Python Boot Manager

PC Interface for loading Python scripts
30/08/07

Making machines talk.

Contents

1.1 Scope

This document describes the tool for loading Python scripts on a large quantity of modules, copying the memory of a main module, where these scripts have been tested, to all the other modules.

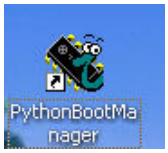
1.2 Introduction

Loading and compiling on several modules a Python script can be a long operation, depending obviously on time necessary to compile the script. If this is too long then the alternative is to load and compile on a main module the script and then to copy the memory to all the other modules. The main advantage of this method is a constant time to flash each module, independently from the complexity of the script.

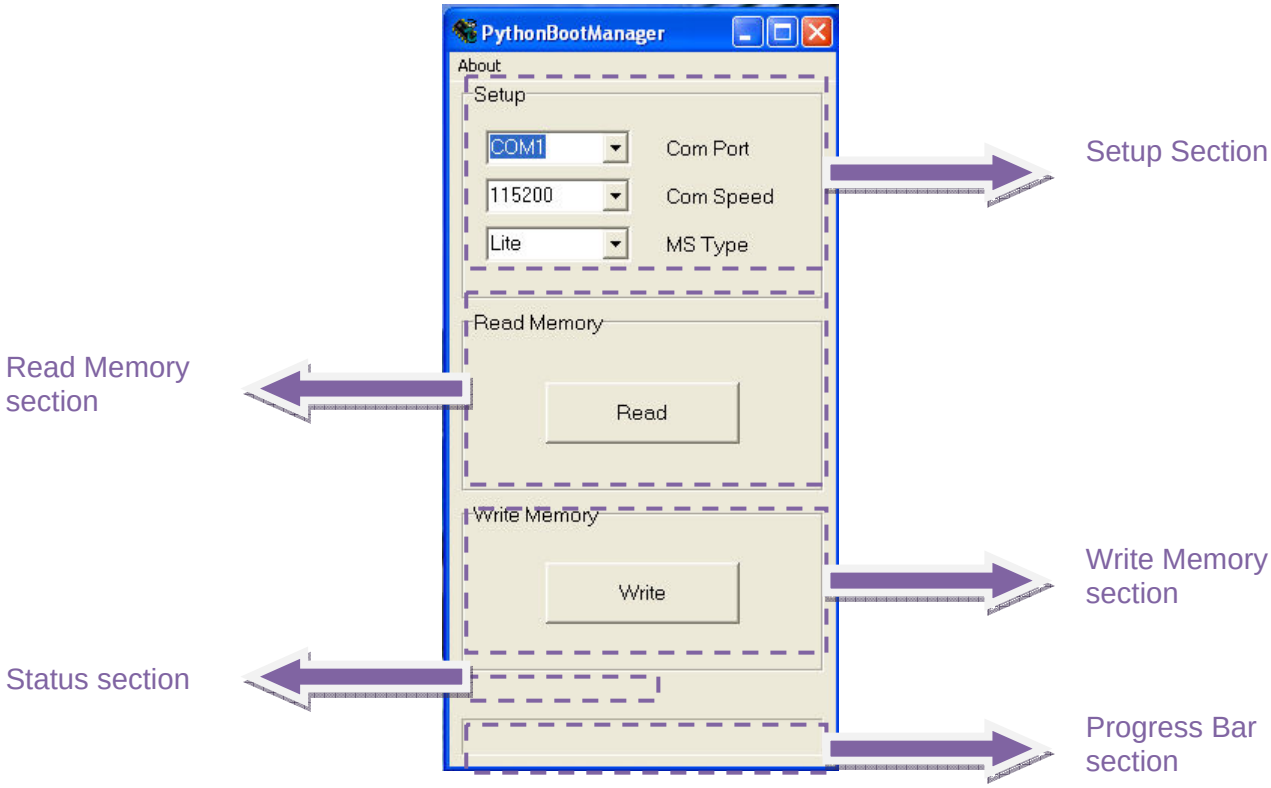
NOTE: All modules must have the same firmware version as the main module.

