# Reset of Leonardo boards #30

New issue

Closed joh opened this issue on Jan 4, 2013 · 37 comments

**joh** commented on Jan 4, 2013

Reset of Leonardo type boards is done a bit differently:

> Rather than requiring a physical press of the reset button before an upload, the Leonardo is designed in a way that allows it to be reset by software running on a connected computer. The reset is triggered when the Leonardo's virtual (CDC) serial / COM port is opened at 1200 baud and then closed. When this happens, the processor will reset, breaking the USB connection to the computer (meaning that the virtual serial / COM port will disappear). After the processor resets, the bootloader starts, remaining active for about 8 seconds.

Please consider adding support for this to **ard-reset-arduino**. I'm no Perl wiz, but the following Python code did the trick:

```
import serial

ser = serial.Serial("/dev/ttyACM0", 1200)
ser.open()
ser.close()
```

Cheers!

Labels

None yet

Milestone

0.10.5

Assignee

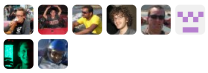
No one assigned

Notifications

Subscribe

You're not receiving notifications from this thread.

8 participants



**guicho271828** commented on Apr 5, 2013

I encountered the same problem.

**guicho271828** commented on Apr 5, 2013

I fixed the problem with your code joh, along with minor fixes. see #349d254

**guicho271828** commented on Apr 5, 2013

sorry, it was #37

**sudar** commented on Jun 8, 2013 Owner

I have created a separate branch by taking **@guicho271828** code. https://github.com/sudar/Arduino-Makefile/tree/leonardo

Kindly check it and let me know if it works properly. Will merge it with master once it is confirmed to work.

**sudar** closed this issue from a commit on Jun 11, 2013

Add support for leonardo. ⋯ f26a134

**sudar** closed this in f26a134 on Jun 11, 2013

**maleadt** commented on Jun 20, 2013

Shouldn't this also be the case for the Arduino Micro ( `BOARD_TAG = micro` )?

> Rather than requiring a physical press of the reset button before an upload, the Micro is designed in a way that allows it to be reset by software running on a connected computer. The reset is triggered when the Micro's virtual (CDC) serial / COM port is opened at 1200 baud and then closed. When this happens, the processor will reset, breaking the USB connection to the computer (meaning that the virtual serial / COM port will disappear). After the processor resets, the bootloader starts, remaining active for about 8 seconds. The bootloader can also be initiated by pressing the reset button on the Micro. Note that when the board first powers up, it will jump straight to the user sketch, if present, rather than initiating the bootloader.

I just tested the new `ard-reset-leonardo` , and it also works for an Arduino Micro.

**sej7278** commented on Jun 20, 2013 Collaborator

there is no ard-reset-leonardo now, we merged the functionality into the makefile as part of commit `60ca7d2`

could you try `ard-reset-arduino --leonardo /dev/ttyUSB0` (needs 0.10-6 makefile or later) and if that works we can add the micro tag to the leonardo routine

**sudar** referenced this issue on Jun 20, 2013

**Add support for resetting "micro"** #80 Closed

**sudar** commented on Jun 20, 2013 Owner

> Shouldn't this also be the case for the Arduino Micro (BOARD_TAG = micro)?

The only reason why this is not added is because I don't have a micro to test it with :)

> I just tested the new ard-reset-leonardo, and it also works for an Arduino Micro.

We have rewrote the way reset is handled for leonardo and the bin/ard-reset-leonardo script is merged inside bin/ard-reset-arduino itself in commit `60ca7d2` . Also check out #62

Can you kindly test whether this works for micro as well. If yes, then I can do the required code changes and push an update.

I have also created #80 to track this.

**sudar** commented on Jun 20, 2013 Owner

**@sej7278** You beat me by a couple of minutes ;)

**sej7278** commented on Jun 20, 2013 Collaborator

mwahaha, you can't beat me even with your time machine!

i'll have a pull request up in a mo as soon as i can think of a way to check for leonardo OR micro in an ifeq()

**maleadt** commented on Jun 20, 2013

Strangely, the new `ard-reset-arduino` script does not work properly on my Micro, at least not in all circumstances. More specifically, when something has been using the serial port (boblight in my case), `ard-reset-ardiuno` doesn't manage to reboot the Micro, while the Python script which was used in `ard-reset-leonardo` does work (even without the redundant open & close calls it originally contained):

```
#! /usr/bin/python

import sys
import serial

ser = serial.Serial(sys.argv[1], 57600)
ser.setBaudrate(1200)
ser.close()
```

If the Micro had been idle, `ard-reset-arduino` succeeds in rebooting the device. Strange.

