Output generation automation

This document provides information on an automated output generation for Kicad. The automated output generation tool should only be accessible when opening a project, in the project window. This tool would allow the user to :

- Generate schematics PDF (from Eeschema)

- Generate BOM (from Eeschema)

- Generate ERC report (from Eeschema)

-Plot the board (pdf, gebers, dxf, …) (from pcbnew)

-Generate drill files (from pcbnew)

-Generate footprint position files (from pcbnew)

-Generate netlist (from pcbnew)

-Generate footprint report (from Pcbnew)

-Generate DRC report (from Pcbnew)

- Generate 3d STEP model of the board (from Pcbnew)

-Call python script. (The python API would not be the topic of this document)

Question : Is the BOM from Pcbnew relevant ? It contains less information than the one from Eeschema, and the we can access the one from Eeschema as the entire project should be open in order to use the automated output generation.

Generating output

User interface

In order to generate the output, the user simply selects the desired output generation configuration, and clicks the “build” button (Here the hammer icon).

To ease the use of this feature, Kicad would ship with some default configurations:

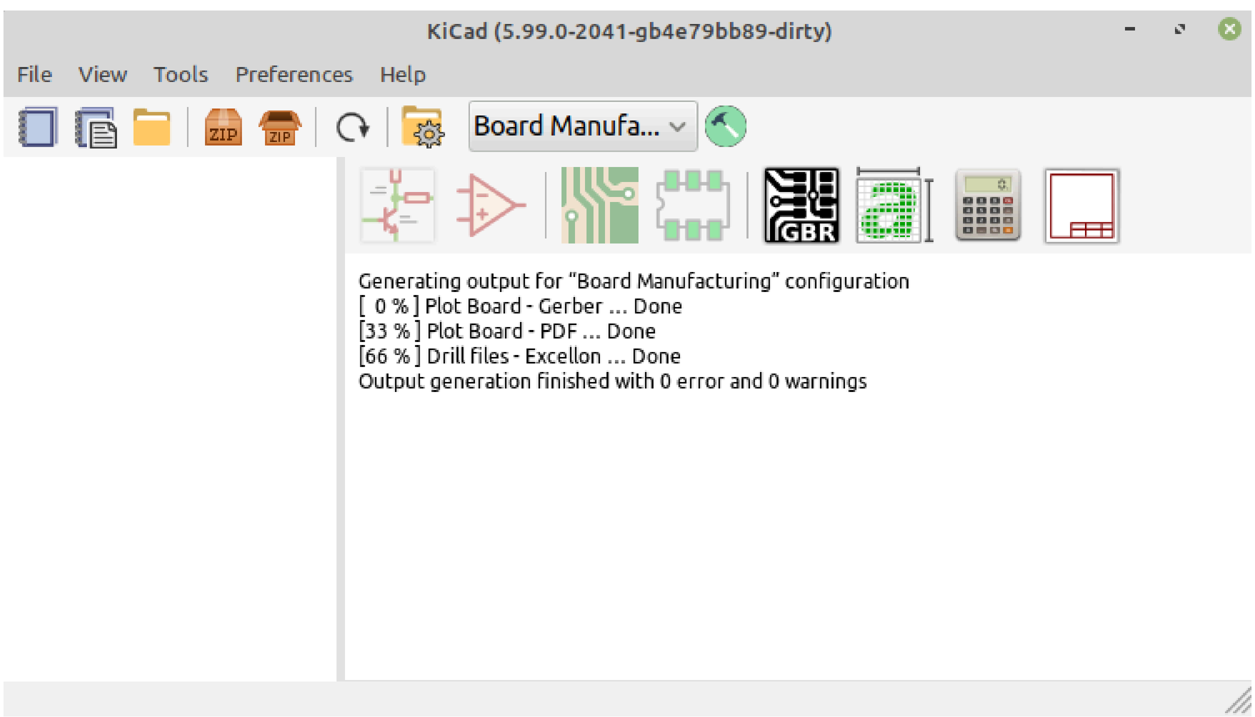
- “Project” (everything)

- “Board Assembly” (No drilling files, …)

- “Board manufacturing” (Nothing related to components)

Those are examples of output generation configurations the user could use, but the user can add more of them / edit existing ones.

source of the icon : https://www.flaticon.com/free-icon/hammer_236857



Feedback - Normal

The generation is sequential. When generating, for each document there is a line in the project window with an estimate of the completion percentage, the name of the step, and a status.

Generating output for “Board Manufacturing” configuration [0 %] Plot Board - Gerber … Done [33 %] Plot Board - PDF … Done [66 %] Drill files - Excellon … Done Output generation finished with 0 error and 0 warnings

Users can save this file text in a file, as part of the output generation. In this document it will be referred as “output report”

Feedback - Errors

In case of error (like a write protected output directory), the generation process continues, but errors are shown in the log. At the end of the generation, the user knows if the generation finished with or without errors.

Ideally, errors would be in red.

Generating output for “Board Manufacturing” configuration [0 %] Plot Board - Gerber … Done [33 %] Plot Board - PDF … ERROR - Can’t delete file “F.Cu.pdf” ERROR - Can’t delete file “B.Cu.pdf” [66 %] Drill files - Excellon … Done Output generation finished with 2 errors and 0 warnings

In addition to I/O errors the following errors are defined : (more could be added)

Error	**Condition for testing**
No schematics file	There is at least one output that requires the schematics file (schematics pdf, BOM, ...)
No board file	There is at least one output that require the board file (gerber, drill file, …)
Python script not found	Whenever we call a Python script
Python script error	When calling a python script. The python error log should be displayed.

