AgALag users guide

Gregor
Axoris

06/03/2003

1 What Is AgALag

AgALag is an *Axoris Group AL3101 Application generator*.

AL3101 is a *Digital Sound Processor* (DSP) used to perform audio effects. Once you have an effect (distorsion, phaser, ...) you can load it in the chipset. However, in real world, you often need more than one single effect (*e.g. compression, distorsion, equalizer, noise gate*). Once you have all your separated effects, AgALag will allow you to enqueue and connect them to build everything in a complete effect. That effect can be loaded in the hardware stuff.

Practically, the structure of the full effect is described by a project file (with *axp* extension). This script contains the separated effects you need, the links between them and their parameters. However, writing your own scripts is a tedious task[1]. AgALag is a graphical interface which allows you to create your effects by clicking on the modules you want and linking them by simple mouse clicks.

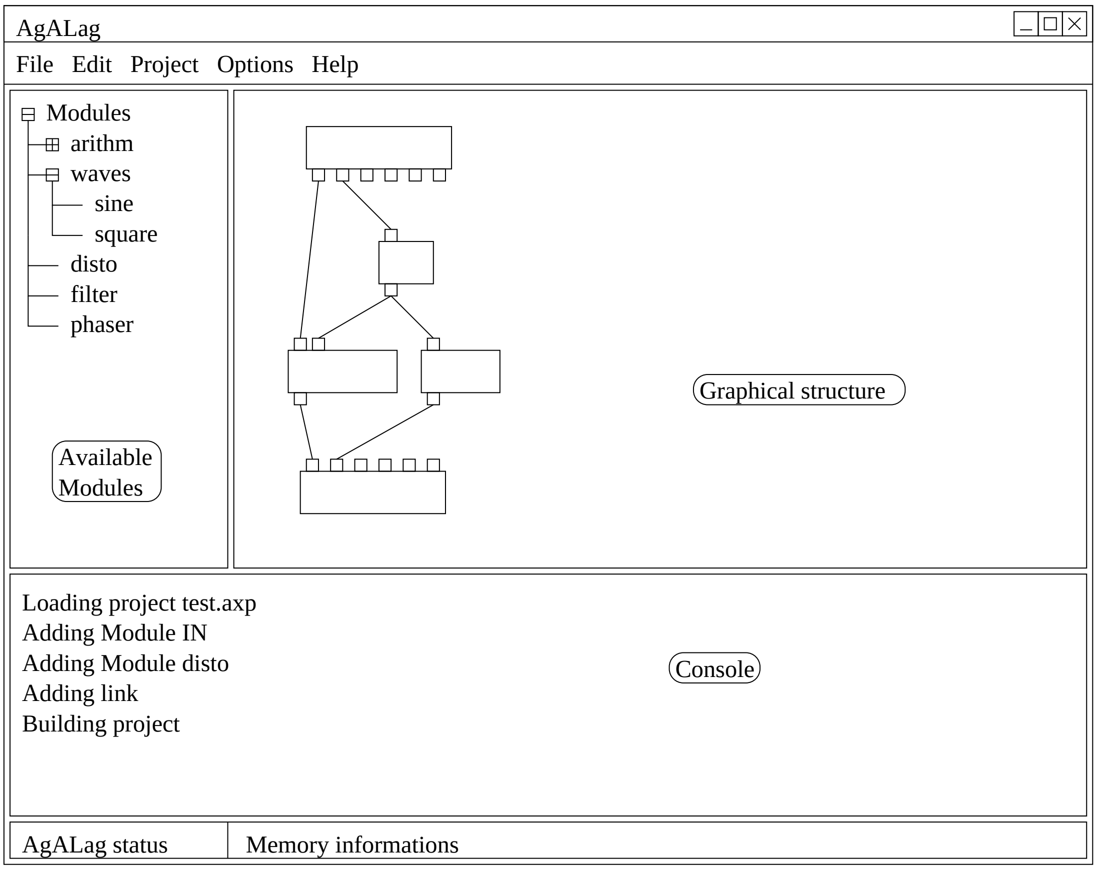
IMPORTANT NOTE

You may not include generic codes for AL3101 in the structure. The modules you will use must be compoant with the AgALag coding rules[2]. If you try to include an invalid module, the program will give you the message *Module doesn't fit the template, using default values.* In such a case, it is very likeky that project generation will fail.

[1]The fearless reader who wants to write its own scripts from scratch can have a look at the *script.pdf* file

2 Window frames

The graphical interface of AgALag is presented in the figure below



```
Loading project test.axp
Adding Module IN
Adding Module disto
Adding link
Building project
```

Here is a brief description of the main parts:

- **Menus** allow you to perform actions on the project (open, save, options, build, ...) and to configure AgALag.
- **Module list** is the list (shown as a tree control) of available effect modules. These will be the blocs for the builded structure. Effects are taken from the *effect directory*, configurable from the menu.
- **Graphical structure** is the place where you will build the effect chain. It will display the modules and the link between them.
- The **Console** will show messages summarizing the actions performed (adding modules, links, build messages, error messages, ...)
- The **Status bar** contains two parts:

  - A **General status message** giving you short informations about e.g. menu actions.

