Ressurecting NXT 2.0 dead brick.

Andrey Borozdin

December 20, 2013

Abstract

This article is devoted to solving the problem of a dead (locked/clicking) NXT 2.0 brick. There are known ways to solve this problem using USB port of the brick, but sometimes it is impossible because brick can't be seen by computer. Here we discuss this particular case. We use JTAG port of the brick, flyswatter clone JTAG dongle and openOCD software to fix the problem.

Contents

1 Problem

NXT Intelligent Brick is a brick-shaped computer made by Lego for purpose of creating and programming robots. It's widely used in schools to teach kids robotics and programming. Switching of or plugging off the cable while flashing firmware is rather common situation there. When such thing happends, the brick might become locked, i.e. stop responding to any commands. There are ways, widely discussed in the internet, how to fix the problem using USB. But sometimes the bricks locks in some special way, so that it becomes impossible to fix the

problem using only USB. We are solving this particular problem.

It's well known that nowadays almost all ICs have JTAG port for debuggind. So the ICs inside NXT brick do. So, we will use JTAG port for low-level programming of the NXT brick.

2 What is inside NXT brick

Inside NXT brick you will find one board with everything on it except for the LCD.
There are two main controllers onboard: Atmel AT91SAM7S256 (LQFP64 package) and Atmel ATMEGA48 (TQFP44 package).
The first one is the heart of the NXT brick. It gathers all information from the sensors, motor encoders, it is connected to the USB and to the speaker.
ATMEGA48 generally controlls power. And also it checkes if there is AT91SAM7S256 controller on board. And if there is not (or dead) it switches off everything after 5 minutes.
Both controllers have debug ports, there are places for the debug connectors but there are no connectors. So, to fix the problem one will need to solder either wires or connector.

3 Tools I used

The tools I used to fix the problem are very numerous and you won't need them all, so I devided them into 2 lists.

Tools that I used

- Some small common instruments: screwdrivers, tweezers and so on.
- Oscilloscope. I used Owon SDS7102.
- Soldering station with hot air. I used Luckey 868.
- JTAG dongle. I used flyswatter clone made from Pinboard rev.II. `http://shop.easyelectronics.ru/index.php?productID=152`[17]
- Connector to solder onto the NXT brick mainboard.
- Self-mage adaptor to connect NXT brick mainboard to JTAG dongle.
- Some wires.
- Computer with OS Linux. I used my Lenovo T420 with Linux Mint 14 32-bit.
- OpenOCD open software. I used version 0.7 from official site.
- Lego firmware for NXT 2.0. File.rfw. Don't worry it is just binary format, so don't care about .rfw.

Minimal tools set

- Some small common instruments: screwdrivers, tweezers and so on.
- JTAG dongle. Any supported by OpenOCD.

- Some wires.
- Solder.
- Computer with OS Linux.
- OpenOCD open software.
- Lego firmware for NXT 2.0.

Anyway you'll need the solder to connect to the mainboard.
I my case, I needed to replace the AT91SAM7S256 controller, in this case you'll need some skills.

4 My ideas for tools choise

The main idea was ẗry to use what you already have:

Each vendor makes his own tools for flasing devices. So you can just order them and forget about trying to connect together things that were not made for that. But!

- The tools from Atmel are relatively pricey.
- I want to be able to flash any controller using my tools in future, not only Atmel, maybe STM32 or something else.

So I had Pinboard rev.II with STM32F103C8 ARM Cortex-M3 module from `http://easyelectronics.ru/`[16] which includes CooCox CoLink clone (`http://www.coocox.org/Colink.htm`[5]) JTAG dongle onboard for needs of flashing STM32F103C8.

(Board with that module: `http://shop.easyelectronics.ru/index.php?productID=152`[17])

Also I wanted to use Linux, so I was looking for universal free software for Linux. I found OpenOCD. But OpenOCD does not support CooCox CoLink. But thanks to this thread `http://forum.easyelectronics.ru/viewtopic.php?f=23&t=8059`[7] I found that CoLink is almost a clone of TinCanTools Flyswatter dongle. Both CoLink and Flyswatter are free to make, you can find their schematics on official websites.

So I decided to use my CoLink clone, but to make OpenOCD think that I'm useng flyswatter. By the way, there is very nice free soft from CooCox for compiling and flashing comtrollers (for Windows only), but unfourtunately it doesn't support AT91SAM7S256 the main NXT controller.

