kicost Documentation

Release 0.1.35

XESS Corporation

Apr 24, 2017

Contents

Contents:

CHAPTER 1

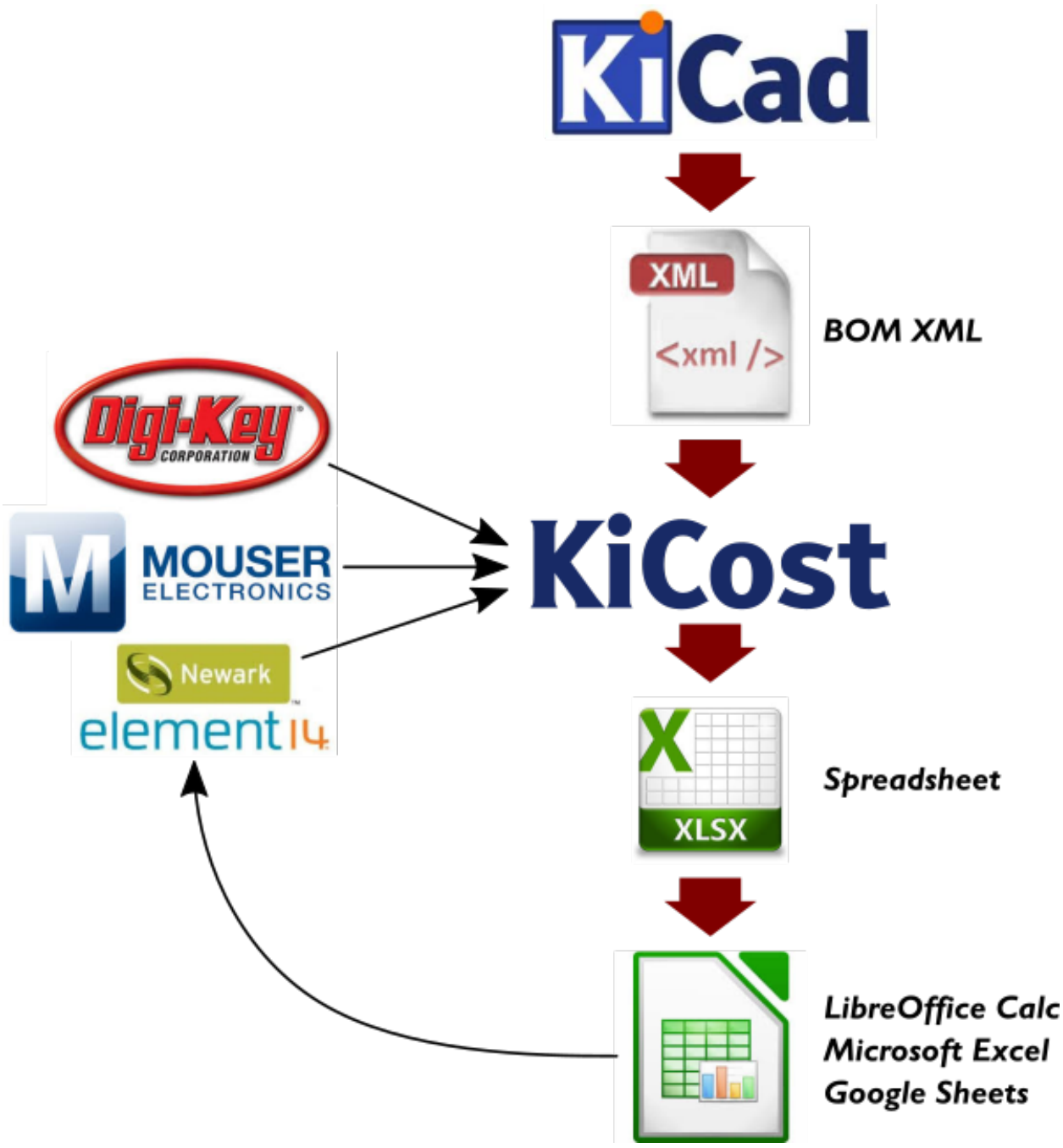
KiCost

KiCost is intended to be run as a script for generating part-cost spreadsheets for circuit boards developed with KiCad.