Feedback - Warnings

Warnings are computed before the output generation, when possible . The following warnings are defined: (more could be added)

Warning	Condition for testing
Board and schematics not in sync	There are both a schematics file and a board file
ERC errors detected	There is at least one output that require the board file (gerber, drill file, …)
DRC errors detected	There is at least one output that require the board file (gerber, drill file, …)
Invalid netlist	There is a schematics file

Ideally, warnings would be in blue (or green)

```
Generating output for "Board Manufacturing" configuration
WARNING - DRC errors detected
[  0 % ] Plot Board - Gerber … Done
[33 % ] Plot Board - PDF … Done
[66 % ] Drill files - Excellon … Done
Output generation finished with 0 errors and 1 warnings
```

In case there are both warnings and errors, if there is a color distinction, the color of the last message should be the error color

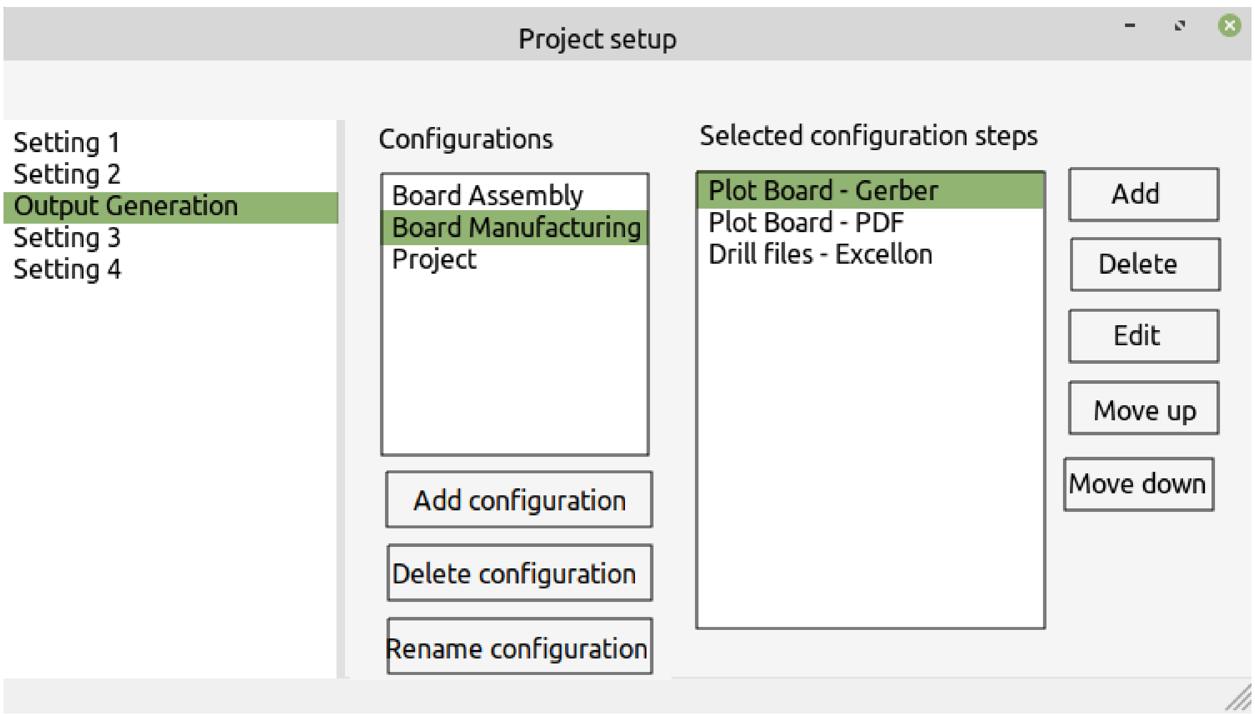
```
Generating output for "Board Manufacturing" configuration
WARNING - DRC errors detected
[  0 % ] Plot Board - Gerber … Done
[33 % ] Plot Board - PDF …
ERROR - Can't delete file "F.Cu.pdf"
ERROR - Can't delete file "B.Cu.pdf"
[66 % ] Drill files - Excellon … Done
Output generation finished with 2 errors and 1 warnings
```

Editing output configurations

Configuration edition window

To configure the output generation, the user goes in File > Project Setup > Output Generation.

The project setup then opens up and going to the Output Generation setting, the user sees the window below :



There are two main frames. If any of them is too small, sliders appear.

- "Configurations" : List of all configurations, this list is always alphabetically sorted.
- "Selected configuration steps" : List all generated documents by a given configuration. The highest document is generated first. There can be identical documents.

User interactions :

- clicking a configuration, the "Selected configuration steps" content changes.

- clicking "Add configuration", a prompt asks the user for the name, and then adds it in the configuration list. We also ask the user if he wants to create a project-only configuration, or a machine-wide one.The configuration steps are by default empty.

