Introduction

The purpose of this document is to define the format for the Kicad schematic symbol library files. This format supersedes the old file format defined in the file “library_file_format_EN.odt.

This file format is based on the Specctra 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 the component library developers.

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 a 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 character 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. Inline comments are not allowed.

Using this document

This document is nothing more than several well commented part 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 library part file editor or library part viewer in Kicad's schematic capture program EESchema. It may not actually draw a useful component or be a very useful part but it should always be syntactically correct. The comments for a specific keyword always appear 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 and therefore the library parts will be based on either a 10nm or 100nm internal coordinate system. This will allow for 42.95 meter or 429.5 meter drawing space. The tool framework will allow the schematic editor to provide snapping to symbol pins. This guarantees all

---

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

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.

Syntax for all drawable item types.

```
# File header:
# kicad_sym - defines the file as a kicad symbol library file.
# version - the version of the host program used to generate the symbol.
# host - the name and version of the host used to generate the symbol library file.
(kicad_symbol_lib (version “20171130”) (host “kicad_symbol_editor” "5.0.0-rc2-147-g34c6393b7")

 # The symbol element is the root of the S expression used to define a library symbol.
 # The symbol name is NAME where the name of the symbol is an escaped string.
 # Escaped strings will allow for more robust symbol names than using the file
 # name as the symbol name.
 (symbol “NAME”
  # Optional symbol library flags.
  # power - defines the symbol as power symbol and removes the old # special character behavior.
  # required - symbols defined in multiple unit symbol are required and should be checked by the ERC.
  # atomic - symbols declared as atomic must follow the rules outlined in the "Kicad Atomic Components" document
  #          and will be checked by the ERC accordingly.
  [power][required][atomic]

  # Library symbols can be inherited from another symbol by using the “extends” keyword. The LPID refers to a
  # symbol in this library only and will not include a library nickname.
  [(extends “LPID”)]

  # Extended symbols can have a defined unit name. Otherwise, the ordinal position of the symbol inside
  # a multiple unit symbol.
  [unit ”UNIT_NAME”]

  # For future expansion to ensure symbols can be assigned a UUID.
  [(uuid “UUID”)]

 # Change the anchor position to something other than 0, 0.
  [(anchor (at X Y))]

 # Set the global pin number text size, pin name text size, and pin name offset.
  [(pin_numbers hide)] [(pin_names [(offset OFFSET)] [hide])]

 # Set the include in BOM and include in board. This only applies to root symbols.
 (in_bom yes|no)(on_board yes|no)

  # The “Reference” property is required and reserved. Optional reserved properties are “Value”, “Footprint”,
  # “Datasheet”, “ki_fp_filters, and ki_keywords. Except for the original four mandatory fields, property
  # names used by KiCad are prefixed with “ki_” to prevent existing user defined field name clashes.
  [(property “NAME” “VALUE”

   # The index required to handle the legacy mandatory fields.
   (id INDEX_NUMBER)
```

```
 # Position requires an X and Y coordinates.  Angle is in degrees and defaults to 0 if not defined.
 (at X Y [ANGLE])

 # Effects define how a property is displayed.  Effects can also be used to
 # override the effects of a property in an inherited symbol.  When used to
 # override the effects of an inherited property, the PROPERTY element must
 # be defined.
 [(effects

   # The FONT value needs to be defined.  Currently, EESchema does not support
   # different fonts.  In the future this feature may be implemented and at
   # that time FONT will have to be defined.  Initially, only the font size and
   # style are required.  Italic and bold styles are optional.  The font size
   # height and width are in units yet to be determined.
   (font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD] [(color r g b a)])