  - **Memory informations** tell you the state of the program and data memory of the DSP (both limited to 1K)

3 Basic concepts

Let's start with a short step-by-step presentation of how AgALag works. Details of each step can be found in next sections of this document.

Assuming that we start a project from the beginning, the first thing to do is to *create the project*, which means taking some effect modules and adding links between them (see Section 4. You can also specify the parameters of each effects (e.g. frequency of a filter). Since AgALag is designed in such a way that the effects you build can be use for further structures, you also need to specify the basic informations (name, description, ...) for your project. All these informations (Project informations, effects used, links between effects and effects parameters) are stored in the project file (extension .axp).

Once you have built the project file, the second step is to convert all its informations into a full independent effect. This will *create the program code* for your new effect setup (see section 5.

In addition to this basic features, we have added a bunch of useful tools:

- An *Assemble* command, which converts the program (.asm) code into an object code (.obj, directly loadable into the device),
- A *software simulation*, allowing software tests of your effects,
- An *integrated loader*, used to upload your effect into an external device.

4 Creating a structure

The aim of the game is to put modules in the "structure frame" and linking them together

4.1 INPUT and OUTPUT modules

The sound is supposed to flow from the INPUT module to the OUTPUT module. We have 8 inputs and 8 outputs (the first one beeing at the left). The effects chain must therefore stand between those modules and be connected to them.

4.2 Dealing with modules

Each module takes some input(s) and furnish some output(s). For each module, inputs are at the top and output at the bottom. The operations you ca, make are the following

- **Adding a module:** you can add a module from the *Module List* by simple drag & drop. You drag it from the module list and you drop it in the structure where you want to place it.
- **Removing a module:** right clicking on a module in the structure gives you a contextual menu where you can delete the module.

- **Moving a module:** moving a module in the structure is done by drag'n'dropping it inside the structure.

4.3 Editing effects properties

Each module has parameters (such as gain, treashold or level). These parameters have default values in each modules but these values can of course be set by the user. To do so, right click on the module you want to parametrize and select properties. A dialog box will appear containing the list of each parameters (a name and a value). You can then set the value you want and close the dialog box.

Remarks:

1. if the same module is used twice in the structure, their properties are specified independently.
2. the value of the parameters must be larger than -8 a,d smaller than 8.
3. some of the parameters are specified indirectly (*e.g.* for a triangle wave generator, we give the step increment instead of the frequency). In these cases, the way parameters are obtained from "real" properties should be found in the module file[3].

4.4 Dealing with links

Links are connections between effect modules. They go from tho output of a module to the input of another one. To edit links, you may use the following tricks:

- **Adding a link:** links between modules are added by clicking on the output channel of the starting module and on the input channel from final module (in any order).
- **Removing a link:** you can remove a link by right-clicking on one of its ends.

4.5 Editing project informations

5 Building the effect

Once you have created all your effect chain, you still need to convert your project into some stand-alone effect that you will be able to use. You will convert your .axp project into a .asm program code (this is the same extension as for all effect modules). This conversion is simply performed by the command *Make* accessible from the menu. Comments about the process will appear in the console (you can choose the level of comments in the project options).

[3]presumably in the header, in section 1 or in sction 2

6 Additionnal tools

6.1 Assembling

In order to be understood by the external device, the program code (assembler) must be converted into an object code (binary). This step is done by the *Assemble* command. It will convert the .asm file into a .obj file.

Remark: Note that the effect is mainly constituted of two informations: the program of the effect and the values of the parameter. The latter must be stored into the data memory of your device. This can be done in two ways

1. All information is placed in the effect's code which deals the initial values of the parameters.

2. We store the parameter information into a *Memory file* (with extension .mem), and we let the computer dealing with these two files.

When assembling, the second case generates one additional file, but produces a smaller code. Since the effect's code is limited, this allows for more complicated effects. Therefore, if you encounter memory overflow problems, you should activate the memory file. You can do that from the project options.

6.2 Software simulation

If you want to test your effects, we have included a software simulation. Choosing the *Simulate* option in the menu, you will have a dialog box. You can specify a input an an output file. When you will launch simulation, your effect will be applied on the input file and the result will be saved into the output file.

6.3 Loader

... to be done ...

7 AgALag options

There are basically two types of properties the user can set:

1. Various directories such as the directory where your effects are stored

2. The project options (Project-¿Options menu):

   - The verbose level for the *Make* command, which controls the number of messages displayed in the console when you execute *Make*,

   - The verbose level for the *Assemble* command, which controls the number of messages displayed in the console when you execute *Assemble*,

- The *Generate memory map* option. If checked, *Assemble* will split the object code in two parts: a program code and a data code containing values of the parameters. This option is useful to produce lighter program code and to avoid program code overflow.

8 Bug reports or comments

If you have any bug report or comment, please send it to

Gregor.Soyre@swing.be

Contents