HD-64 Developer’s Guide

Author: Vittorio Pascucci (Side Projects Lab)

If you wish to get in touch with the developer, please join the SPL Discord server.

LICENSE: This work is provided under the **Creative Commons CC BY-NC-ND 4.0** License:
https://creativecommons.org/licenses/by-nc-nd/4.0/

1. Introduction

The HD-64 is a commercial product and, as such, not all of its source code is available to the public:

- All hardware design files are publicly available for non-commercial use so that all users have all information necessary to freely repair/modify their board
- All the sources for the VIC-II emulator subsystem are free software available under the GPLv3 license (free as in free speech) so that anyone can contribute to improving the system and get closer to a 100% accurate reproduction of the original VIC-II

Without the complete codebase being available it is not possible for contrinutors to set up a local development environment. This would be an issue because contributors need to test the effects of their adjustments on the real hardware and, thus, need to be able to build HD-64 update packages autonomously.

For this reason the SPL build server has been made available to the public so that contributors can use it freely to build new binaries and have them delivered via GitHub, without having to set up a local development environment. This document explains in detail how to contribute code to the VIC-II subsystem of the HD-64 and how to use the SPL build server to that end.

2. How to Build New Binaries (TLDR)

Building a new set of binaries, together with the python utilities to update the HD-64 firmware can be done in a few simple steps:

1. Get in touch with the maintainer (sideprojectslab) through Discord to align on what the intended contribution will be and to become an "authorized user" of the SPL build server.

NOTE	This step is crucial as only "authorized users" are able to use the SPL server

1. Fork the project and make some changes to the VIC-II VHDL code base available under `/development/firmware/vicii`
2. Create a Pull Request (PR) to the main branch of the original repository, add a descriptive title and make sure the keyword **@build** is in the description
3. After some time (about a minute if the SPL server is idle, more if it is busy building) a new "draft" release will appear on the original HD-64 repository with title: **"AUTOBUILD: <date-and-time-of-the-last-commit>"** containing the original title of the PR in the description
4. After about 10 minutes an "asset" will be added to the release (with the same name), containing:
    - the build log from Vivado (the xilinx FPGA toolchain) under `/dump/pass` or `/dump/fail`

depending on whether the build has succeded or not

- the new binary, python update utility and python debug interface under `/output/misc/HD-64 vX.XX/src`

5. Ensure you have python 3.10 or higher installed and that the following python3 packages are installed:
    - pyserial
    - ezpath
    - ipython
6. To update the HD-64 with the new firmware run: `python3 firmware_update.py` within the `/output/misc/HD-64 vX.XX/src` directory while following the same steps described in the README (Firmware Update section)
7. To interact with the HD-64 system registers and enable debug features run: `python3 console.py` within the `/output/misc/HD-64 vX.XX/src`. An interactive console will open, allowing direct access to the HD-64 memory map. Please refer to Section 4 for a detailed explanation of the console's operation.

3. The SPL Build Server

The SPL build server is not directly accessible to the public. Instead, it interacts with GitHub through its API to grab new source code from pending Pull-Requests (PR), build new binaries, and then deliver them back to the developer through GitHub, in the form of "draft releases":

- The SPL server polls the HD-64 repository on GitHub once every minute and checks if there are pending PRs
- If any PR is pending the SPL server checks whether the author of the PR is present in a private list of authorized users and if the keyword "@build" is present in the PR description.
- If the conditions above are met, a new "draft release" is created on GitHub titled: **"AUTOBUILD: <date-and-time-of-the-last-commit>"**. The release description contains the original PR title, a link to the last commit and the PR unique index assigned by GitHub
- The server clones the source repository of the PR, grabs the VIC-II VHDL sources and replaces them in the latest stable codebase of the HD-64
- The server runs the build, collects compilation logs and creates an update package with the new binaries
- The compilation artifacts above are compressed and added as "asset" to the release on GitHub
- The release on GitHub is deleted after 24 hours after being created.

4. Python Console

NOTE	It is not possible to brick the device by abusing the "console", however it **is** possible to intentionally erase the current HD-64 firmware and/or configuration data. Please get in touch with the maintainer for basic coaching before using this feature.

run `python3 console.py` within the `/output/misc/HD-64 vX.XX/src`. An interactive console will open, allowing direct access the HD-64 memory map.

- refer to the file `/output/misc/HD-64 vX.XX/src/memmap_hd64.py` for a complete description of the HD-64 memory map. The HD-64 memory map is structured hierarchically and consists of any combination of the following items:
- Registers (REGISTER)
- Register Fields (FIELD)
- Memory Ranges (MEMRANGE)
- Other (nested) Memory Maps (MEMMAP)
- as an example type `hd64.` and press `Tab`. If IPython is installed correctly this will show all registers/register-fields/memory-ranges visible at the current level of the memory map hierarchy.
- as an example run `hex(hd64.version.val)` to read the aggregate version register of the HD-64 in hexadecimal format.
- as an example run `hd64.version.fwv_minor` to read the minor firmware version of the HD-64.
- as an example run `hd64.debug.mark_bdln = 1`, this will mark all badlines on screen in cyan.
- as an example run `hd64.debug.mark_bdln = 0` to disable the badline debug feature.

The basic rules within the console are:

- memory ranges (MEMRANGE) can be indexed by word, so `hd64.flash_mng.firmware[12]` will access the 13th word of the firmware range. Words are 4-byte wide in the HD-64 firmware
- whole registers (REGISTER) are accessed through the property ".val" like for the `hd64.version.val` example seen above
- register fields (FIELD) are accessed by their name, like for the `hd64.version.fwv_minor` example seen above