in the schematic editor is used.
   [(color r g b a)]
  )] # end effects
)] # end property

# Defined variants for this schematic.
(variants
  ("NAME" "DESCRIPTION")
  ...
)

# The hinting system provides a method for passing useful information to external applications.  An example would
# be net classes passed to the board layout program via the net list or some other intermediate file.  All hints defined
# in the root schematic are applied to the entire schematic.
[(hints
  # Associate a net with a net class.
  (net NET_NAME NETCLASS_NAME [(color r g b a)])
)]

# Graphical elements.

# Either polyline or line can be used to define a line or series of graphic (not electrically connectable) lines.
(polyline or
  (pts (xy X Y) (xy X Y) (xy X Y) (xy X Y) (xy X Y))

  # Line widths are in units as defined above. Default line width if not defined.
  # Dash, dot, dash-dot, etc.  Solid line if undefined.
  # Line color.  Default color if not defined.
  [( stroke [(width WIDTH)] [(style LINE_STYLE)] [(color r g b a)])]

  # Valid fill types are none, background, and outline.
  [(fill FILL_TYPE [(color r g b a)])]
)


(text "This is the text that gets drawn."
  (at X Y [ANGLE])

  [(effects
    # If a font is not specified, the default graphic text font and color defined in the schematic editor is used.
    [(font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD]) [(color r g b a)]]

    # By default, all graphic text objects are justified left horizontally and bottom vertically.
    [(justify HORIZONTAL_JUSTIFY VERTICAL_JUSTIFY)]

    # By default, graphical text objects are visible.  If for some reason you wish to hide the text, use the hide token.
    [hide]
  )]
```

```
 [(fill FILL_TYPE)]
)

# Define a graphic image.
# Type, alpha, vertical scaling, and external file linking are for future expansion.
(image [TYPE] (at X Y [ANGLE])  [(alpha VALUE)] [(scale HORIZONTAL VERTICAL)]
 (data IMAGE_DATA) | (file IMAGE_FILE_NAME)  # Possibly link to external image file in future.
)
```

Future graphical objects. Drawing tools currently do not exist for these objects. #####.

```
# Define a graphic rectangle.
(rectangle (start X Y) (end X Y)
 [(stroke WIDTH [(color r,g,b,a)])
 [(fill FILL_TYPE [(color r,g,b,a)])]
 )
# Define a graphic circle.
(circle (center X Y) (radius LENGTH)
 [(stroke WIDTH [(color r,g,b,a)])
 [(fill FILL_TYPE [(color r,g,b,a)])]
)

(arc  (start X Y) (end X Y) (radius (at X Y) (length LENGTH) (angles START_ANGLE END_ANGLE))
 [(stroke WIDTH [(color r,g,b,a)])
 [(fill FILL_TYPE [(color r,g,b,a)])]
)

(bezier
 (pts (xy X Y) (xy X Y) (xy X Y) (xy X Y) (xy X Y) (xy X Y))
 [(stroke WIDTH [(color r,g,b,a)])
 [(fill FILL_TYPE [(color r,g,b,a)])]
)

# Shape objects combine multiple single drawing elements into a closed form
# that can optionally include a fill.  Missing segments where XY points do not overlap
# will be filled with a straight line segment that will be saved to the file on the next save
(shape

 # line, arc and bezier elements may be repeated here as many times as needed
 # When saving, Eeschema will begin with the upper left most point and continue to
 # the end of the element with the closest end point, stepping through the full compound object.
 # If the two elements share both endpoints, then which is used will be undefined.
 # The last xy coordinate listed in the compound object will be the same as the first xy coordinate.
 # Overlapping compound objects are allowed.
 [(polyline (pts (xy X Y) (xy X Y) ... )]

 # start and end are the two  end points of the arc
 # mid is a point on the circumference, half-way between start and end
 [(arc (start X Y) (mid X Y) (end X Y)  )]
```

```
 [(bezier (pts (xy X Y) (xy X Y) (xy X Y) (xy X Y) ) ]

 # The stroke width is required.  Line type defaults to solid when undefined.  The
 # supported line types are solid, dash, dot, and dash_dot.  More line types can be
 # added in future iterations.  The line color defaults to the line color defined in
 # the editor.
 [(stroke (width WIDTH) [(type LINE_TYPE)] [(color r g b a)])]

 # Valid fill types are outline and background.  No fill if undefined.
 # Fill colour defaults to the fill colour defined in the editor.
 [(fill FILL_TYPE [(color r g b a)])]
)
```

End future graphical objects.

Connectable elements.