5 Tools: software

So the main tool we are going to use is OpenOCD. In my Linux Mint 14 I even can install it just using

```
sudo apt-get install openocd
```

But it installs version 0.5 and it doesn't support some commands. So let's download version 0.7 from here: `http://sourceforge.net/projects/openocd/files/openocd/0.7.0/`[9].
After you unzip it, you need to configure it with

```
./configure --enable-ft2232_libftdi --enable-ftdi
```

It might require some other packages like libusb and will tell you about that. If it does, just install them using apt-get.
You can find large user guide on OpenOCD here `http://openocd.sourceforge.net/doc/html/index.html`[10]
Also you will need a firware for NXT. You can dowload it just from official Lego site. As I said, don't worry aboit it's format it is just binary, so don't care about .rfw and rename to .bin.

6 Tools: hardware

So the first thing about hardware is the connection to the mainboard. I prefered to colder a connector to the mainboard at least on the first NXT brick for convinience.
The important fact is that you will need a rather rare (at least here in St.-Petersburg, Russia) type of connector. It's 2 times smaller than the common one and has 1.27 pitch. I found it here `http://chipster.ru/catalog/connectors/board-pin-connectors/` but with the following restrictions:
the male connectors are avaliable for surface mount only. I had to straighten out it's leads.
the female connectors are of large size, so I had to cut them and then to file them to look good.
I hope that it's the local problem of St.-Petersburg.
The next thing is making the CoLink look like Flyswatter. They are almost clones of each other, both of them use very-well known chip FT2232D from FTDI (`http://www.ftdichip.com/Products/ICs/FT2232D.htm`[18]). The only difference is place of SRST signal. (TRST differs too, but AT91SAM7S256 doesn't have TRST pin, so don't care). Flyswatter uses ADBUS5 pin of FT2232D to set SRST signal and ACBUS1 pin to listen it back. CoLink uses ACBUS1 pin to set SRST signal and nothing to listen it back. So I had to change the scematics of my adaptor.
Luckily, the adaptor in my case of Pinboard rev.II is made of tho parts: pinboard itself contains FT2232D chip, and small daughter board contains 74HCT541DW buffer. So to change schematics I just plugged the daughter board to pinboard with wires.
And finally I needed small adaptor co connect connector on JTAG dongle (with 2.54 pitch) and connector on NXT. I could make special wire, but I decided to make small adapter board which has 1.27 female connector on one side and 2.54 male connectors on the other.
If you would like to use FT2232 dongle, but don't have one I would advice you to look at this one: `http://easyelectronics.ru/universalnaya-plata-ft2232.html`. It's relatively cheap and contains only FT2232 soldered, so it's up to you to turn it into any dongle.

7 Using OpenOCD (general commands)

After you install openocd and connect everything, you can run debugging by

```
sudo openocd -f ./your_config_file.cfg
```

So -f key tells OpenOCD to take the config file, so you'll need one.
I use two of them: one to tell OpenOCD what dongle I'm using, the other to specify the target

(i.e. my chip), so I run:

```
sudo openocd -f ./flyswatter.cfg -f ./at91_my.cfg
```

So, lets discuss them

Listing 1: flyswatter.cfg

```
interface ft2232

#ft2232_device_desc "Flyswatter"
ft2232_layout "flyswatter"
ft2232_vid_pid 0x0403 0x6010
```

Here everything is simple:
First I tell the OpenOCD programm to use the ft2232 interface. The ability to use interfaces depends on the –enable-... keys passed to the configure script.
Then I had to comment specification of device description, because I use CoLink instead of Flyswatter. I could write right string here but anyway vid and pid are enough.
The next string specifies board layout. There are ways to create your layout in OpenOCD scrits instead of using wires, but wires are simpler.
The last line specifies vid and pid of my FT2232 chip. FT2232 is shipped with these values.