   # Visible by default.
   [hide]
 )] # end property
) # end symbol

# 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

 # 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)])]

 # 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)])]

 # 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.
 [(line (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) ) ]
```

```
)

 # Polylines a line or series of lines.
 # A polyline with two points is assumed to be line.
 # The LINE_TYPE specifies the type of line such as dot, dot-dash, dash, etc.  No LINE_TYPE specifies a solid line.
 (polyline
  (pts (xy X Y) (xy X Y) (xy X Y) (xy X Y) (xy X Y))

  # The line width is required.  Line type defaults to solid when undefined.
  # The line color defaults to the line color defined in the editor.
  [stroke (width WIDTH)] [(type LINE_TYPE)] [(color r g b a)]
 )

 # The rectangle token is only maintained for handling legacy file formats.  Once the
 # polygon support is implemented, all new file format code should use polygons.
 (rectangle (start X Y) (end X Y)
  [(stroke (width WIDTH) [(type LINE_TYPE)] [(color r g b a)])]
  [(fill (type FILL_TYPE) [(color r g b a)])]
 )

 (circle
  (center X Y)

  # Radius length is in units if defined or mils.
  (radius LENGTH)
  [(stroke (width WIDTH) [(color r g b a)])]
  [(fill (type FILL_TYPE) [(color r g b a)])]
 )

(arc (start X Y) (mid X Y) (end X Y)
  [(stroke (width WIDTH) [(color r g b a)])]
  [(fill (type FILL_TYPE) [(color r g b a)])]
)

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

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

  # Valid horizontal justification values are center, right, and left.
  # Valid vertical justification values are center, top, and bottom.  Default
  # justification is centered for both horizontal and vertical justification.
  [(effects
    (font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD] [HORIZONTAL_JUSTIFY]
     [VERTICAL_JUSTIFY] [mirror] [(color r g b a)] [hide])]
    )
   )]
```

```
 )

# A pin's type is its electrical connection.  Valid connection types are
# input, output, bidirectional, tristate, passive, unspecified, power_in,
# power_out, open_collector, open_drain, emitter_follower, sourc_follower
# unconnected.  Valid pin shape values are none, line, inverted, clock,
# inverted_clk, input_low, clock_low, falling_edge, and non_logic
# The optional SCOPE is used to specify if a pin label should be connected
# to all nets in a schematic by the pin name.  The only valid setting for
# SCOPE is “global”.  When SCOPE is not defined the connection is local to
# the pin connection point.
(pin TYPE SHAPE [SCOPE]

 # Pin coordinates must be integers.  The angle values 0, 180, 90, and 270
 # replace the current right, left, up, and down specifiers respectively.
  # No other angles are valid.
 (at X Y [ANGLE])

 # Length of the pin in units.  The pin length can be non-integer.  If no pin name
 # or pin number offset and/or angle is provided, the default pin name and number
 # offsets are used.
 (length LENGTH)
 (name “NAME” [(at X Y [ANGLE])]
  [(effects
    (font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD] [(color r g b a)])
    [hide]
  )]
 )
 (number “NUMBER” [(at X Y [ANGLE])]
  [(effects
    (font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD] [(color r g b a)])
    [hide]
  )]
 )
 (property “ATTRIBUTE_NAME” [ATTRIBUTE_VALUE]
 )
 [hide]

 # Alternate pin naming scheme allows a single pin to have a different name
 # pin type, and pin shape.  This allows for user to select a different pin
 # definition for symbols such as micro-controllers and FPGAs.  More than one
 # alternate is permitted.
 (alternate “ALT_NAME” TYPE SHAPE)
)

# Pin merge is used to group a number of pins that are connected to the same
# point internally and shows the list or an ellipsized list of pins .  It is
# typically used to hide power pins on symbols that have a large number of
# power connections such as FPGAs and micro-controllers.
# More than one pin_merge is permitted, but the same pin cannot be in more
```

```
  # than one pin_merge list.
  [(merge “NUMBER1” “NUMBER2” ...)]

  # The pin swap hinting flag is used to indicate to an external tool ( i.e. an
  # auto router ) that the pins defined are functionally equivalent and
  # interchangeable.
  # More than one hint_pin_swap is permitted.
  [(swap “NUMBER1” “NUMBER2” ...)]
)
```

