Chameleon De-bricking Part II

First Steps with ChaCo, the Chameleon Control Software.

ChaCo is a Windows application that uses WxWidgets for the GUI. This makes it look different from other Windows applications, but it will give us the chance to compile it for Linux or for the Mac in the future. However, we will only provide a version for Windows for the first release. **If you require ChaCo for a different OS such as Mac or Linux, please do not buy the Chameleon now.**

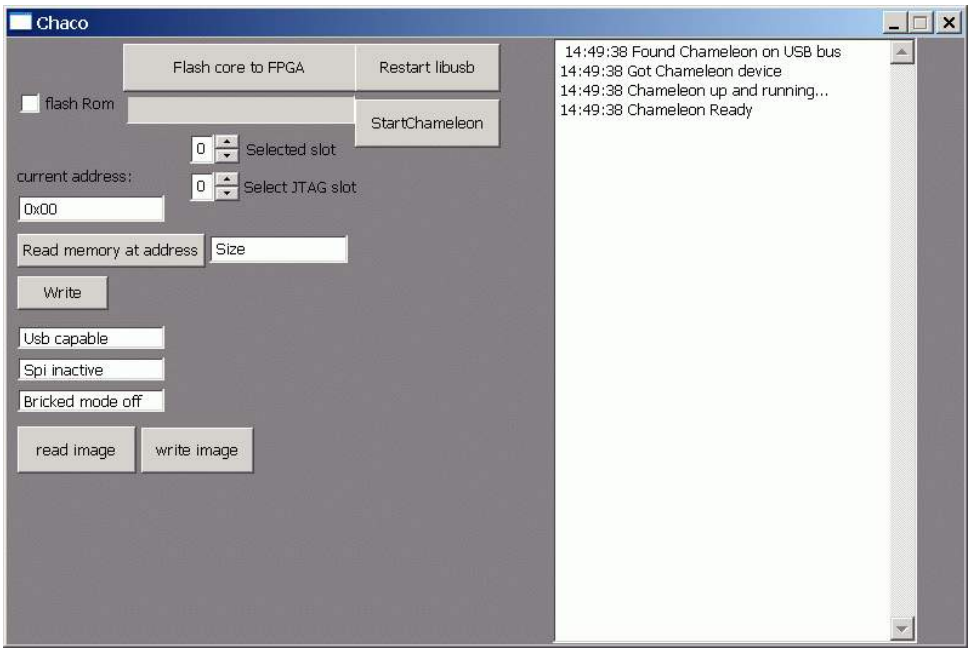
Before starting with ChaCo, you should know some details about the structure of Chameleon. After you learned about the USB microcontroller in Episode I, you may wonder what else may be "bricked" on Chameleon. There is a flash chip built into Chameleon that's purely for system purposes and hence can't be utilized by the user. The flash carries FPGA cores and their respective ROM files. The FPGA core is the true core of Chameleon's operations – it defines the hardware behaviour. If we fix a bug in the hardware (such as a VIC effect emulation or register functions of a freezer), we will provide the fix in the form of an FPGA core file.

The size of the onboard flash is 16 MByte, organized in 16 slots of 1 MByte each. Every slot can carry one FPGA core and the corresponding ROM file. If Chameleon is powered up, the FPGA is started from slot number 0. If starting the FPGA was not successful, then the microcontroller will tell you so by sending out a blink code on the red LED of Chameleon. The blink code will tell you which slot was attempted to be used. Just count the number of blinks, subtract one and you'll know which slot of the flash chip is corrupted. If for example slot 0 is corrupted, then the LED will flash once, then pause for a second and start over. If the LED blinks four times, then slot number three is the one you should take a closer look at.

ChaCo is an engineering tool at the moment. There is hardly any user-friendliness built into the tool at this point. We haven't put any time into catching user errors until now, so it's likely that you make a mistake and the tool won't even point you to it. You will need to follow the instructions closely in order to avoid such errors. **If you do not feel comfortable using a tool that's on such a low level, please do not buy Chameleon yet.**

One of the errors may be that you try to start ChaCo without a Chameleon connected to the computer. The window will open, it won't find Chameleon and therefore quit without further notice.

With a Chameleon connected, ChaCo will open with this screen:



..and here's a short description of the buttons:

Flash core to FPGA: This button start the flash procedure of writing a core to a selected slot. The name of this button may be changed to something more accurate in the future.