Listing 2: at91_my.cfg

```
source [find target/at91sam7sx.cfg]

telnet_port 4444
gdb_port 3333
adapter_khz 500
```

This config is even simpler. It just relies on what is written in at91sam7sx.cfg file which comes with OpenOCD and in my case can be found in /usr/local/share/openocd/scripts/target/.
Then it specifies ports for telnet and gdb and the JTAG speed in Khz.
The last line is important. The higher speed the faster we can flash the controller but if it's too high we won't be able to flash anything. In case of NXT 500 Khz works.
So, after checking all the connections, run

```
sudo openocd -f ./your_config_file.cfg
```

You will get reply from OpenOCD like this:

Listing 3: Openocd started

```
TEXT
```

Note that there should not be any errors reported. If there are any, refer to the OpenOCD manual.
If everything is OK you can run telnet client. Open another terminal and type

```
telnet 127.0.0.1 4444
```

Where 127.0.0.1 is the localhost address and 4444 is the port specified in at91_my.cfg.
You should get the reply:

Listing 4: Openocd debugger reply

```
TEXT2
```

To flash the chip or make any other actions you should type commands here. Refer to OpenOCD user guide for full list of commands. We will deed some small amount but before we start we must put our chip into flashable state. To do that execute the following (I found it in at91sam7sx.cfg file that comes with OpenOCD):

```
soft_reset_halt
mww 0xfffffd00 0xa5000004
mww 0xfffffd44 0x00008000
mww 0xfffffd08 0xa5000001
mww 0xffffffc20 0x00000601
sleep 10
mww 0xffffffc2c 0x00481c0e
sleep 10
mww 0xffffffc30 0x00000007
sleep 10
mww 0xffffff60 0x00490100
```

You should not get any reply for that.
To get chip state (halted/running) type

```
poll
```

To get information about flash use

```
flash info 0
```

We ask information about flash bank number 0. The AT91SAM7S256 controller doesn't have any other banks by the way.
On this command you should get reply like this:

Listing 5: Flash info 0 debugger reply

```
#0 : at91sam7 at 0x00100000, size 0x00040000, buswidth 4, chipwidth
    0
        #  0: 0x00000000 (0x4000 16kB) not protected
        #  1: 0x00004000 (0x4000 16kB) not protected
        #  2: 0x00008000 (0x4000 16kB) not protected
        #  3: 0x0000c000 (0x4000 16kB) not protected
        #  4: 0x00010000 (0x4000 16kB) not protected
        #  5: 0x00014000 (0x4000 16kB) not protected
        #  6: 0x00018000 (0x4000 16kB) not protected
        #  7: 0x0001c000 (0x4000 16kB) not protected
        #  8: 0x00020000 (0x4000 16kB) not protected
        #  9: 0x00024000 (0x4000 16kB) not protected
        # 10: 0x00028000 (0x4000 16kB) not protected
        # 11: 0x0002c000 (0x4000 16kB) not protected
        # 12: 0x00030000 (0x4000 16kB) not protected
        # 13: 0x00034000 (0x4000 16kB) not protected
```

```
        # 14: 0x00038000 (0x4000 16kB) not protected
        # 15: 0x0003c000 (0x4000 16kB) not protected

at91sam7 driver information: Chip is AT91SAM7S256
Cidr: 0x270b0941 | Arch: 0x0070 | Eproc: ARM7TDMI | Version: 0x001
   | Flashsize: 0x00040000
Master clock (estimated): 10428 KHz | External clock: 4000 KHz
Pagesize: 256 bytes | Lockbits(16): 0 0x0000 | Pages in lock region
   : 0
Securitybit: 0 | Nvmbits(2): 0 0x0
```

Note some important things here:

```
#0 : at91sam7 at 0x00100000, size 0x00040000, buswidth 4, chipwidth 0
```

tells us that flash has size 0x40000 (which is 256Kb) and starts from address 0x00100000. We'll need this info for flashing/reading commands.
This message:

```
Master clock ( estimated ) : 10428 KHz
```

means that clock generator inside the chip is working and the frequency is about 10Mhz.

Now we are ready to flash the controller.

8 Flashing AT91SAM7S256

The controller can be flashed from telnet client after all the preparations with just two commands:

```
flash write_image erase unlock "new_firmware.bin" 0x00100000 bin
dump_image "file_dump.bin" 0x00100000 0x40000
```

The first one writes file new_firmware.bin into the flash. The second one dumps the whole flash to file file_dump.bin.
Note that files fill be searced and created in the directory from which you started OpenOCD itself, not telnet!
To flash controller execute them and then check if file file_dump.bin is equal to new_firmware.bin. You can do this with diff command on Linux. And if they are equal - we win! Type

```
shutdown
```

and turn NXT brick on in usual way.

