gEDA/gaf Symbol Creation Document

Ales V. Hvezda, ahvezda@geda.seul.org

July 6th, 2004

Contents

1 Overview

This document describes the creation of component symbols, including style conventions, and hints/tips and things to look out for when drawing symbols for the gEDA/gaf system.

2 Component symbol creation

Component symbols (from here on known as "symbols") are drawn using gschem just like drawing a schematic sheet. Here are the steps in a symbol in the gEDA/gaf system:

1. Run gschem and find a blank page or run: gschem filename-1.sym
2. Draw the symbol (see the style guide below for some conventions).
3. Translate the symbol to the origin using Edit/Symbol Translate...
   - Zoom in at least one step.
   - Make sure the snap is ON (this is critical).
   - Make sure grid snap size is set to 100 (this is critical).
   - Select "Symbol Translate..." or the press equivalent hotkey.
   - Enter 0 into the entry field and press OK.

   Translating the symbol to the origin is a required step. To translate a symbol elsewhere, enter a offset (in mils) which is a even multiple of 100. Make sure all pins are snapped to a 100 mil grid point.
4. Save the symbol using Save or SaveAs... Here are some symbol naming conventions:
   - Symbols are named: symbolname-#.sym
   - Symbols end with a .sym extension.
   - Symbols have a -# where # is a number. Typically # is 1 but if there are multiple symbols for a device then this number simply increments.
   - Symbol names are typically lowercase but letters which are part of a part number are uppercase.
   - The above case rule can be broken if the filename looks incorrect or wrong.
5. Place the symbol in one of the directories specified by the component-library keyword in the system-commonrc file. Once this is done, the symbol should be visible immediately and can be selected and placed with the "Add/Select Component..." menu item.

3 Requirements

This section describes the various requirements which must be met in order to create a valid symbol which will display and netlist in the gEDA/gaf system. Most of the requirements center around having certain attributes attached or inside the symbol.

Running `gsymcheck` will check that all of these requirements are met. `gsymcheck` will output fatal errors which are quite serious and must be corrected. `gsymcheck` will also output warnings on things which should be fixed but are not fatal.

For more information on the attributes presented here, please see the Master Attribute Document.

- **device**=DEVICENAME should be placed somewhere in the symbol and made invisible. **device**= is the device name and is required. Typically the devicename is in all caps (capital letters). This attribute should not be used as a label. Use a separate text object for the label. If the object is a graphic then **device**= should be set to none (**device**=none). It is no longer required to attach this attribute anything; just having it exist as **device**=DEVICENAME is good enough.

- **graphical**=1 should exist somewhere in a symbol which is purely graphical (such as a title block or decon symbol). Symbols which have this attribute have no electrical or circuit significance. Don't forget to set **device**=none.

- **description**=text should exist somewhere in the symbol. This attribute provides an one line description of the symbol.

- All pins should have a pair of attributes attached to them: **pinseq**=# and **pinnumber**=#. The first attribute, **pinseq**=# is just a sequence number and increments sequentially starting at 1. The second attribute **pinnumber**=# is the number of the pin. When a symbol is netlisted, the pin numbers are output in order of pin sequence. The pin number can be alphanumeric (i.e. like E or C).

- All pins should also have **pinlabel**=value attached to them. This attribute is the name or label of the pin (vs the pin number). This attribute is also used when a symbol is used in a hierarchical schematic. Please make this attribute green (instead of the default attribute yellow).

- All pins should also have **pintype**=value attached to them. This attribute describes the kind of a pin. Possible values are: in, out, io, oc, oe, pas, tp, tri, clk, pwr. Please see the Master Attribute Document for more info.

- If a component has multiple slots in a package (such as a 7400 (NAND) which has 4 NANDs per package) then you need a **numslots**=# attribute.

The # is the number of slots the device has. numslots= should be exist somewhere in the symbol and made invisible. Additional slot related required attributes are described below.

- If a component has multiple slots in a physical package then you also need to include a **slotdef**=#:#,#,#... for every slot. The first # corresponds to the slot number. If a device has 4 slots then there should be **slotdef**=1:..., **slotdef**=2:..., **slotdef**=3:..., and **slotdef**=4:..., attributes existing somewhere in the symbol and made invisible. The subsequent # have a one-to-one correspondence to **pinseq**=# attributes and specify what corresponding **pinnumber**=# should be when that slot is set. See The attached 7400-1.sym as an example of how this should all work.