Syntax for Inheritance.

```
(symbol NAME_HINT (extends "LPID")

  # Remove pin NUMBER defined in the base symbol.  Only valid for extended symbols.
  [(del “NUMBER”)]

  # Swap pin NUMBER1 and pin NUMBER2 in the base symbol.  Only valid for extended
  # symbols.
  [(swap “NUMBER1” “NUMBER2”)]
  # Change the number of pin NUMBER1 to NUMBER2 in the base symbol.  Only valid
  # for extended symbols.
  [(renum “NUMBER1” “NUMBER2”)]

  # Change the name of pin NUMBER to NAME in the base symbol.  Only valid for
  # extended symbols.
  [(rename “NUMBER” “NAME”)]

  # Remove property NAME from the base symbol.  Only valid for extended symbols.
  # If NAME is omitted, all inherited properties will be removed.
  [(property_del [“NAME”])]

  # Alternates are used to define multiple symbol-per-package devices and/or
  # alternate body styles (DeMorgan).  Symbols must be defined in order.  There
  # can more than one alternate defined.
  [(alternate "LPID" ["LPID"...])]

  # Add a new property to this symbol.  Also can be use to override a property
  # defined in the base symbol.
  [(property “NAME” “VALUE”
    [(effects (at X Y [ANGLE])
      (font [FONT] (size HEIGHT WIDTH) [ITALIC] [BOLD])
      [hide]]
    )]

  # The alternates swap hinting flag is used to indicate to an external tool
  # that the symbols defined in the alternates list are functionally equivalent
  # and interchangeable.
  [(swap "LPID" "LPID" ...)]
```

```
  )
)
```