- Clicking “Delete configuration” deletes the configuration. A confirmation window appears. If the configuration is machine-wide, highlight the fact that it will be removed from all projects. Only works if a configuration is selected.
- Clicking “Rename configuration”, a prompt asks the user for the new name. Fails if a configuration already has the same name. If the configuration is machine-wide, highlight the fact that it will be renamed for all projects
- Clicking “Import configuration” opens the file explorer to ask for a configuration file. When importing, asks if the user wants to make ita project-only or a machine-wide configuration
- Clicking “Export configuration” opens the file explorer to ask for a file location/ name. This feature could be used by manufacturers to ensure the file formats / hierarchy.

- Clicking “Add”, a prompt asks the user for the entry name and the output type (gerber, drill files, schematics, …).
- Clicking “Delete” deletes the entry. A confirmation window appears.
- Clicking “Edit” opens the configuration window for that given document type. (ie : drill files and gerbers do not open the same window).
- Clicking “Move up” moves the selected entry up in the list. Changes the generation order
- Clicking “Move down” moves the selected entry down in the list. Changes the generation order

Adding a new generation step

When clicking “Add”, a pup op with a combo box opens. There should be a separator between Eeschema outputs, Pcbnew Output, and python scripts. The choices available are :

- Bill of Material
- Electrical Rule Check
- Schematic

- Board plot
- Design Rule Check
- Drill
- Footprint position
- Footprint report
- Netlist
- 3D model

- Python script

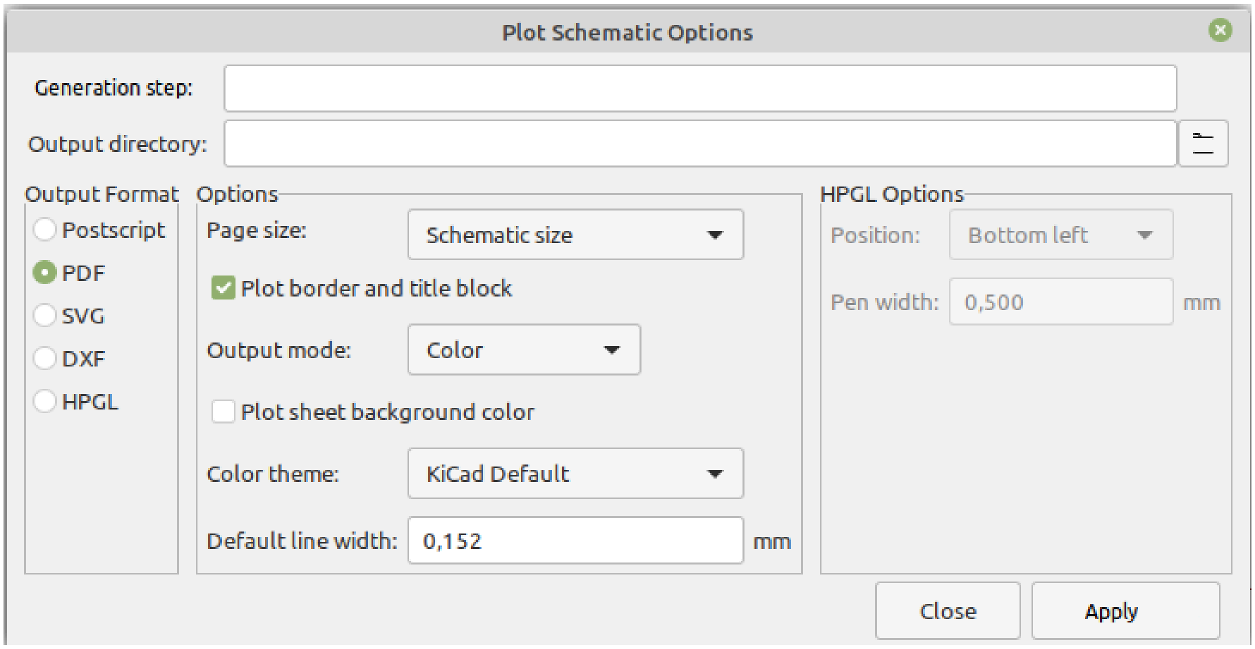
Question : Do we expand shorts like ERC, DRC, BOM ?

Configuration windows

When clicking “edit” for a configuration step, a configuration window opens. The content of the window depends on the nature of the configuration step. (editing a schematics steps will not lead to the same window as editing drill files steps).

In the next few sections, we will go through some window proposals. Most of them are windows that are already in Kicad, but some changes were made to fit the need.

Configure schematic window



We use the same "plot schematics options" window but we :