```
# A symbol is an instance of a symbol from the "symbols" section  above defined by LIB_NAME or VALID_LIB_ID.
# Only parts which exist in the "symbols" section above can  be instantiated.
#
# The optional "lib_name" token is used to look up the symbol when a deviation from the original VALID_LIB_ID is required.
# The "lib_id" token must point to the original library symbol.  It is used to look up the symbol when "lib_name" is
#    not defined.
# The optional "mirror" token requires at least one axis (x or y) but can be both axes.
# The optional "unit" token is defined when the unit is anything other than unit 1 on symbols that have multiple units.
# The optional "convert" token is defined when the convert (DeMorgan) is anything other than convert 1 on symbols that
#    have multiple body styles.  Note, this can only be 2 because the currently KiCad only supports one convert.
# The optional "not_in_bom" token is used to exclude the symbol from the bill of material export.
# The optional "not_on_pcb" allows for bill of material only symbols that do not get added to the netlist.
(symbol [(lib_name "LIB_NAME" )] (lib_id "VALID_LIB_ID" ) (at X Y [ANGLE]) [(mirror x | y)] [(unit #)] [(convert #)]
 [not_in_bom] [not_on_pcb]

 (uuid UUID)

 # Apply effect to property name defined in the parts list table.
 [(property "NAME" "VALUE"

  # Position of the property relative to the position of the component.
  (at X Y [ANGLE])

  # If a font is not specified, the default property font defined in the schematic editor is used.
  [(font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD])]

  # By default, all component property object text is justified centered horizontally and vertically around it's position.
  [(justify HORIZONTAL_JUSTIFY VERTICAL_JUSTIFY)]

  # If a color is not specified, the default property color defined in the schematic editor is used.
  [(color r,g,b,a)]
```

```
   # The visibility state of the property can be changed by using the show and hide elements.  By default the
    # reference designator and value properties are visible and all other properties are hidden.
   [hide]
  )]
)

(wire (pts (xy X Y) (xy X Y)) [(width LINE_WIDTH)] [(color r,g,b,a)]])

(bus (pts (xy X Y) (xy X Y)) [(width LINE_WIDTH)] [(color r,g,b,a)]])

# Bus entries are only for bus to wire entries.  Legacy bus to bus entries will be converted to diagonal bus segments.
# Bus entry orientations are limited to '/' and '\'.
(bus_entry (at X Y ORIENTATION) [(width LINE_WIDTH)] [(color r,g,b,a)]])

(label "label" (at X Y [ANGLE])
  [(effects
    [(font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD])] [(color r,g,b,a)]
  )]
)

(global_label "label" TYPE (at X Y [ANGLE])
  [(effects
    [(font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD])] [(color r g b a)]
  )]
)

(hierarchical_label "label" TYPE (at X Y [ANGLE])
  [(effects
    [(font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD])] [(color r g b a)]
  )]
)

(junction (at X Y))

(no_connect (at X Y))

(bus_alias "NAME" (members "MEMBER1" "MEMBER2" ... )

# A sheet instantiates a circuit defined with the schematic.  It's purpose is to allow for simple and complex schematic
# hierarchies.  A complex hierarchy is defined by a schematic that uses a circuit more than once  in a schematic.  Sheet
# names must be unique within a schematic and cannot instantiate the circuit where they are defined.
# Sheets will support user defined properties at some point in the future.  The mandatory fields "ki_sheet_name" and
# "ki_sheet_file" replace the "F0" and "F1" definition in the legacy file format.
(sheet (at X Y) (size WIDTH HEIGHT)
  [(stroke (width WIDTH) [(type LINE_TYPE)] [(color r g b a)])]
  [(fill FILL_TYPE [(color r g b a)])]

  # This defines how and where the sheet name text is displayed.
  (property "ki_sheet_name" "SHEET_NAME"  (at X Y [ANGLE])
    [(effects
```

```
    [(font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD])]
    [(justify HORIZONTAL_JUSTIFY VERTICAL_JUSTIFY)]
    [(color r g b a)]
    [hide]
   )]

  # This defines how and where the sheet schematic file name text is displayed.
  (property  "ki_sheet_file" "SHEET_FILE_PATH"  (at X Y [ANGLE])
   [(effects
    [(font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD])]
    [(justify HORIZONTAL_JUSTIFY VERTICAL_JUSTIFY)]
    [(color r g b a)]
    [hide]
   )]
  )

  # The sheet pins are only specified in sheets that are not derived from another sheet and are immutable.  Labels
  # cannot be defined in derived sheets.
  [(pin "PIN_ NAME" ELECTRICAL_TYPE (at X Y ORIENTATION)
   [(effects
    (font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD] [(color r g b a)])
   )]
  )]

 # All sheet instances for the entire schematic.  This gets saved for all hierarchies and is only saved in the root sheet.  This
 # must be saved at the end of the root schematic to ensure the entire schematic is loaded before attempting to set schematic
 # symbol sheet path information.
 (sheet_instances
  (path PATH/INSTANCE (page "PAGE_NUMBER"))
 ) # Closes sheet paths

 # All symbol instances (paths in the legacy file format) for the entire schematic.  This gets saved for all hierarchies
 # and is only saved in the root sheet.  This must be saved at the end of the root schematic to ensure entire schematic
 # is loaded before attempting to set schematic symbol sheet path information.
 (symbol_instances
  (path PATH/INSTANCE (reference REFERENCE) [(unit UNIT)] [(value VALUE)][(footprint "LIB_ID")]

  # The variant system allows for creating bill of material variants.
  # If a variant name is not defined for a symbol, the default value of a given field is exported to the bill of material.
  # The "unused" token removes the symbol from the BOM.
  [(variant "NAME" [unused]
    ("PROPERTY_NAME" "VARIANT_VALUE")
    ... # All variants for this symbol path go here.
   )]  # Closes variant
  )  # Closes path
 ) # Closes sheet paths
) # Closes kicad_sch
```