- Free software: MIT license
- Documentation: https://xesscorp.github.io/KiCost.

Features

- Processes the BOM XML file from your KiCad schematic to create a part-cost spreadsheet by scraping the web sites of several popular distributors for price and inventory data. (You can also enter your own quantity-adjusted pricing data for specialized parts or those not found at the supported distributors.)
- The spreadsheet contains quantity-adjusted pricing from each distributor for individual parts and the total board.
- Enter the number of boards to be built in a spreadsheet cell and all the pricing for the total board and individual parts is updated.
- The spreadsheet also shows the current inventory on-hand for each part at each distributor.
- Enter the quantity of each part that you want to purchase from each distributor and lists of part numbers and quantities will appear in formats that you can cut-and-paste directly into the website ordering page of each distributor.


KiCad
XML
<xml />
BOM XML
Digi-Key
CORPORATION
MOUSER
ELECTRONICS
Newark
element14
KiCost
XLSX
Spreadsheet
LibreOffice Calc
Microsoft Excel
Google Sheets

CHAPTER 2

Installation

At the command line:

```
$ easy_install kicost
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv kicost
$ pip install kicost
```

Note that if you install KiCost using `pip` on a Windows system running Python 2.7, using the default option that web scrapes with parallel processes may cause **MANY** errors. You can avoid this problem by:

- using `easy_install` to install KiCost, or
- use the `-s` KiCost option to serialize the web scraping.

CHAPTER 3

Usage

KiCost is mainly intended to be run as a script for generating part-cost spreadsheets for circuit boards developed with KiCad. Typical use is as follows:

1. For each part in your schematic, create a field called *manf#* and set the field value to the manufacturer's part number. (You can reduce the effort of adding this information to individual parts by placing the *manf#* field into the part info in the schematic library so it gets applied globally.) The allowable field names for part numbers are:

```
mpn          pn           p#
part_num     part-num     part#
manf_num     manf-num     manf#
man_num      man-num      man#
mfg_num      mfg-num      mfg#
mfr_num      mfr-num      mfr#
```

2. Output a BOM from your KiCad schematic. This will be an XML file. For this example, say it is *schem.xml*.

3. Process the XML file with KiCost to create a part-cost spreadsheet named *schem.xlsx* like this:

```
python kicost -i schem.xml
```

4. Open the *schem.xlsx* spreadsheet using Microsoft Excel, LibreOffice Calc, or Google Sheets. Then enter the number of boards that you need to build and see the prices for the total board and individual parts when purchased from several different distributors (KiCost currently supports Digi-Key, Mouser, Newark, Farnell and RS). All of the pricing information reflects the quantity discounts currently in effect at each distributor. The spreadsheet also shows the current inventory of each part from each distributor so you can tell if there's a problem finding something and an alternate part may be needed.

5. Enter the quantity of each part in your schematic that you want to purchase from each distributor. Lists of part numbers and quantities will appear in formats that you can cut-and-paste directly into the website ordering page of each distributor.

Custom Part Data

The price breaks on some parts can't be obtained automatically because:

- they're not offered by one of the distributors whose web pages KiCost can scrape, or
- they're custom parts.

For these parts, you can manually enter price information as follows:

1. Create a new field for the part named *kicost:pricing* in either the schematic or library.
2. For the field value, enter a semicolon-separated list of quantities and prices which are separated by colons like so:

   ```
   1:$1.50; 10:$1.00; 25:$0.90; 100:$0.75
   ```

   (You can put spaces and currency symbols in the field value. KiCost will strip everything except digits, decimal points, semicolons, and colons.)

You can also enter a link to documentation for the part using a field named *kicost:link*. The value of this field will be a web address like:

```
www.reallyweirdparts.com/products/weird_product.html
```

After KiCost is run, the price information and clickable link to documentation for the part are shown in a section of the spreadsheet labeled **Local**. If you want to associate the pricing and/or documentation link to a particular source or distributor, just place an extra label within the field key to indicate the source like so:

```
kicost:My_Weird_Parts:pricing
kicost:My_Weird_Parts:link
```

Then the pricing and documentation link for that part will appear in a section of the spreadsheet labeled **My_Weird_Parts**.