Restart libusb: This button will re-initialize libusb, a piece of software that is used to talk to the USB part of Chameleon. It is unlikely that you will need this button, but we required it quite often during development. Since we're still in development, this artefact of development is still there.

flash Rom: Tick this box if you want to add a rom file to the core file. Some cores may not need a rom file, some other cores will *require* a rom file to work properly. Please read the documentation of the core you're about to flash. Remember that this box must be ticked *before* clicking the “Flash core to FPGA” button.

Start Chameleon: Click this button to launch the selected slot.

Selected slot: Enter the number of the slot that you want to flash into this field.

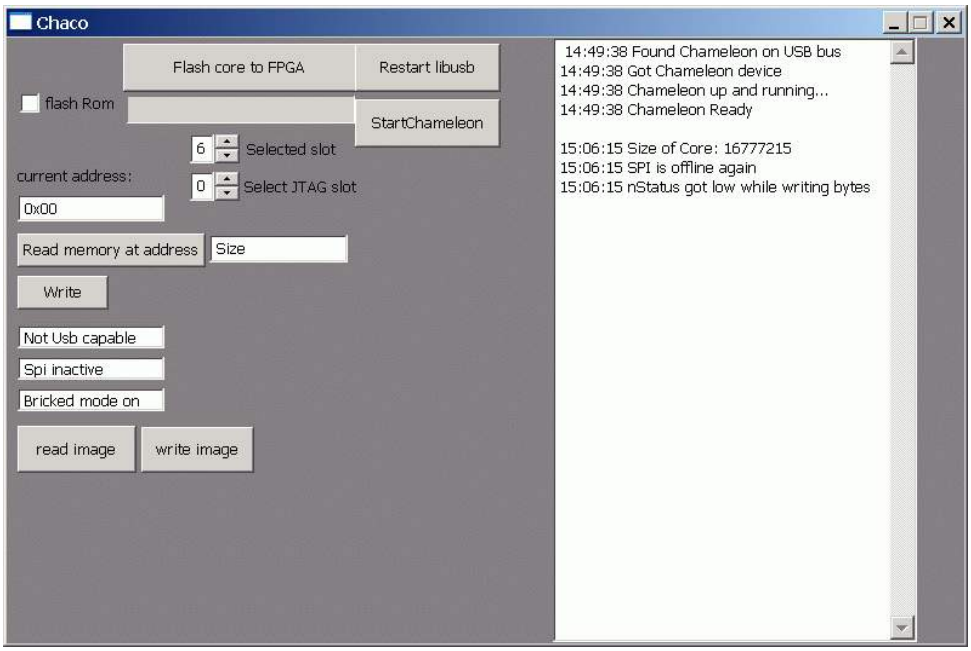
Select JTAG slot: This is a setting for FPGA developers. If you launch the FPGA through the JTAG interface, the microcontroller will report this number to the core. It will behave as if the FPGA was launched from this slot. Don't worry if you don't understand this setting – it's a developer thing, and ChaCo is an engineering tool ;-)

Current address, **Read memory at address**, **write**: These will let you access the memory of the running FPGA core (see description further down).

The three fields “USB capable”, “SPI inactive” and “Bricked mode off” are status fields. You cannot enter anything here.

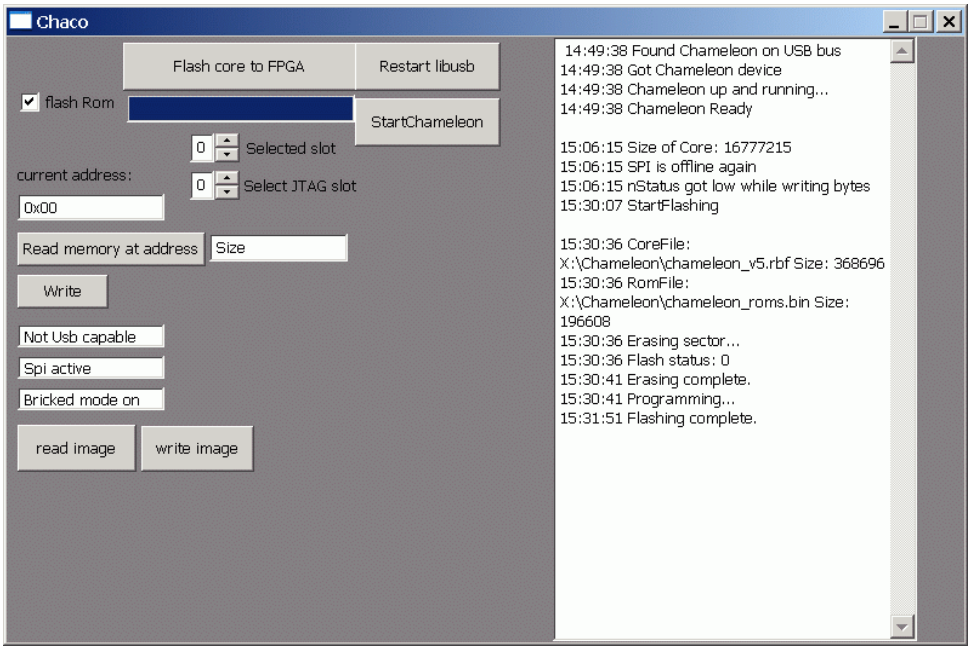
The buttons “read image” and “write image” will let you backup/restore the whole 16MByte flash into or from a single file.

