LeafLabs Maple QA and Preparation Procedures

Bryan Newbold <bnewbold@leaflabs.com>

Last modified 07/27/2010

There are several degrees of quality assurance testing to be done on newly fabricated Maple boards. First, after any design change, there is in-depth electrical characterization and feature testing to be done. Second, with any new batch of boards produced (especially by a new manufacturer or alternative components) there is a comprehensive electrical checklist. Finally, every single board that is sold must be flashed with a bootloader and initial program, then run through a rapid electrical test. This document describes the steps for the later two; characterization of new features or design changes is a more subtle task for an engineer.

See the end of this document for a per-board testing checklist.

Per-batch Electrical Testing Instructions

The intent of this testing is to ensure that the boards work "as advertised" with the manufacturing techniques and sourced components that are usually in some way unique to a given batch of boards. For example, if a cheaper oscillator crystal with all the same specifications from a different source is used, all the functionality of the board needs to be tested.

This testing isn't as straight forward as the per-board directions below. It's up to the technician/engineer to be a detective and search out problems; we usually spend an hour or two running these tests and writing up the results. Tests should be run on at least 2-3 boards randomly selected from the batch, with at least one NOT from the top or bottom sheet. Common tests are:

- Test EXT power up to the rated voltage (maximum and minimum)
- SerialUSB and USART read/write at maximum speeds, looking for corruption or dropped data
- Look at PWM and communications waveforms with an oscilloscope; check for excess noise/ringing/crosstalk
- Test power consumption
- Test battery charging circuit and status LEDs
- Inspect the V_{cc} and V_{analog} regulated voltages for accuracy and USB and USART crosstalk using an oscilloscope.
- Test PWM on all pins labeled as such
- Test ADC input on all pins labeled as such with GND, ~1.5v (battery), and V_{cc}. All pins should make statistically comparable readings.
- Check GPIO read/write on all pins
- Look at the oscillator waveform on an oscilloscope
- Let the board crunch numbers, charge battery, and USB read/write for a couple minutes and make sure no components get hot

Testing ADC noise and accuracy is the most ambiguous; it's tricky to do without an communications channel running which will increase crosstalk noise, and we haven't researched the STM32 ADC peripheral enough to distinguish between systematic noise in our design and improper sampling technique (eg, mismatched impedance).

After all this testing, the first 10 boards that go through the per-board testing may turn up additional issues, so it's helpful to get those prepared at the same time to catch all problems as soon as possible.

Per-board Preparation and Testing Instructions

These tests should be run on every single Maple board that will be shipped. When the customer receives the board it should be in a static baggie labeled with the revision number and instructions (or a URL to instructions) on it, the board should have two jumpers on the power select header (battery charge "open"), and the board should be programmed with both the bootloader and a recent version of the interactive test program.

In preparation, you'll need:

- a partially charged 3.3v LiPo battery with a compatible connector
- a computer with all the required software
- Mini-B USB cable
- Serial-USB adapter (FTDI) with jumper wires to connect to Maple USART
- a supply of header jumpers (2x per board)
- a "test shield" (a modified Maple board with different headers; instructions below)

This entire process goes better as an assembly line with one person at a computer doing software and another person doing the more mechanical tests and attaching/removing the test shield. For speed we usually put aside any board that seems suspicious at all with a post-it note or tape attached explaining the issue; we then revisit these boards at the end or pass them off to an engineer for investigation or repair. We average something like 1 in 12 boards being flakey for any reason; some of these may even be due to human error during testing (eg, twitchy hands, keyboard typos, paranoia, whatever). These boards are still appropriate for software development or internal projects, so we retest or fix them quickly by hand but never ship them. Any errors on the part of the manufacturer are noted and communicated as feedback to help them with their QA/process.

Note that the Maple rev4 is identical to the rev3 except for silkscreen fixes. The rev3 needed part of the silkscreen erased or sharpied over during preparation.

Brief visual inspection of board

This first step doesn't make much sense until you pick up a board and notice that one of the buttons just isn't there or that the silkscreen is out of focus, or that two headers are soldered together, etc. The care that should be taken depends on how much the batch is trusted as a whole: things to look out for should be identified in the per-patch inspection. The most common things we have trouble with are the buttons (give them firm nudge and make sure they don't pop off), shorts between STM32 pins, bad soldering on the female headers, and big solder lumps on the barrel jack connector.

For the first 10 boards in a batch it's worth spending 30 seconds a board and perhaps using a loupe or microscope. After that it's usually a 10 second glance to check the silkscreen and any large issues. The later testing steps should catch most other problems.

Attach Jumpers

Usually the power select jumpers are not attached during production and are added at this stage. The power select jumper should be on USB and the battery charger should be half attached.