-Remove the "Output Messages" section

-Remove the "Plot Current Page" button

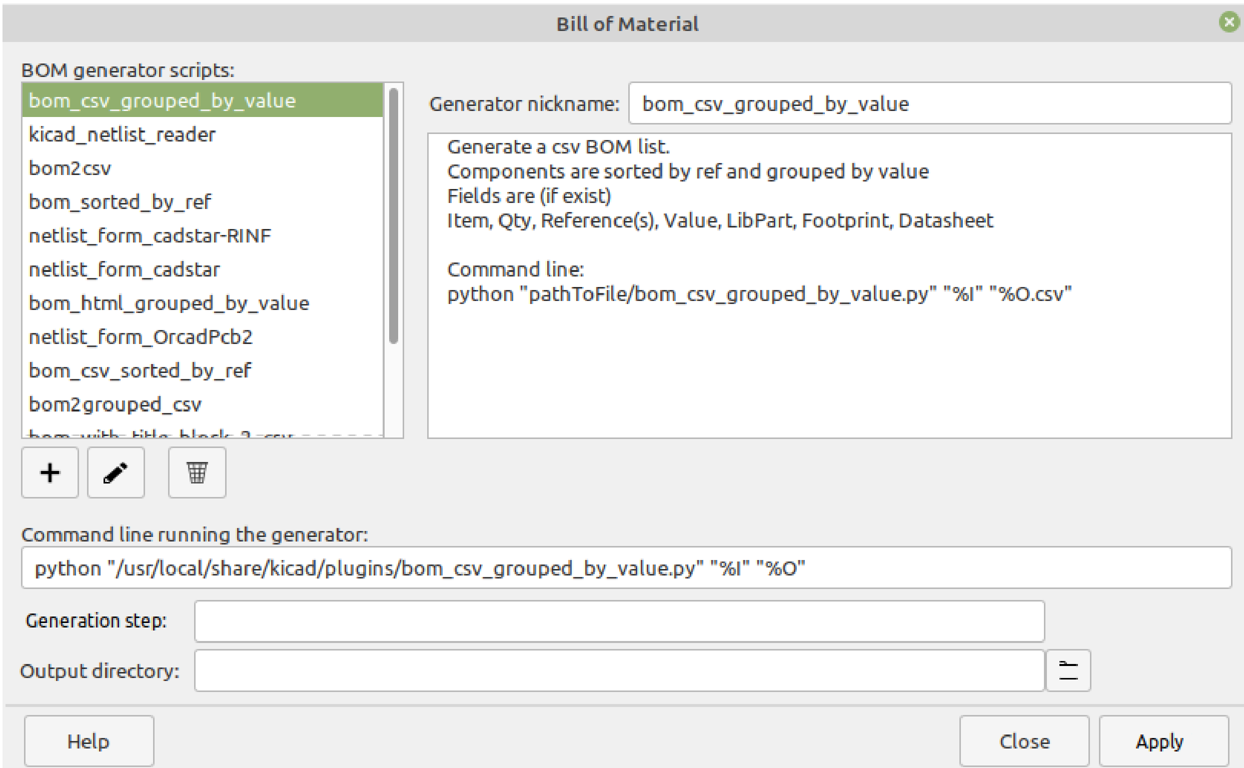
-Remove the "Plot All Pages" button

-Add a "Generation step" field which will appear in "Selected configuration steps" in the project setup window.

-Add an "Apply" button. Checks the unicity, for a given generation configuration, of "Generation step"

These changes do not apply to the window when called from Eeschema.

Configure BOM window



We use the same “Bill of material” window but we :

-Remove the “Generate” button

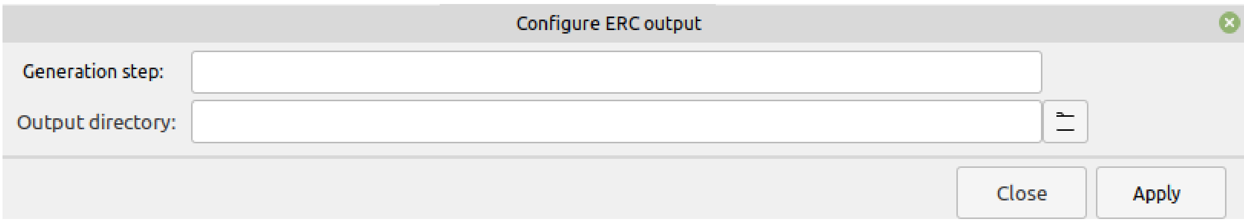
-Add an “Output directory” field.

-Add a “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-Add an “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”.

These changes do not apply to the window when called from Eeschema.

Configure ERC window



This is a new window with :

-An “Output directory” field.