7400 Dual Input NAND Gate Sample

```
# This is an example of a dual input NAND gate A of a 7400.
(kicad_symbol_lib (version "20171130") (host "kicad_symbol_editor" "5.0.0-rc2-147-g34c6393b7")
  (symbol "dual_input_nand_a" (unit "A")
    (property "Reference" "U" (id 0) (at 0 1.27))
    (arc (pos 0 0) (radius 3.81) (start 0 -3.81 ) (start_angle -90) (end 0 3.81) (end_angle 90) (fill background))
    (polyline (xy 0 3.81) (xy -3.81 3.81) (xy -3.81 -3.81) (xy 0 -3.81) (fill background))
    (pin input line (at -7.63 2.54 180) (length 3.81)
      (name "" (font (size 1.27 1.27)) (visible yes))
      (number "1" (font (size 1.27 1.27)) (visible yes))
    )
    (pin input line (at -7.63 -2.54 180) (length 3.81)
      (name "" (font (size 1.27 1.27)) (visible yes))
      (number "2" (font (size 1.27 1.27)) (visible yes))
    )
    (pin output line (at 7.63 0) (length 3.81)
      (name "" (font (size 1.27 1.27)) (visible yes))
      (number "3" (font (size 1.27 1.27)) (visible yes))
    )
    (hint_pin_swap "1" "2" )
  )
)

# Logic gate power symbol with a filled body.
  (symbol "logic_gate_power_body")
    (rectangle (start -5.08 7.62) (end 5.08 -7.62)(fill (type solid)))
    (pin power_in line (at 0, -7.62 5.08 90)(length 2.54)(name "GND") (number "7"))
    (pin power_in line (at 0 7.62 270)(length 2.54)(name "VCC")(number "14"))
  )

# Logic gate power symbol with a filled body.
  (symbol "logic_gate_power_no_body")
    (pin power_in line (at 0, 12.7 5.08 90) (length 5.08) (name "GND") (number "7"))
    (pin power_in line (at 0 -12.7 270) (length 2.54) (name "VCC") (number "14"))
  )

  # All LPIDs below this point assume that the base symbol is located in the same
  # library as the symbols that are derived from them.  See the "Distributed Library
  # & EESchema Symbols List Design Documentation" in the Kicad source for more
  # information on LPIDs.

  # This is the B gate of a 7400.
  (symbol "dual_input_nand_b" extends "dual_input_nand_a" (unit "B")
    (renum 1 4)
    (renum 2 5)
```

```
  (renum 3 6)
)

# This is the C gate of a 7400.
(symbol "dual_input_nand_c" extends "dual_in_nand_a" (unit "C")
  (renum 1 9)
  (renum 2 10)
  (renum 3 8)
)

# This is the D gate of a 7400.
(symbol "dual_input_nand_b" extends "dual_in_nand_a"  (unit "D")
  (renum 1 12)
  (renum 2 13)
  (renum 3 11)
)

# This is a DeMorgan representation of a dual input NAND gate A.
(symbol "dual_input_nand_demorgan_a" (unit "A")
  (property "ki_reference" "U" (at 0 1.27))
  (arc (pos -9.144 0) (radius 6.5532) (start -3.81 3.81) (end -3.81 -3.81) (fill))
  (arc (pos -1.1938 -1.3208) (radius 20.4228) (start 3.81 0) (end -0.6096 -3.81) (fill))
  (arc (pos -1.1938 1.3208) (radius 20.4228) (start 3.81 0) (end -0.6096 3.81) (fill))
  (polyline (xy -3.81 -3.81) (xy -0.635, 3.81 ))
  (polyline (xy -3.81 3.81) (xy -0.635, 3.81 ))
  # The poly line below is required to properly fill the shape.
  (polyline (xy -0.635 3.81) (xy -3.81 3.81) (xy -3.175 2.7178) (xy -2.7686 1.4986)
     (xy -2.6162 0.254) (xy -2.6924 1.0414) (xy -3.0226 -2.2606) (xy -3.556 -3.4036)
     (xy -3.175 -2.7178) (xy -3.81 -3.81) (xy -3.81 -3.81) (xy 0.635 -3.81) (fill))
  (pin input inverted (at -7.62 -2.54 180) (length 4.318) (name "") (number "1"))
  (pin input inverted (at -7.63 2.54 180) (length 4.318) (name "") (number "2"))
  (pin output line (at -7.63 0) (length 3.81) (name "") (number "3"))
)

# This is the DeMorgan representation of gate B of a 7400.
(symbol "dual_input_nand_demorgan_b" extends "dual_in_nand_demorgan_a"  (unit "B")
  (renum 1 4)
  (renum 2 5)
  (renum 3 6)
)

(symbol "dual_input_nand_demorgan_c" extends "dual_in_nand_demorgan_a" (unit "C")
  (renum 1 8)
  (renum 2 9)
  (renum 3 10)
)

(symbol "dual_input_nand_demorgan_d" (extends "dual_in_nand_demorgan_a") (unit "D")
  (renum 1 11)
  (renum 2 12)
```

```
  (renum 3 13)
 )

 # An example of putting it all together to create a 7400.
 (symbol "quad_dual_input_nand_gate"
  (value "quad_dual_input_nand_gate")
  (alternates
   "dual_in_nand_a"
   "dual_in_nand_b"
   "dual_in_nand_c"
   "dual_in_nand_d"
   "logic_gate_power_body"
  )
  (alternates
   "dual_in_nand_demorgan_a"
   "dual_in_nand_demorgan_b"
   "dual_in_nand_demorgan_c"
   "dual_in_nand_demorgan_d"
   "logic_gate_power_no_power"
  )
  (alt_swap
   "dual_in_nand_a"
   "dual_in_nand_b"
   "dual_in_nand_c"
   "dual_in_nand_d"
  )
 )

 # A 74LS00 is as simple as this.
 (symbol "74LS00" (extends "quad_dual_input_nand_gate")
  (property "Value" "74LS00" (id 1)  (at 0 -1.27))
 )
 # This is a so called "atomic" or "fully defined" symbol definition of a Texas
 # Instruments part number SN74HCT00NSR.
 (symbol "SN74HCT00NSR" atomic (extends "quad_dual_input_nand_gate/rev1")
  (property "Value" "SN74HCT00NSR" (id 1)  (at 0 -1.27))
  (property "Footprint" "Package_SO:SOIC-14_3.9x8.7mm_P1.27mm" (id 2) (at 0 0) (hide))
  (property "Datasheet"
   "http://focus.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=sn74hct00&fileType=pdf"
   (id 3) (at 0 0) (hide))
  (property "ki_keywords" "LOGIC NAND" (id 4) (at 0 0) (hide))
  (property "ki_description" "IC GATE POS-NAND QD 2IN 14-SOP" (id 5) (at 0 0) (hide))
  (property "Manufacturer" "Texas Instruments" (id 6) (at 0 0) (hide))
  (property "Vendor" "Digikey" (id 7) (at 0 0) (hide))
  (property "Vendor Part Number" "SN74HCT00NSR-ND" (id 8) (at 0 0) (hide))
 )
)
```