**sej7278** commented on Jun 20, 2013 Collaborator

can you try modifying the old ard-reset-leonardo python script so that it starts at 1200 baud instead of 57600, as there was something about having to change from the initial baud rather than just starting at 1200bps in some comment somewhere. so i guess:

```
#!/usr/bin/python

import sys
import serial

ser = serial.Serial(sys.argv[1], 1200)
ser.setBaudrate(1200)
ser.close()
```

if that fails to reset, then we can modify ard-reset-arduino --leonardo to start at 57600 and change to 1200

**maleadt** commented on Jun 20, 2013

Nope, doesn't matter. So, this works:

```
#! /usr/bin/python

import sys
import serial

ser = serial.Serial(sys.argv[1], 1200)
ser.close()
```

... while this doesn't (always):

```
#!/usr/bin/perl

use strict;
use warnings;

use Device::SerialPort;

my $portName = shift || die();
my $port = new Device::SerialPort($portName, 0) or die();
$port->baudrate(1200) or die();
$port->write_settings() or die();
$port->close() or die();
```

**sej7278** commented on Jun 20, 2013 Collaborator

can you try your perl without the dies as you have to open, set baud, close; and if one of those dies it won't reset. but i've not idea why otherwise.

**maleadt** commented on Jun 20, 2013

I added the `die()` statements in case one of the commands would fail; but none of them fail, so removing the `die()` statements doesn't change anything...

**sej7278** commented on Jun 20, 2013 Collaborator

well if none of the commands fail, that's very odd. points to an OS issue perhaps, what platform is this - don't say fscking OSX! ;)

**maleadt** commented on Jun 20, 2013

Heh no, it's Linux 3.8, running under Ubuntu 13.04. I'd rather think it to be a Device::SerialPort bug, but I don't have time to look at it myself...

**mjoldfield** commented on Jun 20, 2013 Collaborator

If you want to debug this further, it might be worth logging the system calls made by the python and Perl programs. On linux strace should do this.

**sej7278** commented on Jun 20, 2013 Collaborator

**@mjoldfield** strace is a good idea thanks.

**@maleadt** I did have a bit of a grok around pySerial to compare with Device::SerialPort but its too much work for the moment. strace might show something obvious hopefully - probably something not freed or blocking the filehandle, or maybe something python does silently.

i'm not even sure if wait-connection-leonardo is required these days.

**sudar** commented on Jun 20, 2013 Owner

I tried to reset leonardo by using both perl and python in Ubuntu.

Both the scripts reset the board properly for me.

I have also run strace and have attached the output of both perl and python as well as the code I used in this gist

https://gist.github.com/sudar/5823250

One observation I had right away is that the trace of python has more than double the number of lines when compared with the perl version

**sej7278** commented on Jun 20, 2013 Collaborator

they're both opening in the same way, perl seems to be going over-the-top changing baud rates and then failing to close the filehandle:

```
20:03:03.849542 open("/dev/ttyACM0", O_RDWR|O_NOCTTY|O_NONBLOCK|O_LARGEFILE) = 3 <0.001769
20:03:03.851439 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:03:03.851527 _llseek(3, 0, 0xbfb8cf30, SEEK_CUR) = -1 ESPIPE (Illegal seek) <0.000020>
20:03:03.851589 fstat64(3, {st_mode=S_IFCHR|0666, st_rdev=makedev(166, 0), ...}) = 0 <0.00
20:03:03.851663 fcntl64(3, F_SETFD, FD_CLOEXEC) = 0 <0.000008>
20:03:03.851712 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:03:03.851838 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:03:03.851890 ioctl(3, SNDCTL_TMR_START or TCSETS, {B9600 -opost -isig -icanon -echo ...
20:03:03.851944 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B9600 -opost -isig -icanon -echo
20:03:03.852053 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B9600 -opost -isig -icanon -echo
20:03:03.852106 ioctl(3, SNDCTL_TMR_START or TCSETS, {B1200 -opost -isig -icanon -echo ...
20:03:03.852347 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:03:03.852437 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:03:03.852490 ioctl(3, SNDCTL_TMR_START or TCSETS, {B1200 -opost -isig -icanon -echo ...
20:03:03.852543 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:03:03.852606 ioctl(3, TCFLSH, 0x2) = 0 <0.000010>
20:03:03.852676 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:03:03.852728 ioctl(3, SNDCTL_TMR_START or TCSETS, {B1200 -opost -isig -icanon -echo ...
20:03:03.852781 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:03:03.852835 close(3) = 0 <0.017473>
20:03:03.870372 close(3) = -1 EBADF (Bad file descriptor) <0.000014>
```

python just does:

```
20:02:51.924738 open("/dev/ttyACM0", O_RDWR|O_NOCTTY|O_NONBLOCK|O_LARGEFILE) = 3 <0.001577
20:02:51.926379 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:02:51.926504 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:02:51.926559 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:02:51.926612 ioctl(3, SNDCTL_TMR_START or TCSETS, {B1200 -opost -isig -icanon -echo ...
20:02:51.927346 ioctl(3, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo
20:02:51.927409 close(3) = 0 <0.017918>
```