- It is recommended that all symbols which have slots have a **slot**=1 attribute inside the symbol.

- **footprint**=PACKAGENAME should exist somewhere in the symbol which might be used with the PCB netlister. PACKAGENAME is the PCB footprint or package type like DIP14 or DIP40. Please see the **Footprint naming conventions** chapter for further detail. See also the PCB documentation and gnetlist/docs/README.pcb for more info on this attribute.

- You should put a **refdes**=U? attribute inside the symbol. Make only the value visible and it will be promoted (attached to the outside of the symbol (so it can be edited) when the symbol is placed in a schematic.

- The label= attribute should not be attached anywhere in the symbol. It is obsolete.

- The name= attribute should not be attached anywhere in the symbol.

- The netname= attribute should not be attached anywhere in the symbol. It is only used in schematics.

4 Style

This section describes the style in which is used in the standard gEDA/gaf symbol library.

4.1 Text

- All Text labels should all be 10 pt in size.

- Text (labels not attributes) should be color number 9 (text — green).

4.2 Attributes

- Pin numbers (which are attributes) should all be 8 pt in size.
- Attached attributes should be yellow. The color is set automatically to yellow if the text item is attached.
- The only exception to this is pinlabel= attributes, those should be color number 9 (text — green). If every text item within a symbol is yellow, the symbol looks too yellow.
- Attributes can be attached to some part of the symbol. Toplevel attributes (like the device= or net= attributes) used to be required to be attached to something to be attributes, but now they just have to exist in the symbol file as name=value.
- Expanding a bit on the last sentence, as long as the text item has the format name=value, it is considered an attribute. Attributes inside a symbol do not have to be attached to anything. In order to see hidden attributes in gschem select Edit/Show/Hide Inv Text.
- There is a symbol content versioning system in libgeda which is based on the **symversion**= attribute. Please see the Master Attribute Document for more information on using this versioning scheme.

4.3 Graphics

- Lines, boxes, arcs, and any other graphics should be color number 3 (graphic — green).
- Polarity bubbles should be color number 6 (logic bubble — cyan)
- If you are unsure on how to make a new symbol look or how big to make a new symbol, look at the existing symbols to get a feel for the appropriate appearance and size.

4.4 Pins

- Pins should all be 300 mils (3 grid spaces) long.
- For pins which are next to a logic bubble, make the pins 200 mils (2 grid spaces) long and then make the logic bubble 100 mils in diameter. In order to draw a 100 mil diameter circle, you will need to change the snap spacing to 50.
- A pin has two ends: one end has a red endpoint and one end that does not. The red endpoint is where nets can be connected. You can either rotate the pin so that this active end is in the right place or manually edit

the symbol file changing the "whichend" parameter on the pin object. See the File Format document for more info.

- Be that all endpoints of pins which are meant to be connected to are on the 100 mil grid. The endpoint which is not active can be off the grid if necessary.
- Pins should be color number 1 (pins — white).
- Leave 400 mils (4 grid spaces) between (vertically) pins, unless you are drawing a special symbol, then just try to make it look good.
- Pin number attributes should be 50 mils above (or below; which ever makes the most sense) the pin which they are attached to.
- Input pins belong on the left and output pins belong on the right of the symbol.
- Please do not mix inputs and outputs on the same side of the symbol, unless absolutely necessary.
- You can have pins on the top or bottom of a symbol.
- The order for rows of pins (buses) should be LSB (least significant bit) to MSB (most significant bit). When drawing pins which are part of a bus, make sure the LSB of the bus is at the top (or for pins on top/bottom of a symbol, left of the rest of the other pins). Look at 74/74181-1.sym for a correct example of this order (A0 on top through A3 and B0 on top through B3). Violating this rule will make connecting buses much more difficult.
- When placing pins on logic gates, be sure to place the smallest pin numbers toward the top (or left) and then increment going down (or across).

4.5 Electrical

- Do not draw power and ground pins. That information will be conveyed using attributes (see the netattrib document).
- The above rule can be broken if necessary, but keep in mind most of the standard library does not have power pins showing.
- Keep in mind, symbols are supposed to be symbolic, they do not represent the physical package that the device comes in.
- There is some disagreement on above, so this is okay too: Arrange the pins on a symbol logically so that they promote an uncluttered schematic. Note that this is frequently not the same pin arrangement as the physical device.