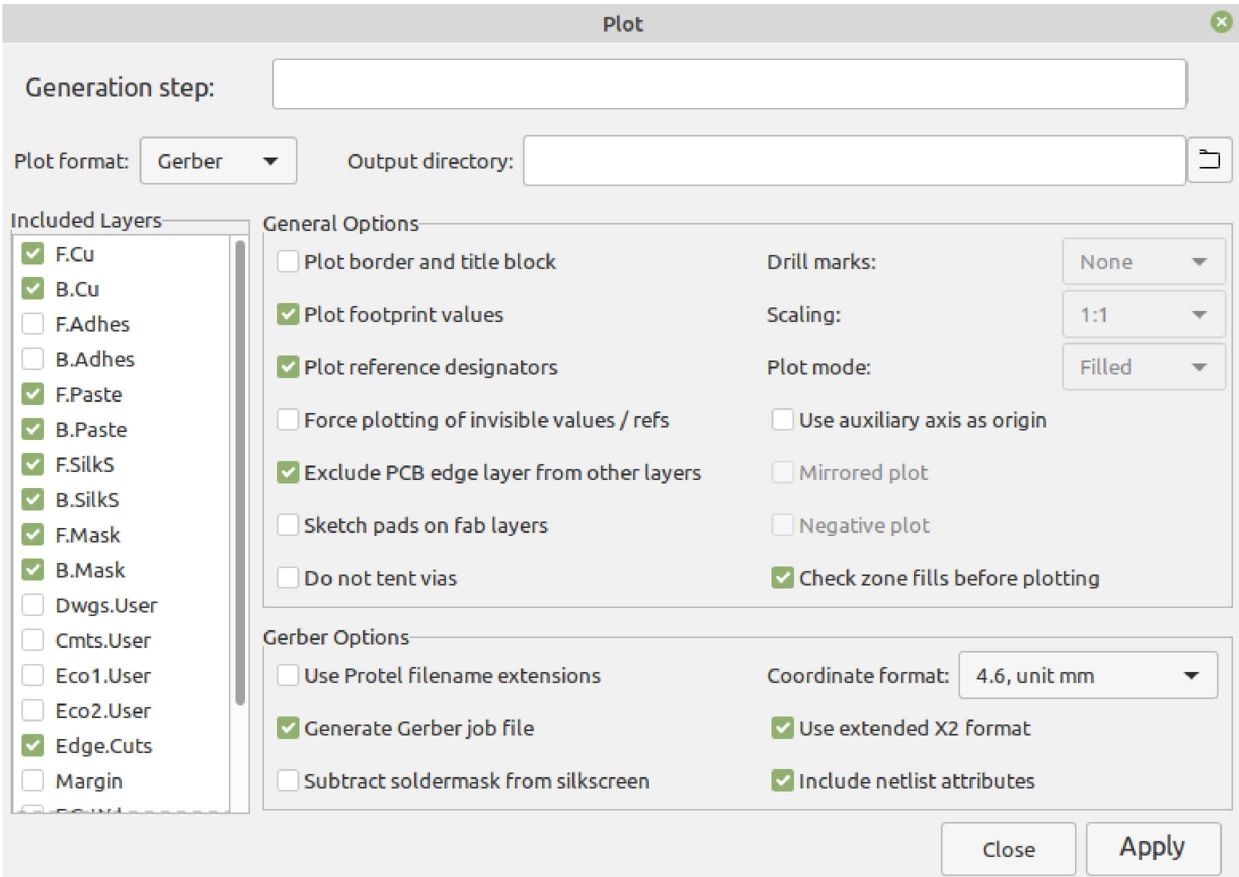
-A “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-An “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”.

-A “Close button” that closes the window and does not save any change.

All the ERC settings are retrieved from Eeschema.

Configure Board plot window



We use the same “Generate Drill Files” window but we :

-Remove the “Output Message” section

-Remove the “Run DRC” button

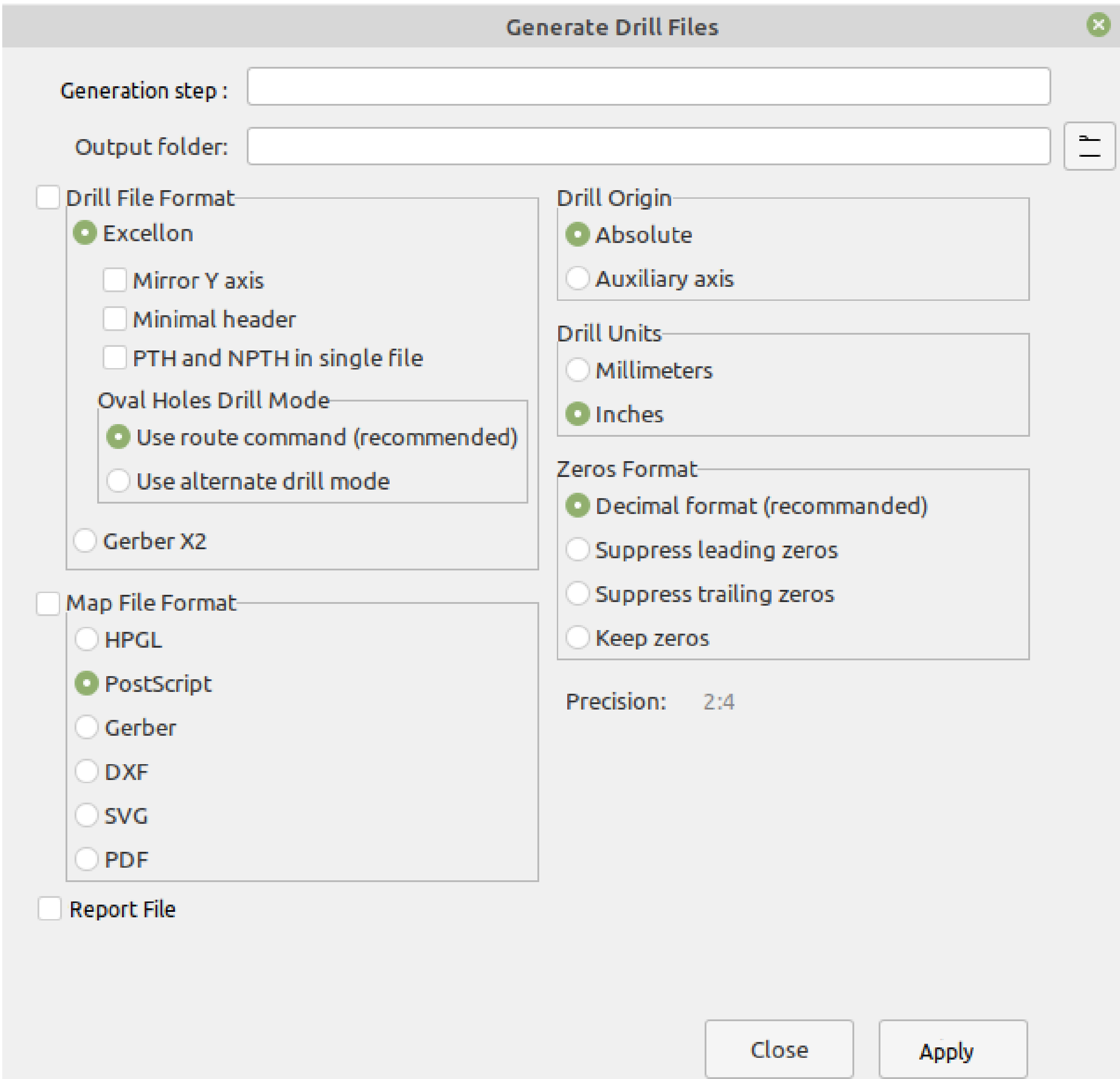
-Remove the “plot” button

-Add a “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-Add an “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”