**sej7278** referenced this issue from a commit in sej7278/Arduino-Makefile on Jun 20, 2013

Added Micro to the Leonardo reset functions ⋯ `2abed1e`

**sej7278** referenced this issue on Jun 20, 2013

**Added Micro to the Leonardo reset functions** #83 Closed

**sej7278** commented on Jun 20, 2013 Collaborator

I've added a pull request just to add the Micro to the same reset routine as the Leonardo.

We can revisit why its not working all the time when we have some time to debug, but with my Boarduino I cannot get it to fail once.

**sudar** referenced this issue from a commit on Jun 21, 2013

Add support for reseting "Micro" Arduino. ⋯ `c284ec9`

**sudar** commented on Jun 21, 2013 Owner

> I've added a pull request just to add the Micro to the same reset routine as the Leonardo.

Thanks. I just merged it.

**matthijskooijman** commented on Jul 10, 2013

It seems there is still an issue with the Leonardo, mine will not properly reset on my Linux machine:

```
/home/matthijs/docs/Electronics/Arduino/Arduino-Makefile/bin/ard-reset-arduino --caterina
(...)
Connecting to programmer: .
Found programmer: Id = "TESTATA"; type = A
```

Manually running ard-reset-arduino and then checking dmesg also shows the device is not properly resetting.

**matthijskooijman** commented on Jul 10, 2013

My strace log (made with `-e trace=file,ioctl,close`) is a bit different, though:

```
stat("/dev/ttyACM0", {st_mode=S_IFCHR|S_ISVTX|0660, st_rdev=makedev(166, 0), ...}) = 0
open("/dev/ttyACM0", O_RDWR|O_NOCTTY|O_NONBLOCK) = 4
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B57600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B57600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B57600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_START or TCSETS, {B9600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B9600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B9600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_START or TCSETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_START or TCSETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, TCFLSH, 0x2)                   = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_START or TCSETS, {B57600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B57600 -opost -isig -icanon -echo ...}) = 0
close(4)                                = 0
close(4)                                = -1 EBADF (Bad file descriptor)
Forcing reset using 1200bps open/close on port /dev/ttyACM0
stat("/dev/ttyACM0", {st_mode=S_IFCHR|S_ISVTX|0660, st_rdev=makedev(166, 0), ...}) = 0
/dev/ttyACM0 has come back after reset
close(3)
```

The close works, but it actually resets the baudrate to 57600 before closing it seems :-S

**matthijskooijman** commented on Jul 10, 2013

Oh wait, the close didn't work, I was looking at the wrong close :-)

**matthijskooijman** commented on Jul 10, 2013

Also, it seems that just before the close, the perl library resets the baudrate to whatever it was at the start. This doesn't happen in the traces posted by sej7278, but I suspect this is because his python script left the baudrate at 1200 and doesn't try to change it back.

Also, it seems the descriptor is closed twice: The first time it works, the second time it (obviously) fails.

**matthijskooijman** commented on Jul 10, 2013

Just running `stty -F /dev/ttyACM0 1200` works to reset my Leonardo and afterwards the ard-reset-arduino script also works. I suspect that this means that the "restore baudrate before close" thingy actually messes up the reset and that the reset is triggered by the baudrate set at port close time.

**sudar** commented on Jul 10, 2013 Owner

Yeah even I was facing this issue when I try to reset leonardo using Perl.

If I use the Python script, then it resets all the time. But if I use Perl script, then it resets, only if it was previously reset using Python script or by using `stty` or through Arduino IDE.

**matthijskooijman** commented on Jul 10, 2013

Heh, apologies for the spam run, but I think I've found a solution. Not sure if I like it, though, but at least it confirms my previous suspicions.

I just added an explicit file descriptor close in the script, which circumvents the "restore baudrate" feature. The relevant code now looks like:

```
$p->baudrate(1200);
$p->write_settings;
POSIX::close($p->{FD});
$p->close;
```

Some things I wonder:

- Does this script need to run on !Linux and does this work there?
- Wouldn't it make sense to actually restore the baudrate *after* the device has reset? It seems that this 1200 baud now sticks around, even accross the reset detach/reattach of ttyACM0. OTOH, if you use a terminal emulator or avrdude, it will set the baudrate anyway, so it shouldn't hurt to leave it at 1200.