Let's cause an error, so you can see the LED blink: Set “selected slot” to 6 and click on the “Start Chameleon” button.



The error message in the window reflects what you should see on Chameleon: The red LED should blink seven times, then pause, then start over with blinking.

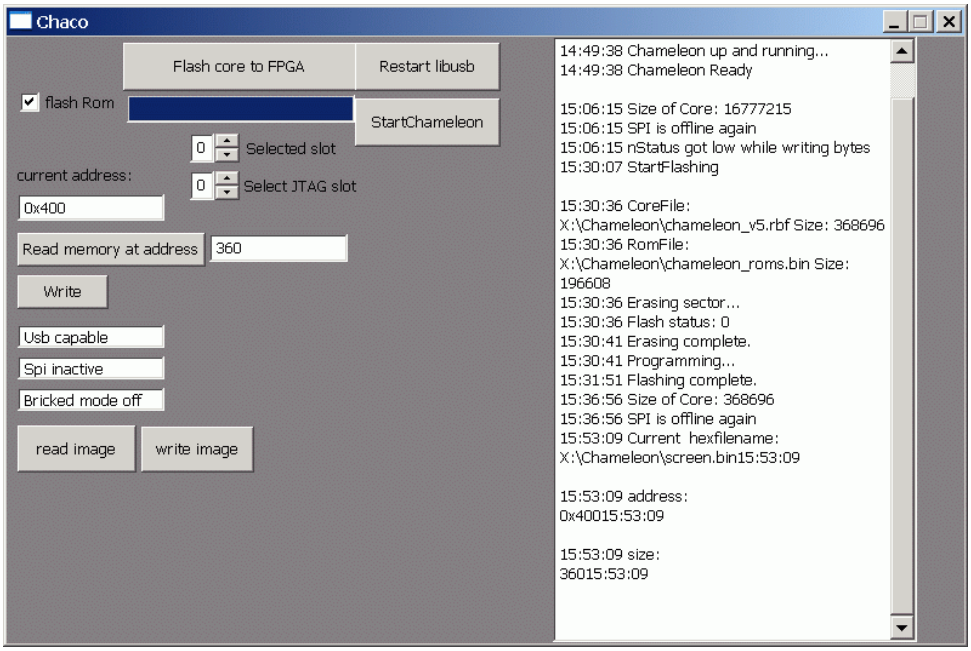
Now set the Selected slot number back to 0. Tick the box “flash Rom” and click on “Flash core to FPGA”. Two file requesters will open – please observe closely what each file requester is for! The first file you need to choose is the FPGA core, the second file is the ROM file that belongs to the core. These two files will be distributed with each Chameleon update. After choosing the second file, the flash process will be started without further notice. Again, this is an engineering tool, so we have to give another word of **Caution:** The contents of the flash slot will be erased without any user interaction!



Now click on “Start Chameleon” to test the core you've just flashed.