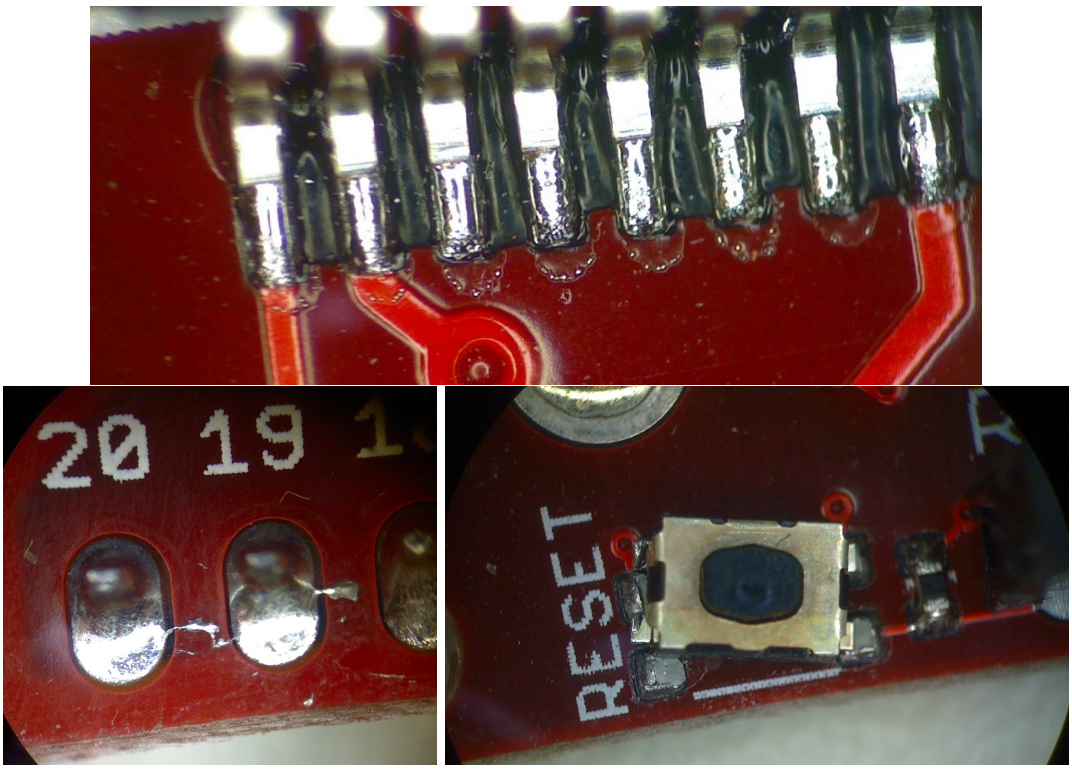

Figure 1: A selection of defective Maple boards: solder shorts (left), un-melted solder paste (top), and an overheated Reset button (right)

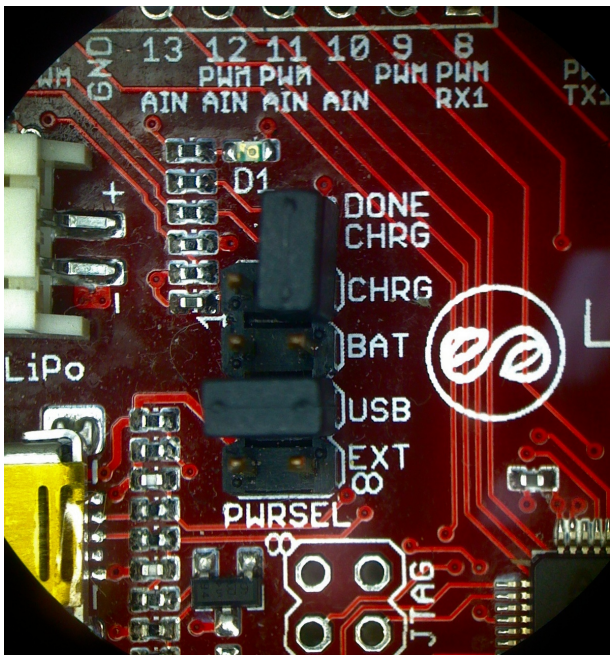

Figure 2: Proper jumper configuration: USB power selected and battery charger jumper attached but "open"

This half-attachment is a bit controversial.. the connection could be closed with no real problem except that charging LEDs have undefined state and could be confusing to the user. The jumper could also be floating in the static bag, but then it could be lost or forgotten.

Flash bootloader

The bootloader can be flashed either over JTAG or via the hardware serial bootloader interface. The JTAG header is not populated by default and using the serial bootloader tests that feature, so we usually use that.

You can get the stm32loader.py script from our maple-bootloader git repository at `http://github.com/leaflabs/maple-bootloader`; this script (written by a third party) requires the PySerial library as well as a python runtime installed.