1.3 Using Python Boot Manager Application

The tool is identified by the following Icon on your desktop:



When running, the main window is the following:

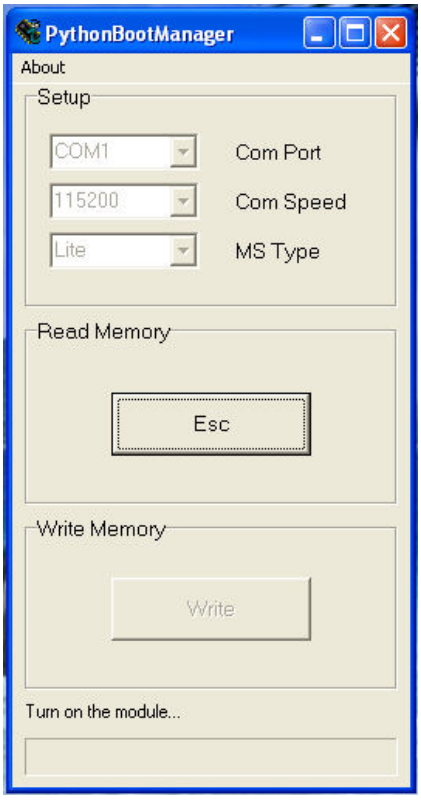


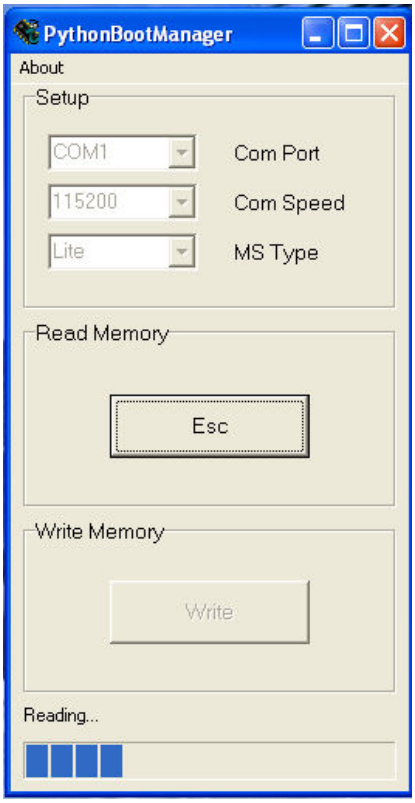
There are five different sections:

- Setup: with three different combo boxes to choose the COM port to use, its speed and the kind of module technology, V3 or Lite.
- Read Memory: pushing this button you can read the module memory and save it in a binary file. A dialog asking for name and path of this file appears when pushing the read button.
- Write Memory: pushing this button you can load in the module memory a binary file previously created with the read section. You can choose it with a dialog appearing when the write button is pushed.
- Operation Status: Box explaining the operation the tool is executing.
- Progress Bar: a bar showing the progress of the task.

1.3.1 Reading the memory

When we are sure that the python script running in the main module is OK, we can turn off the module and read the memory. This is very simple: we must only push the button Read, choose the name of the memory file, turn on the module as suggested from the status box and then wait until the OK message appears in the Status section.