These changes do not apply to the window when called from Pcbnew.

Configure Drill File window



We use the same “Generate Drill Files” window but we :

-Remove the “Messages” section

-Remove the “Hole Counts” section

-Remove the “Generate Report File” button

-Remove the “Generate Map File” button

-Remove the “Generate Drill File” button

-Add a “Report File” entry below “Map File Format”

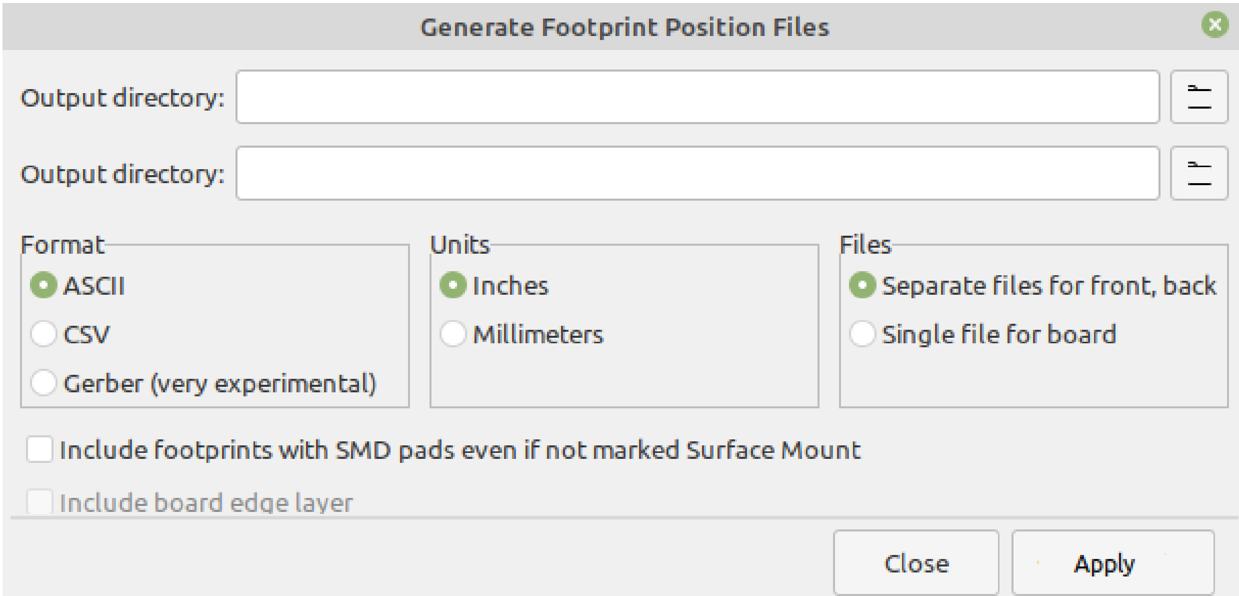
-Add checkboxes next to “Drill file format”, “map file format”, and “Report File”. When checked, a corresponding file shoul be generated (a single configuration entry can generate all three documents)

-Add a “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-Add an “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”

These changes do not apply to the window when called from Pcbnew.

Configure Footprint position files window



We use the same “Generate Footprint Position Files” window but we :

-Remove the “Output Messages” section

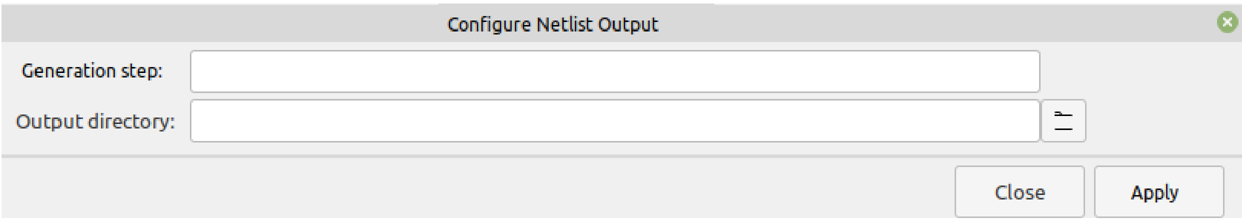
-Remove the “Plot Position Fille” button

-Add a “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-Add an “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”

These changes do not apply to the window when called from Pcbnew.