5 Footprint naming conventions

This section describes the conventions for naming of footprints used in gEDA/gaf.

The purpose of the naming convention is to establish a standard to maintain the same naming convention through the different phases of the CAD chain. This helps in ensuring that the collaborative effort of gEDA/gaf is not lost.

5.1 Notes

- Unless otherwise noted, numerical pin names will be used, starting from 1.
- n is for the pin count.
- m is for the pin spacing in mils.
- x is for the x dimension of the package (excluding pins). In particular this is used for the QFP package family.
- SMT means surface mount, other components are through-hole.

5.2 Integrated circuit packages

- Dual in line packages with up to 22 100 mil spaced pins and 300 mil row spacing are called `DIP`n.
- Dual in line packages with 24 or more 100 mil spaced pins and 300 mil row spacing are called `DIP`n`N`.
- Dual in line packages with 100 mil spaced pins and 400 mil row spacing are called `DIP`n`H`.
- Dual in line packages with 24 or more 100 mil spaced pins and 600 mil row spacing are called `DIP`n.
- Shrink dual in line packages with up to 24 70 mil spaced pins and 300 mil row spacing are called `SDIP`n.
- Shrink dual in line packages with more than 24 70 mil spaced pins and 400 mil row spacing are called `SDIP`n.
- Single in line packages with 100 mil spaced pins are called `SIP`n`N`. See also `JUMPER`, below.
- Zig-zag in-line package are called `ZIP`n.
- Plastic leadless chip carrier with pin socket are called `PLCC`n`X`.

5.3 Integrated circuit SMT packages

- Small outline SMT packages with up to 16 50 mil spaced pins and 150 mil total width are called `SO`*n*.
- Small outline SMT packages with more than 16 50 mil spaced pins and 150 mil total width are called `SO`*n*`N`.
- Small outline SMT packages with 50 mil spaced pins and 200 mil total width are called `SO`*n*`M`.
- Small outline SMT packages with up to 20 50 mil spaced pins and 300 mil total width are called `SO`*n*`W`.
- Small outline SMT packages with more than 20 50 mil spaced pins and 300 mil total width are called `SO`*n*.
- Small outline SMT packages with 44 or more 50 mil spaced pins and 525 mil total width are called `SO`*n*.
- Metric shrink small outline SMT packages with 0.65 mm spaced pins and 323 mil total width are called `MSSOP`*n*. *NOTE: To be confirmed.*
- Metric shrink small outline SMT packages with up to 44 0.65 mm spaced pins and 420 mil total width are called `MSSOP`*n*`W`.
- Metric shrink small outline SMT packages with over 44 0.65 mm spaced pins and 545 mil total width are called `MSSOP`*n*`W`.
- Shrink small outline SMT packages with 0.25 mil spaced pins and 420 mil total width are called `SSOP`*n*`W`.
- Quarter size small outline SMT packages with 0.25 mil spaced pins and 244 mil total width are called `SSOP`*n*.
- Thin small outline SMT packages with 0.2165 mil spaced pins and 535 mil total width are called `TSOP`*n*.
- Thin small outline SMT packages with 0.20 mil spaced pins and 795 mil total width are called `TSOP`*n*`A`.
- Thin small outline SMT packages with 0.20 mil spaced pins and 559 mil total width are called `TSOP`*n*`B`.
- Thin shrink small outline SMT packages with up to 28 0.26 mil spaced pins and 260 mil total width are called `TSSOP`*n*.
- Thin shrink small outline SMT packages with over 28 0.20 mil spaced pins and 319 mil total width are called `TSSOP`*n*.
- Ultra Super Mini SMT packages with up to 16 0.5 mm spaced pins are called `US`*n*.