You can have as many sources for parts as you want, and a part may have multiple sources.

Part Grouping

KiCost groups similar parts together and places their information on a single line of the generated spreadsheet. For parts to be grouped, they must:

- come from the same library (e.g., "device"),
- be the same part (e.g., "R"),
- have the same value (e.g., "10K" but note that this **would not match** "10000" or "10K0"), and
- have the same footprint (e.g., "Resistors_SMD:R_0805_HandSoldering").

To reduce your effort, KiCost will also propagate pricing data among grouped parts. For example, if you place a hundred 0.1 uF decoupling capacitors in 0805 packages in a schematic, you need only assign a manufacturer's number and/or pricing data to one of them and it will be applied to the rest.

There are several cases that are considered when propagating part data:

- If only one of the parts has data, that data is propagated to all the other parts in the group.
- If two or more parts have data but it is identical, then that data is propagated to any of the parts in the group without data.

- If two or more parts in the group have *different* data, then any parts without data are left that way because it is impossible to figure out which data should be propagated to them.

It is possible that there are identical parts in your schematic that have differing data and, hence, wouldn't be grouped together. For example, you might store information about a part in a "notes" field, but that shouldn't exclude the part from a group that had none or different notes. That can be prevented in two ways:

1. Use the `-ignore_fields` command-line option to make KiCost ignore part fields with certain names:

```
kicost -i schematic.xml -ignore_fields notes
```

2. Precede the field name with a ":" such as `:note`. This makes KiCost ignore the field because it is in a different namespace.

Schematic Variants

There are cases where a schematic might need to be priced differently depending upon the context. For example, the price of the end-user circuit board might be needed, but then the price for the board plus additional parts for test also has to be calculated.

KiCost supports this with augmented the part field labels in the schematic in conjunction with the `--variant` command-line option. Suppose a circuit has a connector, J1, that's only inserted for certain units. If the manufacturer's part number field for J1 is given the label `kicost.v1:manf#`, then KiCost will ignore it during a normal cost calculation. But J1 will be included in the cost calculation if you run KiCost like so:

```
kicost -i schematic.xml --variant v1
```

In more complicated situations, you may have several circuit variants, some of which are combined in combination. The `--variant` option will accept a regular expression as its argument so, for example, you could get the cost of a board that includes circuitry for both variants 1 and 3 with:

```
kicost -i schematic.xml --variant "(v1|v2)"
```

Showing Extra Part Data in the Spreadsheet

Sometimes it is desirable to show additional data for the parts in the spreadsheet. To do this, use the `-fields` command-line option followed by the names of the additional part fields you want displayed in the global data section of the of the spreadsheet:

kicost -i schematic.xml –fields fld1 fld2

Parallel Web Scraping

KiCost spends most of its time scraping the part data from the distributor web sites. In order to speed this up, many of the web scrapes can be run in parallel. By default, KiCost uses 30 parallel processes to gather the part data. This can be too much for some computers, so you can decrease the load using the `--num_processes` command-line option with the number of processes you want to spawn:

```
kicost -i schematic.xml -num_processes 10
```

In addition, you can use the `--serial` command-line option to force KiCost into single-threaded operation. This is equivalent to using `-num_processes 1`. (If you encounter problems running KiCost on a Windows PC with Python 2, then using this command may help.)

Command-Line Options

usage: kicost [-h] [-v] [-i [file.xml]] [-o [file.xlsx]] [-f name [name ...]] [-var [VARIANT]] [-w] [-s] [-q] [-np [NUM_PROCESSES]] [-ign name [name ...]] [-d [LEVEL]] [-a] [-e dist [dist ...]] [–include dist [dist ...]]

Build cost spreadsheet for a KiCAD project.

optional arguments:

-h, --help show this help message and exit

-v, --version show program's version number and exit

-i [file.xml], –input [file.xml] Schematic BOM XML file.

-o [file.xlsx], –output [file.xlsx] Generated cost spreadsheet.

-f name [name ...], –fields name [name ...] Specify the names of additional part fields to extract and insert in the global data section of the spreadsheet.

-var [VARIANT], –variant [VARIANT] schematic variant name filter

-w, --overwrite Allow overwriting of an existing spreadsheet.