Configure Netlist window



This is a new window with :

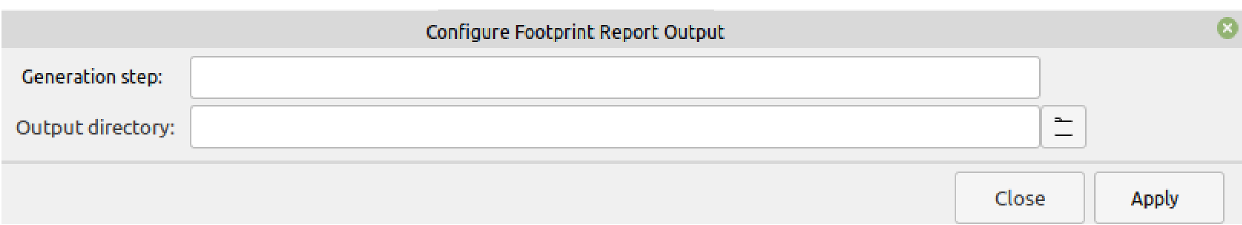
-An “Output directory” field.

-A “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-An “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”.

-A “Close button” that closes the window and does not save any change.

Configure Footprint Report window



This is a new window with :

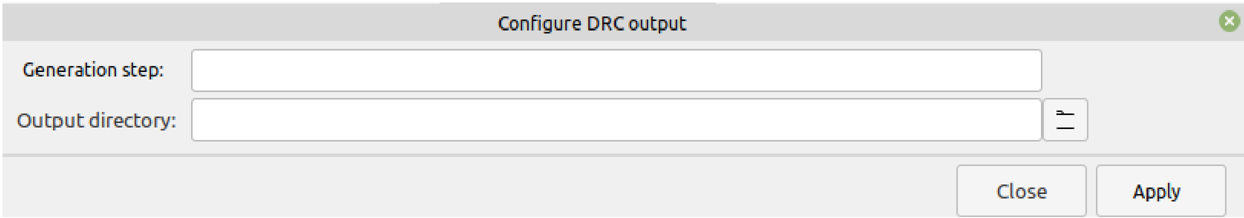
-An “Output directory” field.

-A “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-An “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”.

-A “Close button” that closes the window and does not save any change.

Configure DRC window



This is a new window with :

-An “Output directory” field.

-A “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-An “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”.

-A “Close” button that closes the window and does not save any change.

All the DRC settings are retrieved from Pcbnew.

Configure Python scripts window

-A “Python script” field for the user to select the python file to be executed
-A “Command line” field to feed the script arguments.

-A “Generation step” field which will appear in “Selected configuration steps” in the project setup window.

-An “Apply” button. Checks the unicity, for a given generation configuration, of “Generation step”.

-A “Close button” that closes the window and does not save any chang

- Checkbox “Wait for script to finish”. If not checked, KiCad executes another output command, with current python script left running in the background. Else, execute next batch command only after current python script is finished.

- A “Timeout” field to specify a timeout, in minutes.

Output path changes

Text replacement

When evaluating the output path, for each output step, the user should be able to enter environment variables, and project-wide text replacement. (Preferences -> Configure Paths)

For some parameters, that we should get from the project setup:

$(KICAD_PRJ_REV)	Project revision from project window. This is something that does not exist yet. Could be added in file > Project setup
$(KICAD_SCH_REV)	Schematics revision from Pcbnew
$(KICAD_BRD_REV)	Schematics revision from Eeschema
$(KICAD_PRJ_NAME)	Project name
$(KICAD_TIMESTAMP)	Creation date : - YYYY-MM-DD (preferred for sorting) (other date formats ? how should we handle them ? other parameters ?)

Example :

Output Path :

```
./$(KICAD_PRJ_NAME)_rev_$(KICAD_PRJ_REV)_$(KICAD_TIMESTAMP)/layout/gerbers
```

Could turn into :

```
./BluetoothBoard_rev_2.3_22-10-2020/layout/gerbers
```

Prefix, suffix and extension

When generating files (gerbers) in kicad, the filenames, and file extensions are non customizable. In the project setup (not a thing yet) the user should be able to enter, for each kind of output, a prefix, a suffix and a file extension. Those will be project settings, and therefore, two output jobs for the same output types, cannot have different names.

The name of the generated files will be:

PREFIX + PROJECT_NAME + SUFFIX + “.” + EXTENSION

Example:

- The project name is “MyProject”, and is in revision 03
- The user sets the “GTL” and “GBL” extensions for gerbers, top and bottom, to match with Alitum extensions.
- The user sets suffixes to “-rev-$(KICAD_PRJ_REV)” for all gerbers files
- The user sets prefixes to “gerberTop-” and “gerberBot-”.

The filenames for those two files will be:

- gerberTop-MyProject-rev-03.GTL
- gerberBot-MyProject-rev-03.GBL

Implementation notes

I am trying to explain how I would implement it, but there will be better ideas ! If you think of a better one, or a problem about what is described, please leave a comment !

Code in common/

In common, there would be everything needed to manage automated output generations.

Enum: OUTPUT_JOB_TYPE

This enum will help us to differentiate the output job type.
Found in <common/output_job.h>

```
OUTPUT_JOB_UNDEFINED = -1,
OUTPUT_JOB_BOM,                    // Bill of Material
OUTPUT_JOB_ERC,                    // Electrical Rule Check
OUTPUT_JOB_SCH,                    // Schematic
OUTPUT_JOB_BRD,                    // Board Plot
OUTPUT_JOB_DRC,                    // Design Rule Check
OUTPUT_JOB_DRILL,                  // Drill file
OUTPUT_JOB_POS,                    // Footprint position file
OUTPUT_JOB_FOOT_RPT,               // Footprint report
OUTPUT_JOB_NETLIST,                // Netlist file
OUTPUT_JOB_PYTHON,                 // Python script
OUTPUT_JOB_EXTERNAL,               // Any command
NB_OUTPUT_JOB_TYPE
```

Note : we might have to differentiate the formats, and define :

```
OUTPUT_JOB_BRD_DXF
OUTPUT_JOB_BRD_GERBER
OUTPUT_JOB_BRD_PDF
…. And others
```

Class : OUTPUT_SETTING

This class stores all settings needed to generate an output.