- Plastic leadless chip carrier SMT are called `PLCC`n.
- Square quad-side flat pack SMT are called `QFP`n_x.
- Rectangular quad-side flat pack SMT are called `QFP`n`_R`.
- Square low profile quad-side flat pack SMT are called `LQFP`n_x.
- Square thin quad-side flat pack SMT are called `TQFP`n_x.
- Square Quad-side flat no-lead SMT without exposed paddle (back side contact) are called `QFN`n_x. Pin count is n and package size is x mm.
- Square Quad-side flat no-lead SMT with exposed paddle (back side contact) are called `QFN`n_x_EP. Pin count is n and package size is x mm.
- Thin profile square Quad-side flat no-lead SMT without exposed paddle (back side contact) are called `TQFN`n_x. Pin count is n and package size is x mm.
- Thin profile square Quad-side flat no-lead SMT with exposed paddle (back side contact) are called `TQFN`n_x_EP. Pin count is n and package size is x mm.
- Dual in line style crystal oscillators are `OSC8` and `OSC14`.
- 5 pin SOT SMT packages are `SOT25` and `SOT325`.
- 6 pin SOT SMT packages are `SOT26` and `SOT326`.

5.4 Basic semiconductors

- Axial diodes are called `ALF`m. Pin 1 is the cathode.
- Conventional through hole LED is `LED3` and `LED5` for 3 and 5 mm respectively. Pin 1 is plus. *NOTE: Should probably be changed to be in line with diode convention.*
- TO transistors are `TO5`, `TO92`, `TO126`, `TO220` etc. Suffixes may apply, e.g. `TO126W` is for wide, `TO126S` is for standing, `TO126SW` is for standing, wide.

5.5 Basic SMT semiconductors

- SOD diode SMT packages use their standard package name, e.g. `SOD80`, `SOD87`, `SOD106A`, `SOD110`. There are also `SOD123`, `SOD323` with narrow pads.
- SOT transistor SMT packages use their standard package name, e.g. `SOT23`, `SOT323`. There is also an `SC90`.

- SOT transistor SMT packages with numbering as for diodes (pin 1 is cathode, pin 2 anode) are `SOT23D`, `SOT323D`.
- 4 pin SOT SMT packages are `SOT89`, `SOT143`, `SOT223`.

5.6 Passive components

- Axial non-polar components (typically resistor, capacitor) are called `ACY`m.
- Bottom lead (radial) non-polar circular component (typically capacitor) is `RCY`m.
- Bottom lead non-polar rectangular component (typically capacitor) is `BRE`m.
- A standard crystal is `HC49`, or other HC designations as required.
- Single row 100 mil pin spacing jumpers are `JUMPER`n. The main difference compared to single in line package is the hole size.
- Dual row 100 mil spacing headers with DIP pin numbering are `HEADER`n`_1`. Note that n is an even number.
- Dual row 100 mil spacing headers with ribbon cable numbering are `HEADER`n`_2`. Note that n is an even number.
- Angled full header connectors with latches are `DIN41651_`n.
- Standing full header connectors with latches are `DIN41651_`nS.
- DSUB connectors female are `DB`n`F`.
- DSUB connectors male are `DB`n`M`.
- Female DIN card-to-card connectors are `DIN41612C`n`F`. Add `S` suffix for standing.
- Male DIN card-to-card connectors are `DIN41612C`n`M`. Add `S` suffix for standing.
- AMP modular RJ connectors with screen are `RJ11`, `RJ12` and `RJ45`.

5.7 Passive SMT components

- Standard SMT resistors, inductors, capacitors etc are `0201`, `0402`, `0603`, `0805`, `1206`, `1210`, `1806`, `1812`, `1825`, `2020`, `2706`.
- Tantalum SMT capacitors are `EIA3216`, `EIA3528`, `EIA6032`, and `EIA7343`. Pin 1 is plus.
- SMT electrolytics are designated by can diameter in 1/10 mm: `SME33`, `SME43`, `SME53`, `SME66`, `SME84`, `SME104`.

6 Hints and Tips

This section describes some hints and tips which will make your symbol creation experience easier.

- Avoid drawing things off of the grid. If you do, you cannot move the object(s) using the move command (if the grid is on) since the object will be snapped to the grid. [This was an old bug, which I think has been fixed, but avoid doing this anyway]. Use the symbol translate command instead (or move the object with grid snap off)
- If you need a finer grid then use Options/Snap Grid Spacing... to set a finer grid snap spacing. Just remember to set this back to 100 once you are ready to translate the symbol to the origin.
- If you want to translate a symbol from the origin to elsewhere, then use the "Symbol translate" command and enter a non zero number. Make sure this number is a multiple of 100 (ie 1000, or 1100).
- Pins **MUST** be snapped on the 100 spaced grid (at least the end which will have nets connected to it).
- Pins **MUST** be snapped on the 100 spaced grid (at least the end which will have nets connected to it). Yes this is line a duplicate. I can't stress this point enough.
- Remember that pins are special objects; if you want to add a pin, make sure it is a pin and not a line or net. Use the Add/Pin command to place a pin.
- Don't include nets or buses inside symbols. That is not supported and doesn't make much sense anyway.