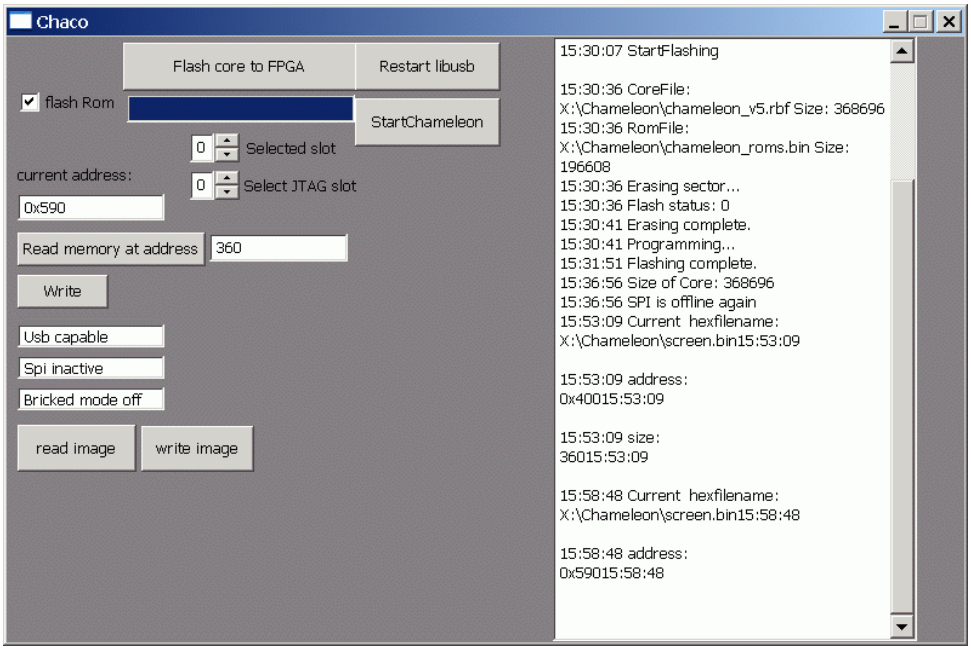
The current core includes a C64 with floppy, MMC64, Retro Replay and REU. The C64 will start with the MMC64 menu. Hit F7 to “exit to Basic” - this will bring you into the startup-screen of the Retro Replay. Hit F7 one more time to “install fastload”. You are now in the familiar “bluescreen” with some added debug-output:

Now let's use ChaCo for some remote debugging! Enter address 0x400 and size 360 into the fields for memory access:

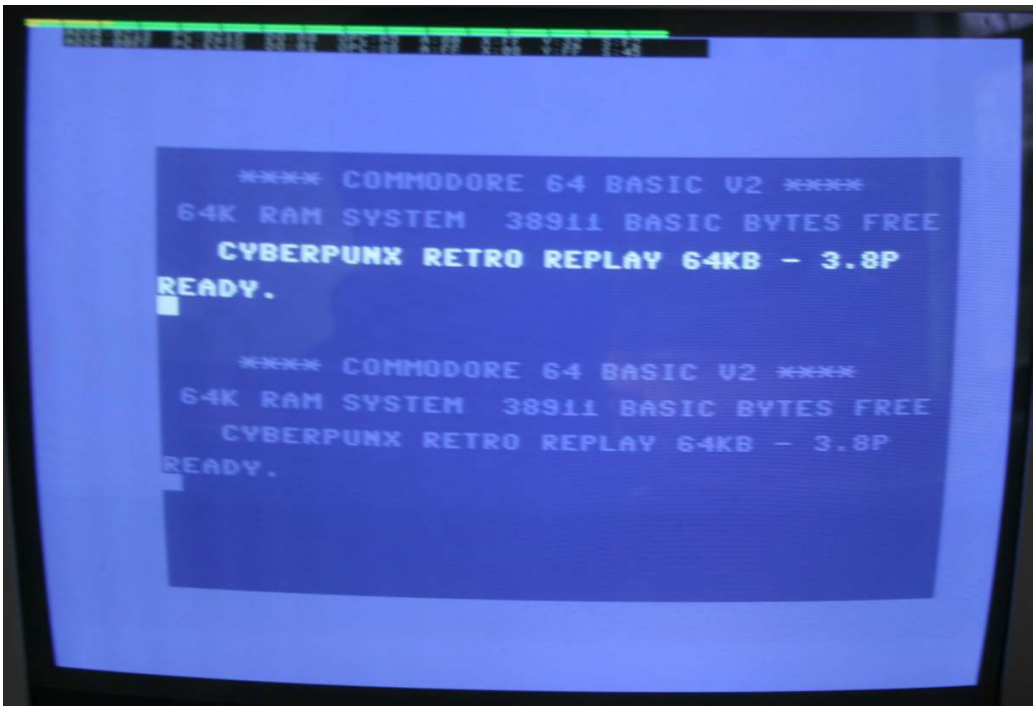


Then click on “Read memory at address”. ChaCo will read the memory of the C64 into a file that you can save to your PC's harddisk. Let's call this file “screen.bin”.

In a second step, we'll write back this same file to the C64 memory, but to a different address. Let's use address 0x590:



Then click on “Write” and choose the file you've just saved (screen.bin). The resulting display on the Chameleon VGA screen should be like this:



What you've just done is to copy the first nine lines of the screen memory to lines ten and following of the C64 screen memory. If you have some imagination, you will know that this feature holds a huge potential. However, this potential is not utilized by the current state of development. We do have plans to provide a fully-featured realtime debugger, but we're far from actually implementing it. **If you are not willing to wait and be a tester for all the forthcoming features, please don't buy the Chameleon yet.**