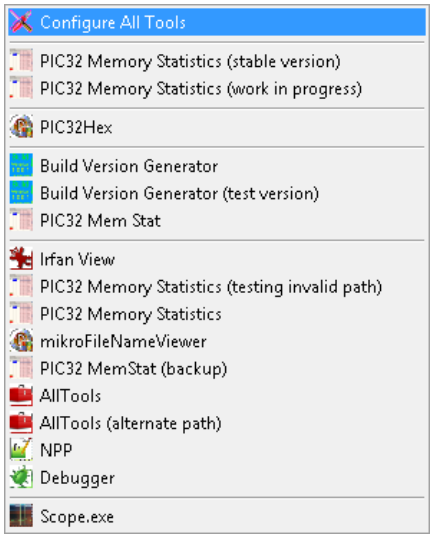
AllTools for mikro IDEs

Author: VCC
Version: 1.0 (December 2016)

All Tools is an application, which allows more than 10 user tools to be started from a mikro IDE. It is a menu-based design, as opposed to a list of buttons, thus taking only one button space on the IDE. When started, it shows a menu, where each item executes a specific tool. In addition, for each installed tool, two more applications can be executed, one before the tool, and the other, after the tool.



Application Features:

- It is started from a mikro compiler as a tool.
- When displaying the menu, a highlighter rectangle points to the position of the mouse cursor.
- It allows running an application before a configured tool and another application after the tool.
- A tool can be started with a timeout with regard to the “before tool” application or with infinite waiting.
- Menu separators can be added between menu items.
- Uses icon cache to allow fast loading of application icons when displaying the menu.
- Displayed icons can be replaced with user icons (it can load .exe and .bmp files).
- It passes macros from mikro IDE to the configured application.
- In addition to standard command line macros defined in a mikro IDE, users can create their own macros.
- All three tools (main tool, “before” tool, “after” tool) can use macros among their arguments.
- The list of tools and “before”/”after” applications, is presented as a table and can be easily edited.
- Items from the list of tools can be displayed as 32px or 16px high, to preview application icons.
- A debug window is provided, so that application arguments can be matched against macros.
- Installation in a mikro IDE is done by using a button on the main editor. **All Tools** is able to edit tools.ini.
- Before editing tools.ini from a mikro IDE, **All Tools** makes a backup of that file, using a timestamp.
- The tools installed in a mikro IDE can be imported to **All Tools**, using a button on the main editor.

Requirements / Recommendations:

- The application must be run with all the macros provided by a mikro IDE. Because of a bug in IDEs, an additional parameter must exist after the last macro.
- Macros have to be passed to **All Tools** in the same order, as **All Tools** application expects all of them to exist.
- The **%WORD_AT_CURSOR** macro has to have a blank before it, and it must be enclosed by double quotes to make sure a content is passed as an argument to **All Tools** when that word is empty. The macro should look like this: "**%WORD_AT_CURSOR"**

Limitations:

- No keyboard shortcuts are available.

Table of contents:

1. Main Editor

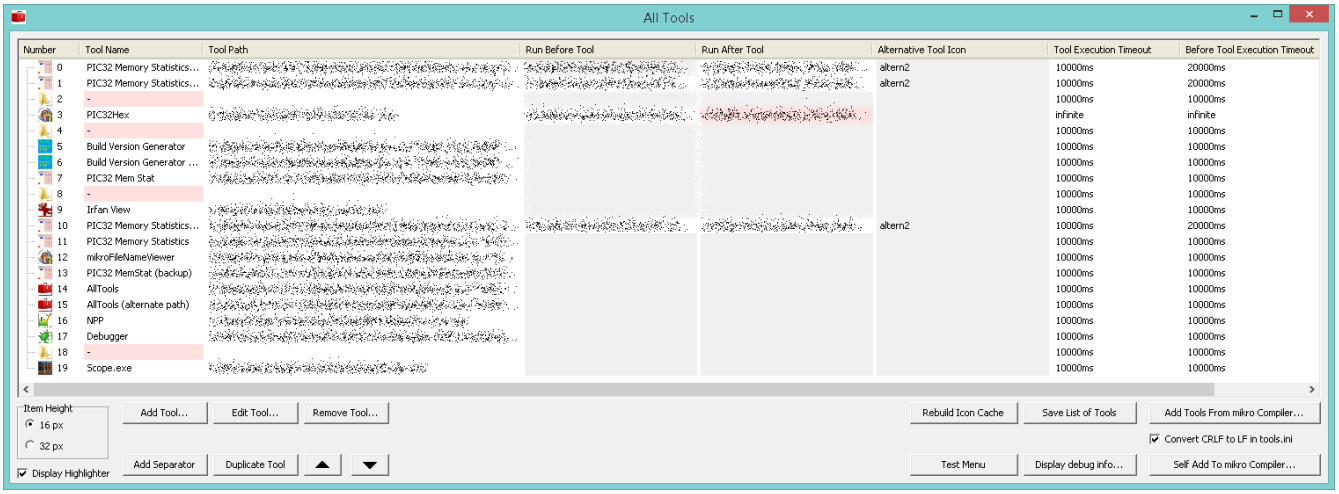


Fig.1 All Tools – main editor

The main editor (Fig.1) is used to visualize the entire list of tools and their settings. From here, tools can be added, removed, duplicated or arranged. From this same window, **All Tools** application can be self installed in a mikro IDE and it can read the IDE's tools.ini file, to import the tools from it.

1.1 Installation

Installation should be easy, because **All Tools** has the ability to edit the tools.ini file from a mikro IDE. It can be installed in the following way:

- Run the application from e.g. Windows Explorer. The main menu is displayed, with only one item if no tool has been added so far. This menu item is “Configure All Tools”.
- Click on “Configure All Tools”, and the main editor opens.
- In the bottom-right corner of the window, there is the “Self Add To mikro Compiler…” button. Click it.
- A file browser opens, asking you to point to a file from the compiler/IDE installation folder. Tools.ini file should be there. For example, you can look for the IDE executable (e.g. “mikroPascal Pro PIC.exe”). It is safe to point to any file from that folder, because **All Tools** looks for tools.ini. If the file does not exist in that folder, **All Tools** creates one. If the file exists, it is backed up (using a timestamp for the new file name) and verified if it can be safely modified. If there is at least one available space in the list of tools for that particular IDE (with a maximum of 10 tools), **All Tools** adds itself to the first available space. It configures its path, its name and all the existing macros as required. A messagebox informs you about the backup of the tools.ini file, and a second messagebox informs about the result of the installation (success or not). The verification is based on reading the new data from the ini file and compared against what should now be there, for the new tool (**All Tools** application). If it reports that no available space could be found in the list of tools, don't worry, because all installed tools can be imported to **All Tools,** in order to make room for more (see next step). Be aware that this button does not fix a broken setting, it just adds the tool again to a new available tool space.
- The second part of the installation is to import all the tools from an IDE to **All Tools** application. For this, click the “Add Tools From mikro Compiler...” button, and point to a file from the IDE's installation folder, near tools.ini or this file itself. The tools.ini file must exist, meaning that at least one tool has to be configured in that particular IDE, in order for **All Tools** application to import something. If the file exists and no tool is configured, nothing is being imported.
- If you have multiple compilers/IDEs installed, you can repeat the process for each of them.

While developing this tool, an IDE bug came out. It prevents any tool from being displayed as a button on the IDE, if configured as the last tool. **All Tools** reports its position after installation, but it is not its job to warn you about this bug. If **All Tools** reports number 10, most likely you won't see it installed, but it's there.

1.2 Adding/Removing tools

The most used features of the main editor are the adding, removing, arranging and previewing the installed tools. For this purpose, there are 7 buttons on the bottom-left side of the main editor ("Add Tool...", "Edit Tool...", "Remove Tool...", "Add Separator", "Duplicate Tool" and the two up/down buttons), and the "Test Menu" button from the bottom-right side. After adding, removing or arranging tools in the list, the "Test Menu" button provides an easy way of previewing how the menu will look like when starting the application from an IDE.

All editing settings are automatically saved, both when closing the main editor, and when clicking the "Save List of Tools" button.

1.3 Other settings

There are few more features that need to be presented, regarding the main editor. To speed up loading application icons when building the menu, **All Tools** uses a file cache with all the icons from the installed tools. This cache is automatically updated when adding/editing/removing/arranging the tools, but some times, it has to be fully regenerated (e.g. when manually removing all the temp icons). For this purpose, you can use the "Rebuild Icon Cache" button.