The strace now looks like:

```
stat("/dev/ttyACM0", {st_mode=S_IFCHR|S_ISVTX|0660, st_rdev=makedev(166, 0), ...}) = 0
open("/dev/ttyACM0", O_RDWR|O_NOCTTY|O_NONBLOCK) = 4
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B57600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B57600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B57600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_START or TCSETS, {B9600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B9600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B9600 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_START or TCSETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_START or TCSETS, {B1200 -opost -isig -icanon -echo ...}) = 0
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, {B1200 -opost -isig -icanon -echo ...}) = 0
close(4)                                = 0
ioctl(4, TCFLSH, 0x2)                   = -1 EBADF (Bad file descriptor)
ioctl(4, SNDCTL_TMR_TIMEBASE or TCGETS, 0x7fff9f375050) = -1 EBADF (Bad file descriptor)
ioctl(4, SNDCTL_TMR_START or TCSETS, {B57600 -opost -isig -icanon -echo ...}) = -1 EBADF (
close(4)                                = -1 EBADF (Bad file descriptor)
close(4)                                = -1 EBADF (Bad file descriptor)
Forcing reset using 1200bps open/close on port /dev/ttyACM0
stat("/dev/ttyACM0", {st_mode=S_IFCHR|S_ISVTX|0660, st_rdev=makedev(166, 0), ...}) = 0
/dev/ttyACM0 has come back after reset
```

**sudar** commented on Jul 10, 2013 Owner

> Wouldn't it make sense to actually restore the baudrate after the device has reset?

The Arduino IDE does this. They explicitly restore the bautrate to 57600 after reset.

**matthijskooijman** commented on Jul 10, 2013

Perhaps it makes sense to drop Device::SerialPort and just do a few calls to POSIX:: directly?

**matthijskooijman** commented on Jul 10, 2013

Or convert to Python altogether :-)

**sudar** commented on Jul 10, 2013 Owner

> Or convert to Python altogether :-)

I am leaning towards this, instead of `POSIX::` calls. Because we can support windows users as well if we choose Python.

**sej7278** commented on Jul 10, 2013 Collaborator

When you say Windows users, how does any of this actually work on Windows - does it use cygwin or something? I think using stty from the Makefile directly is probably preferable to a perl/python script but doubt its very portable.

maybe we should drop windows support (nobody volunteered to test it even works) as they can use the CLI support in IDE 1.5 anyway, are windows users really going to install cygwin/gcc, perl/python and libraries to use this? at best we should limit ourselves to only using what comes bundled with the IDE on windows, which i assume is avr-gcc and avrdude and not much else.

the Device::SerialPort docs seem a bit unreliable as the call to destroy() doesn't even work as its already closed using close()

**sudar** commented on Jul 10, 2013 Owner

Technically if cygwin is installed, then most of what we have should work in Windows. But as you said, no one has tested it so far, so I don't know for sure.

The reason, why I said I would prefer Python over using `POSIX::` calls is that, Python and PySerial are supported in windows and if someone becomes interested in using this in Windows at some point in future, the amount of effort required to add full support for Windows will be less.

This was referenced on Nov 14, 2013

**Upload to Arduino Micro from Raspberry Pi failed with 'programmer is not responding'** #127 Closed

**Makefile does not detect change in com port when Leonardo is resetted** #173 Closed

**sej7278** commented on Mar 22, 2014 Collaborator

I've added the regular arduino (DTR) functionality from ard-reset-arduino into the old python ard-reset-leonardo and tweaked it a bit. it now seems to work on my pro micro and mega2560.

could people please test it on a few platforms/boards please? its on gist

**jonnor** referenced this issue in **microflo/microflo** on Aug 17, 2014

**Flashing: Implement software-reset of Arduino Leonardo/Micro** #58 Open

**RichardBronosky** commented on Sep 30, 2014

**@matthijskooijman**, **@sudar**, do you find that reseting a Leonardo or [ Sparkfun Pro] Micro requires 8 seconds? Is there a way around that?

Thanks for the stty suggestion. I have created https://gist.github.com/RichardBronosky/0500556194a45f2d4855#file-leoreset-sh that I use so I can reset with the device tethered from inside my backpack.

Write Preview Markdown supported

Leave a comment

Attach files by dragging & dropping or selecting them.

Comment