-s, --serial Do web scraping of part data using a single process.

-q, --quiet Enable quiet mode with no warnings.

-np [NUM_PROCESSES], –num_processes [NUM_PROCESSES] Set the number of parallel processes used for web scraping part data.

-ign name [name ...], –ignore_fields name [name ...] Declare part fields to ignore when grouping parts.

-d [LEVEL], –debug [LEVEL] Print debugging info. (Larger LEVEL means more info.)

-a, --altium Allows parsing of an Altium Designer .xml BOM file specified as input.

-e dist [dist ...], –exclude dist [dist ...] Excludes the given distributor(s) from the scraping process.

–include dist [dist ...] Includes only the given distributor(s) in the scraping process.

Adding KiCost to the Context Menu (Windows Only)

You can add KiCost to the Windows context menu so you can right-click on an XML file and generate the pricing spreadsheet. To do this:

1. Open the registry and find the *HKEY_CLASSES_ROOT* => *xmlfile* => *shell* key. Then add a *KiCost* key to it and, under that, add a *command* key. The resulting hierarchy of keys will look like this:

```
HKEY_CLASSES_ROOT
        |
        +-- xmlfile
              |
              +-- shell
                    |
                    +-- KiCost
```

```
                            |
                            +-- command
```

2. Set the value of the command to:

```
path_to_kicost -w -i "%1"
```

   For example, the command value I use is:

```
C:\winpython3\python-3.4.3\scripts\kicost -w -i "%1"
```

3. Close the registry. KiCost should now appear when you right-click on an XML file.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/xesscorp/kicost/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

Write Documentation

kicost could always use more documentation, whether as part of the official kicost docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/xesscorp/kicost/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *kicost* for local development.