OUTPUT_SETTING would inherit from NESTED_SETTINGS .

From this class inherit the following classes :
-OUTPUT_SCH_SETTINGS
-OUTPUT_ERC_SETTINGS
-OUTPUT_BRD_SETTINGS
-OUTPUT_DRC_SETTINGS
-OUTPUT_DRILL_SETTINGS
-....

Each of these classes can be fed to the appropriate output generator

Ie : OutputGenerateSCH(OUTPUT_SCH_SETTINGS* aSettings, KIWAY* aKiway);
(For the KIWAY, see section: Function: OuutputGenerateXXX())

Attributes : (other than the inherited ones)

- m_name : wxString : name
- m_outDir : wxString : output directory
- m_outFile:wxString : output filename (for output types that generate only one file, eg: ERC)
- m_type OUTPUT_JOB_TYPE : job type

Methods : (other than the inherited ones)

- constructor(wxString name)
- Getters / Setters. /!\ Cannot set OUTPUT_JOB_TYPE after construction:
    - setName(wxString aName)
    - setName()
    - getType()
    - setOutDir(wxString)
    - getOutDir()
    - setOutFile(wxString)
    - getOutFile()

Class OUTPUT_SCH_SETTING

This class inherits from OUTPUT_JOB_STEP, and adds methods that are specific to plotting schematic.
More classes are to be defined, one for each OUTPUT_JOB_TYPE

Class: OUTPUT_JOB

This class is a collection of OUTPUT_SETTINGS. Basically, this groups all the steps in an output job. Inherits from JSON_SETTINGS.

Nested architecture :

```
{
  "OUTPUT_JOB_A" = {
      "OUTPUT_SCH_1" = {
          ....
       },
      "OUTPUT_BRD_1" = {
          ....
       }
  },
  "OUTPUT_JOB_B" = {
      "OUTPUT_SCH_1" = {
          ....
       },
      "OUTPUT_BRD_1" = {
          ....
       }
  },
}
```

Methods : (other than the inherited ones)

- run(KIWAY* aKiway) : generate all the required outputs.
- add(OUTPUT_JOB_TYPE aType) : add a new output generation, with a given type
- remove (int aIndex) : move the selected output job step up in the vector
- get(int aIndex): get a pointer for the output job at position aIndex.
- moveUp (int aIndex) : move the selected output job step up in the vector
- moveDown (int aIndex) : move the selected output job step down in the vector.

Functions : OutputGenerateXXX()

For each OUTPUT_JOB_XXX , there is a OutputGenerateXXX function.

OutputGenerateXXX(OUTPUT_XXX_SETTINGS, aSettings ,KIWAY* aKiway).

eg:
OutputGenerateSCH(OUTPUT_SCH_SETTINGS, aSettings ,KIWAY* aKiway).
OutputGenerateERC(OUTPUT_ERC_SETTINGS, aSettings ,KIWAY* aKiway).
OutputGenerateBRD(OUTPUT_BRD_SETTINGS, aSettings ,KIWAY* aKiway).
OutputGenerateDRC(OUTPUT_DRC_SETTINGS, aSettings ,KIWAY* aKiway).
OutputGenerateDRILL(OUTPUT_DRILL_SETTINGS, aSettings ,KIWAY* aKiway).
…

The actual generating function stays in its player (Eeschema / Pcbnew). These functions, that are in common/ , use the KIWAY to get the EDA_FRAME, and then call the actual generating function.

We certainly want an extra argument: a REPORTER. But I have not looked that class yet. (a WX_STRING_REPORTER ?). This reporter would need to propagate to the KICAD_FRAME_MANAGER, and could be saved to a file.

Code in Eeschema/

The class that does the actual output generation is the dialog :
https://docs.kicad-pcb.org/doxygen/classDIALOG__PLOT__SCHEMATIC.html#a856b4fac571d3e3e80341110c8f70ca4
But when we look into the code, almost all lines refer to its parent : the SCH_EDIT_FRAME:
https://docs.kicad-pcb.org/doxygen/classDIALOG__PLOT__SCHEMATIC.html#a7380732eadabb3f23e73deeba5747386

I suggest we move the plotting code from DIALOG_PLOT_SCHEMATIC to the SCH_EDIT_FRAME class, so we don't have to create the plotting dialog in order to plot. Information from the SCH_EDIT_FRAME could be retrieved using the KIWAY.

Each EDA_FRAME that can generate outputs (Eeschema, Pcbnew)

Code in Pcbnew/

Same as code in Eeschema/ but

Code in kicad/

In kicad/ there would be the user interface.

Class PROJECT_SETUP

This is not something yet, we should have something like DIALOG_BOARD_SETUP

Class PANEL_OUTPUT_GENERATION

Panel for the PROJECT_SETUP.