All Tools works with icons of only two sizes 32px x 32px, and 16px x 16px. The 32x32 version can be displayed in the list of tools from the main editor, using the "Item Height" radio group.

Another feature that may be both useful and annoying, is the highlighting rectangle that tells where the menu pops up. It always pops up where the mouse cursor is when starting the application, but a one-item menu might not always be spotted. This is the case when no tool exists yet in **All Tools**, so the menu will contain only the "Configure All Tools" menu item. By adding more and more tools, the menu increases in size, so it becomes even more visible and easy to spot. If the highlighting rectangle becomes useless or annoying at this point, it can be switched off from the "Display Highlighter" checkbox.

The last feature that needs to be addressed here, is the "Convert CRLF to LF in tools.ini" checkbox. This allows saving the tools.ini file using the CRLF or the LF return characters. The mikro IDEs prefer the LF version for the tools.ini file, while having CRLF for other ini files. Although IDEs should be able to handle the CRLF version, **All Tools** allows creating/editing the files as the IDEs like. This feature can be switched off by the "Convert CRLF to LF in tools.ini" checkbox.

2. Tool Editor

The tool editor (Fig.2) opens by clicking the "Edit Tool..." button on the main editor. It allows editing the name, path and command line arguments for each tool. Also here, two more applications can be configured to be executed, one before the tool (called here the "before tool") and the other, after tool (called "after tool").

By default, the displayed icon for a tool, is extracted from the tool's executable itself. If multiple icons are available inside an executable, a default one is picked. **All Tools** allows displaying a different icon for a tool, if the "Use alternative tool icon" checkbox is checked and a valid icon is set in the "Alternative tool icon" editbox. See Fig.3 for an example of a wrong path to an icon. The supported formats for these icons are .bmp and .exe. They are automatically resized to 32x32 and 16x16, as needed.

Most of the tools are used from a mikro IDE, because the IDE provides macros that represent content particular to an open project (files, chip name, clock frequency, cursor position in editor etc). These macros are available to installed tools, as command line arguments. In addition, users can add their own macros.

Executing applications, one after another, implies waiting for execution. However, users may want to set a timeout, in case something goes wrong or simply because the main tool and one or both of the other two applications ("before tool" and "after tool") have to be run in parallel. Set this timeout to -1 if the waiting should be infinite.

From this editor, the macro editor can be displayed, using the "Edit list of macros..." button. Also the debug window can be displayed from here, using the "Display debug info..." button.