7 Example

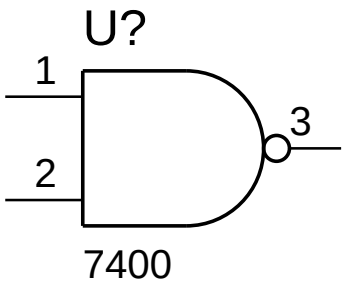
This section provides a simple example which tries to follow all of the above rules. This symbol is of a 7400 (NAND gate).

```
v 20020825
L 300 200 300 800 3 0 0 0 -1 -1
T 300 0 9 8 1 0 0 0
7400
L 300 800 700 800 3 0 0 0 -1 -1
T 500 900 5 10 0 0 0 0
device=7400
```

```
T 500 1100 5 10 0 0 0 0
slot=1
T 500 1300 5 10 0 0 0 0
numslots=4
T 500 1500 5 10 0 0 0 0
slotdef=1:1,2,3
T 500 1700 5 10 0 0 0 0
slotdef=2:4,5,6
T 500 1900 5 10 0 0 0 0
slotdef=3:9,10,8
T 500 2100 5 10 0 0 0 0
slotdef=4:12,13,11
L 300 200 700 200 3 0 0 0 -1 -1
A 700 500 300 270 180 3 0 0 0 -1 -1
V 1050 500 50 6 0 0 0 -1 -1 0 -1 -1 -1 -1 -1
P 1100 500 1300 500 1
{
T 1100 550 5 8 1 1 0 0
pinnumber=3
T 1100 450 5 8 0 1 0 2
pinseq=3
T 950 500 9 8 0 1 0 6
pinlabel=Y
T 950 500 5 8 0 1 0 8
pintype=out
}
P 300 300 0 300 1
{
T 200 350 5 8 1 1 0 6
pinnumber=2
T 200 250 5 8 0 1 0 8
pinseq=2
T 350 300 9 8 0 1 0 0
pinlabel=B
T 350 300 5 8 0 1 0 2
pintype=in
}
P 300 700 0 700 1
{
T 200 750 5 8 1 1 0 6
pinnumber=1
T 200 650 5 8 0 1 0 8
pinseq=1
T 350 700 9 8 0 1 0 0
pinlabel=A
T 350 700 5 8 0 1 0 2
```

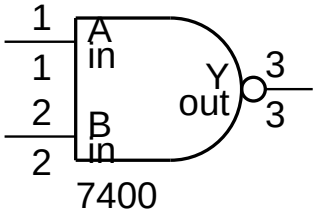
```
pintype=in
}
T 300 900 8 10 1 1 0 0
refdes=U?
T 500 2250 5 10 0 0 0 0
footprint=DIP14
T 500 2450 5 10 0 0 0 0
description=4 NAND gates with 2 inputs
T 500 2650 5 10 0 0 0 0
documentation=http://www-s.ti.com/sc/ds/sn74ls00.pdf
T 500 2850 5 10 0 0 0 0
net=Vcc:14
T 500 3050 5 10 0 0 0 0
net=GND:7
```

This example produces the following (using gschem):



This is the same symbol with all the hidden text visible (via Edit/Show/Hide Inv Text):

net=GND:7
net=Vcc:14
documentation=http://www-s.ti.com/sc/ds/sn74ls00.pdf
description=4 NAND gates with 2 inputs
footprint=DIP14
slotdef=4:12,13,11
slotdef=3:9,10,8
slotdef=2:4,5,6
slotdef=1:1,2,3
numslots=4
slot=1
U?:device=7400



8 Document Revision History

September 14th, 2002	Created symbol.tex from symbols.html
October 31st, 2002	Fixed bad example symbol
February 11th, 2003	Footprint naming conventions added
September 27th, 2003	Applied Dan McMahill's QFP and QFN patch
July 6th, 2004	Added a bunch more details/hints to the pin section