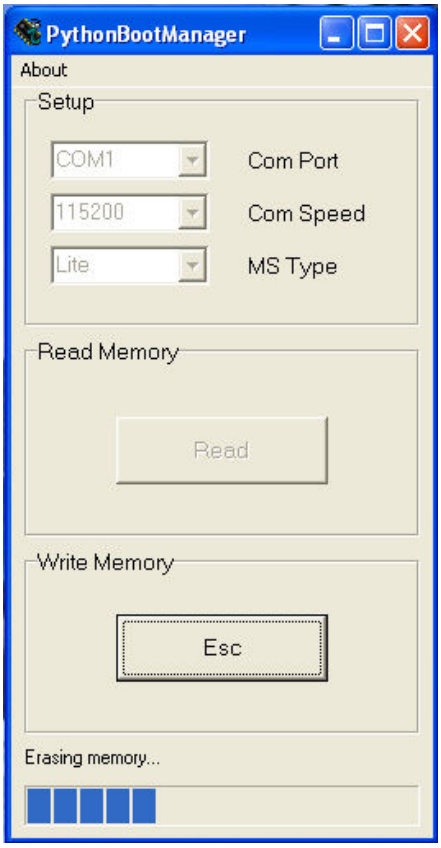


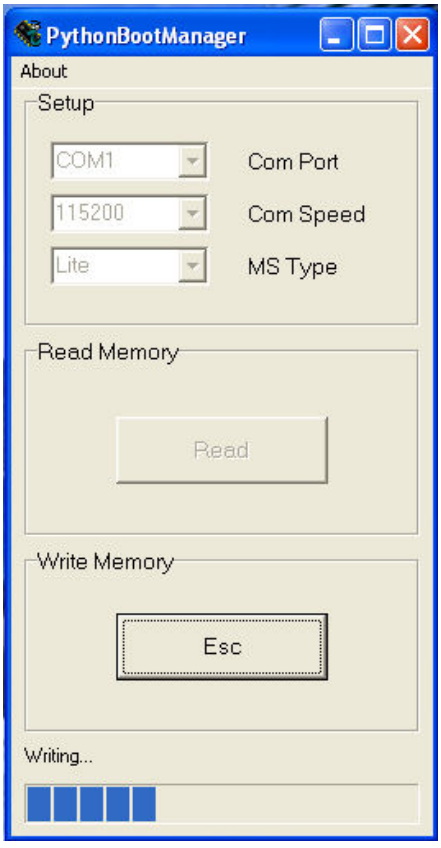


If it's necessary we can interrupt the reading pushing the Esc button.

1.3.2 Writing the memory

If we have a module memory stored in a binary file, then we can proceed writing this memory on another module so all the python scripts loaded in it will be copied in the new module.
After having pushed the Write button and choose the file to load, the memory will be erased and the file loaded in the module.





As you can see, you can abort the operation pushing the Esc button. This operation is not recommended and should be used only if strictly necessary.

1.3.3 Flashing procedures

From an operational point of view, user will be asked for:

1. switching on the new module, without SIM, with DTR ON;
2. loading required Python scripts (.py) through #WSCRIPT, and enabling main script through #ESCRIPT.
3. switching off and switching on the module, without SIM, with DTR OFF; the loaded Python scripts will be compiled, and the corresponding objects (.pyo) will be built;
4. switching off and switching on the module, without SIM, with DTR ON and deleting the Python scripts (.py) through #DSCRIPT if .py scripts should not be visible;
5. switching off the module, running PythonBootManager and linking the module;
6. reading module memory through PythonBootManager and storing it in a PC file;
7. updating the memory of every other module through PythonBootManager, overwriting it with the contents of the previously defined PC file.

NOTE: All modules that will be flashed must have the same firmware version as the main module.

When this procedure is completed on the same directory were Python Boot manger is palced a log file PBMLog will be created. Two different messages can appear in this log file, depending of the flash procedure result:

- Ok - in case the procedure ended correctly
- Error: type of error – in case there's been an error

1.4 Python Boot Manager from Command Prompt

The Python Boot Manager software can also be launched from the Command Prompt. The following command line should be launched:

```
PythonBootManager COM22 460800 Lite C:\PythonBootManager\Memory\memory.bin
```

First of all you should specify the executable program: in case you're already on the directory where the program is placed (as in this case) only the name of the program is required (PythonBootManager) otherwise you should specify the whole path.
The parameters listed above have the following meaning:

- first parameter is COM port
- second parameter is the speed
- third parameter specifies if the module is Lite or V3
- and the last one is the path of the file to upload

1.5 Version for the Production

We have also created the Python Boot Manager version for the production that can only perform the write operation: PythonBootManager1.3_w.