9 Replacing AT91SAM7S256

This section is about what to do if after flashing and dumping the dumped file is not qual to flashed one.
That was exactly my case! I tried hard but could not understand what kind of bug I have. The controller was flashed without errors, but when I dumped firmware, it contained certain differences from the file I tried to write.

So I decided to replace the controller. I bought a new one here `http://www.chipdip.ru/` for 340 RUR (7.5 EUR or 10.5 USD) and replaced it. If you are going to do the same, be sure you have enough skills. It's not so hard ro take away old chip using hot air (I use Luckey 868 340 degrees maximum float by the way), but it's not so easy to solder a new chip.
In my case, after replacement I performed all steps above and it finally worked.

10 Strange bug encountered in my AT91SAM7S256

While trying to flash the AT91SAM7S256 I encountered a very strange bug. After all preparations, I managed to run flash erite command and not get errors. But when I read it back I found that the first byte in memory, which address ended on 0xFB didn't write well. After this byte the whole flash contained a strange mess of bytes.
So I tried to erase the flash, but common erasing commands didn't work. So I tried the command

```
flash fillb address byte length
```

which fills memory from address to address + length with supplied byte.
In my case it looked like:

```
flash fillb 0x00100000 0xFF 0x100
```

I got reply that fillb failed on 0x001000fb. Then I tried

```
flash fillb 0x00100100 0xFF 0x100
```

I got reply that fillb failed on 0x001001fb.
And so on. Bytes with address ending on 0xfb didn't write properly.
I still don't know what it was. After I changed controller all workes well.

11 Conclusion

I hope that these instructions will help you to ressurect dead NXT bricks. I also think that making JTAG connector on the brick and conneting to the mainboard by wires, so that kids could try their skills in JTAG debugging.
I found a lot of manuals, but could not use any of them to the letter. So I just put them into the bibliography.
Please send me any questions/feedback borozduckens@gmail.com (to Andrey Borozdin).

References

[1] *Lego official downloads page.* http://www.lego.com/en-us/mindstorms/downloads/software/ddsoftwaredownload/

[2] *IAR page with NXT source code.* http://www.iar.com/mindstorms/

[3] *TinCanTools Flyswatter page.* http://www.tincantools.com/JTAG/Flyswatter.html

[4] *Flyswatter wiki page.* http://www.elinux.org/Flyswatter

[5] *CooCox Free/Open ARM Cortex MCU Development Tools.* http://www.coocox.org/Colink.htm

[6] *NXTGCC project.* http://nxtgcc.sourceforge.net/nxtgcc.pdf

[7] *Easyelectronics forum thread on using OpenOCD.* http://forum.easyelectronics.ru/viewtopic.php?f=23&t=8059

[8] *Accessing ARM-Controllers with OpenOCD.* http://siwawi.bauing.uni-kl.de/avr_projects/arm_projects/openocd_intro/

[9] *OpenOCD Project Downloads Page.* http://sourceforge.net/projects/openocd/files/openocd/0.7.0/

[10] *OpenOCD User's Guide.* http://openocd.sourceforge.net/doc/html/index.html

[11] *Configuring OpenOCD for the AT91SAM7SE.* http://www.ethernut.de/en/hardware/eir/openocd.html

[12] *Using OpenOCD with TELNET connection.* https://www.olimex.com/Products/ARM/JTAG/_resources/Manual_TELNET.pdf

[13] *AT91SAM7S mit OpenOCD programmieren.* http://www.mikrocontroller.net/articles/AT91SAM7S_mit_OpenOCD_programmieren

[14] *Ada User Guide for LEGO MINDSTORMS NXT.* http://www.dit.upm.es/~str/proyectos/mindstorms/2012/auj11.pdf

[15] *Atmel AT91SAM7S ARM Microcontroller Framework.* http://tech.munts.com/MCU/Frameworks/ARM/at91sam7s/

[16] *Easy electronics main page.* http://easyelectronics.ru/

[17] *Easy electronics shop: PinBoard II R2 STM32.* http://shop.easyelectronics.ru/index.php?productID=152

[18] *FT2232D - Dual USB UART/FIFO IC.* http://www.ftdichip.com/Products/ICs/FT2232D.htm