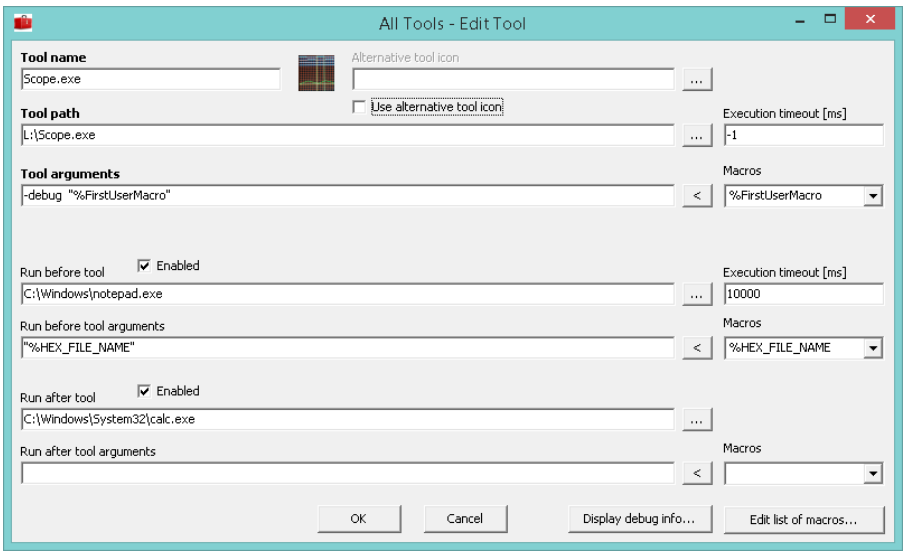


Fig.2 All Tools – tool editor

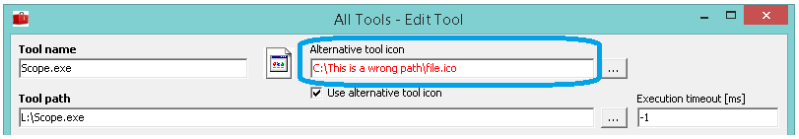


Fig.3 All Tools – tool editor – highlighting invalid paths

3. Debug Window

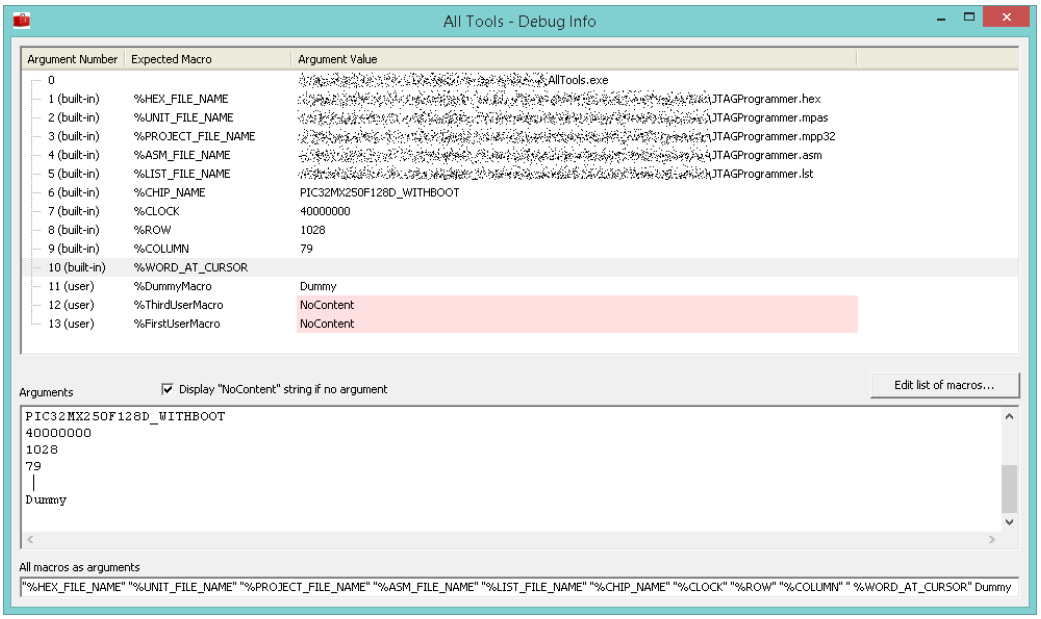


Fig.4 All Tools – debug window

The purpose of this window (Fig.4) is to display what the application arguments are and how are they matched against the list of available macros (built-in macros + user macros). The built-in macros are expected to be matched in their particular order. **All Tools** identifies their content by index. When passed to installed tools, they are identified by name (as string). The purpose of user macros is for future compatibility with mikro IDEs, or even using **All Tools** with other IDEs.

4. Application as a mikro IDE tool

This application is usually run as an IDE tool, see Fig.5. It requires as a command line parameter all the existing macros from the IDE, plus a dummy string after the last macro. The command line parameter, that has to be set can either be automatically set by **All Tools** (see chapter 1.1, Installation), or can be manually copied from the editbox on the debug window (see Fig.4).

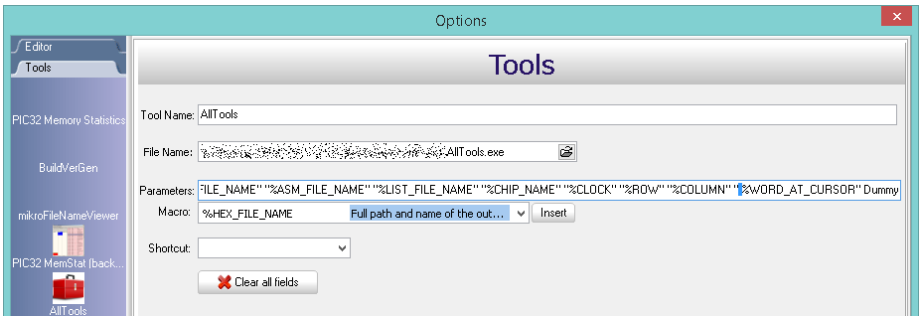


Fig.5 mikro IDE tool settings for **All Tools** application