1. Fork the *kicost* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/kicost.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv kicost
$ cd kicost/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 kicost tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/xesscorp/kicost/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_kicost
```

CHAPTER 5

Credits

Development Lead

- XESS Corporation <info@xess.com>

Contributors

- Oliver Martin: https://github.com/oliviermartin
- Timo Alho: https://github.com/timoalho
- Steven Johnson: https://github.com/stevenj
- Diorcet Yann: https://github.com/diorcety
- Giacinto Luigi Cerone https://github.com/glcerone

CHAPTER 6

History

0.1.35 (2017-04-24)

- Fixed bug in scraping RS website when a part search results in a list of matches instead of a single product page.

0.1.34 (2017-03-31)

- Fixed crash caused by uninitialized array in Digikey webscraping module.
- Place any available scraped part info into spreadsheet even if part is not available from a distributor.
- Removed unused imports from distributor modules.

0.1.33 (2017-02-23)

- Surround worksheet name with quotes in case it contains spreadsheet operators.
- Fixed extraction of product links from Farnell product tables.

0.1.32 (2017-02-14)

- Added options for including or excluding distributors.
- Updated web scrapers for various distributors.
- Added more debugging/logger statements.
- Updated some of the package requirements.

0.1.31 (2016-11-14)

- Giacinto Luigi Cerone added support for distributors Farnell and RS.

0.1.30 (2016-11-07)

- Manufacturer's part number field can now be labeled as 'manf#', 'mpn', 'pn', '#', etc. (See documentation.)
- Manufacturer field can now be labeled as 'manf' or 'manufacturer'.
- Distributor part number fields can now be labeled as 'digikey#', 'digikeypn', digikey_pn', 'digikey-pn', etc.

0.1.29 (2016-08-27)

- KiCost no longer fails if the <libparts>...</libparts> section is missing from the XML file.
- Documentation moved to Github Pages.

0.1.28 (2016-08-18)

- Fixed scraping of Digi-Key pages to correctly detect reeled parts and scrape alternate packaging options.

0.1.27 (2016-07-26)

- Fixed scraping of Digi-Key pages to correctly extract available quantity of parts.

0.1.26 (2016-07-25)

- Progress bar is explicitly deleted to prevent an error from occurring when the program terminates.

0.1.25 (2016-06-12)

- Contents of "Desc" field in component/library were being ignored when generating spreadsheet.

0.1.24 (2016-05-28)

- Fixed part scraping from Newark website.

0.1.23 (2016-04-12)

- Added progress bar.
- Added quiet option to suppress warning messages.
- 'manf#' and 'manf' fields are now both propagated to similar parts.

0.1.22 (2016-04-08)

- Extra part data can now be shown in the global data section of the spreadsheet by using the new `--fields` command-line option. This commit implements issue #8.

0.1.21 (2016-03-20)

- Parts with valid Digi-Key web pages were not appearing in the spreadsheet because they had strange quantity listings (e.g., input fields or 'call for quantities'. This commit fixes #36.

0.1.20 (2016-03-20)

- Prices of $0.00 were appearing in the spreadsheet for parts that were listed but not stocked. Parts having no pricing list no longer list a price in the sheet.
- Parts with short manf. numbers (e.g. 5010) were not found correctly in the distributor websites. The manufacturer name was added to the search string to increase the probability of the search finding the correct part.

0.1.19 (2016-02-12)

- Local parts weren't showing up in spreadsheet because of previous fix to omit parts that had no quantity field (non-stocked; not even 0). Fixed.

0.1.18 (2016-02-10)

- Made change to adapt to change in Digi-Key's part quantity field of their webpages.
- Omit parts from the spreadsheet that are listed but not stocked at a distributor.

0.1.17 (2016-02-09)

- Made changes to adapt to changes in Digi-Key's webpage format.

0.1.16 (2016-01-26)

- Added `--variant` command-line option for costing different variants of a single schematic.
- Added `--num_processes` command-line option for setting the number of parallel processes used to scrape part data from the distributor web sites.
- Added `--ignore_fields` command-line option for ignoring benign fields that might prevent identical parts from being grouped together.

0.1.15 (2016-01-10)

- Fixed exception caused when indexing with 'manf#' on components that didn't have that field defined.
- Replaced custom debug_print() with logging module.

0.1.14 (2015-12-31)

- When scraping a Digi-Key product list page, use both the manfufacturer's AND Digi-Key's number to select the closest match to the part number.

0.1.13 (2015-12-29)

- 'kicost:' can be prepended to schematic field labels to distinguish them from other app fields.
- Custom prices and documentation links can now be added to parts in the schematic.
- Web-scraping for part data is sped up using parallel processes.

0.1.12 (2015-12-03)

- Following the IP address mouser with redirect you to the nearest locale match, so the price will be in Euro if you are in Europe and the price decimal can be a comma.

0.1.11 (2015-12-02)

- Changed BOARD_COST field to UNIT_COST.
- Changed formatting of UNIT_COST field to make use monetary units.
- Changed format of debug messages.

0.1.10 (2015-10-08)

- Pushed lxml requirement back to 3.3.3 so linux mint would have fewer problems trying to install.

0.1.9 (2015-09-26)

- Fixed exception caused by Digi-Key part with 'call' as an entry in a part's price list.
- Fixed extraction of part quantities in Mouser web pages.
- Added randomly-selected user-agent strings so sites might be less likely to block scraping.
- Added ghost.py code for getting around Javascript challenge pages (currently inactive).

0.1.8 (2015-09-17)

- Added missing requirements for future and lxml packages.

0.1.7 (2015-08-26)

- KiCost now runs under both Python 2.7.6 and 3.4.

0.1.6 (2015-08-26)

- Mouser changed their HTML page format, so I changed their web scraper.

0.1.5 (2015-07-25)

- Corrected entrypoint in __main__.py.

0.1.4 (2015-07-09)

- Added conditional formatting to indicate which distributor had the best price for a particular part.
- Fixed calc of min unit price so it wouldn't be affected if part rows were sorted.

0.1.3 (2015-07-07)

- Added global part columns that show minimum unit and extended prices for all parts across all distributors.

0.1.2 (2015-07-04)

- Refactoring.
- To reduce the effort in adding manufacturer's part numbers to a schematic, one will now be assigned to a part if:
    1. It doesn't have one.
    2. It is identical to another part or parts which do have a manf. part number.

3. There are no other identical parts with a different manf. part number than the ones in item #2.

0.1.1 (2015-07-02)

- Fixed delimiter for Mouser online order cut-and-paste.

0.1.0 (2015-06-30)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search