The most recent version of the bootloader is from the 9c5f8e... git revision of the maple-bootloader reporisory. For safety we always use the same compiled binary file, when can be downloaded from `http://static.leaflabs.com/pub/leaflabs/maple-bootloader/maple_boot-rev3-9c5f8e.bin`. Using this file you should not have to checkout or recompile the bootloader; you will need to compile the interactive test session program (below).

On linux with an FTDI device as /dev/ttyUSB0, the command is usually something like:

```
stm32loader.py -p /dev/ttyUSB0 -evw maple_boot-rev3-9c5f8e.bin
```

The procedure is to connect both an FTDI chip and the Maple to a computer via USB; note that the Maple status light will not light up because there is no bootloader installed. Then connect the FTDI chip to the Serial1 USART device, which has TX1 on pin 7 and RX1 on pin 8; wiring Ground to the header between pins 13 and 14 makes the wiring easy:

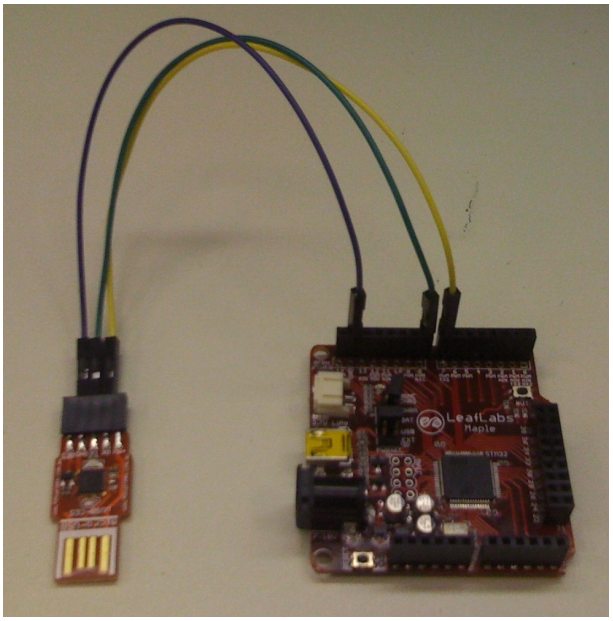

Figure 3: Serial1-to-FTDI wiring example for hardware serial bootloader uploading

Once wired up, you can enter hardware serial bootloader mode by holding the BUT button all the way down, pressing the Reset button for a second or two and release while keeping BUT down, then release BUT a couple seconds later. There will still be no sign of life from the Maple, so this can take a couple tries at first. Finally the above command can be run; the command output should show a long scroll of memory writes, and then an equally long set of verifications, then success. At this point you can reset the Maple and the blue status LED should blink perpetually (the board is in regular perpetual bootloader mode because there is no user program to run). If the board is not in hardware serial bootloader mode then the command will pause for several seconds before giving up; check the FTDI TX LED to make sure it is trying to write and that all the ports are configured correctly on the computer.

If the upload fails at any point after it has successfully started (eg, a verification failure) then that microcontroller is questionable and should NOT be shipped even if a second attempt is successful. If this happens with many boards then there may be a problem with the FTDI device, wiring, or USB voltage levels.

Upload test program

Next is to upload a recent version of the "interactive test session" program. This program can be connected to over SerialUSB and allows a number of tests to be run; it's helpful for the next testing steps and is also a nice thing to have preloaded on the board for users to play with.

We always recompile and upload this program from the 'master' branch of the libmaple git repository (link above) and then use the 'screen' program to interact over SerialUSB. The preferred version is that bundled with v0.0.6; you can checkout exactly this tag from the git repository. See `http://leaflabs.com/docs/libmaple/unix-toolchain/` for directions on getting the unix toolchain set up on Linux; then from the libmaple top level directory copy ./examples/test-session.cpp to ./main.cpp and 'make flash'. It should also be possible to use the "Example" version bundled with the IDE, but this is less convenient because the IDE recompiles the program every time. Either way, make sure the program is uploaded to FLASH, not RAM (it wouldn't fit in RAM anyways). Also make sure that the COMM variable in the source code is set to SerialUSB, not Serial2.

To upload the program run 'make install' or press the upload button with the Maple still plugged in and in perpetual bootloader mode. Then attach to the SerialUSB device with 'screen /dev/ttyACM0' or the serial monitor window in the IDE. You'll usually miss the pretty welcome banner graphics, but you can press spacebar or send 'h' to see some text thrown back to make sure that the connection is working.

Test buttons

"Sticky" buttons have been a huge QA issue. This is probably due to overheating of the buttons, bad button quality, and/or melted plastic inside the button. The real problem is that it can be hard to detect a partially-working button, but then very frustrating for the user down the line. It's possible that a button can work for the first few presses but then stop working. It's also common for the button to get electrically "stuck" in the pressed state even when it has noticeably disengaged mechanically. Some sticky buttons feel "mushy", while some feel just fine and still have intermittent issues. Vigilance is required at this step! It is also not unheard of for the button to completely *fall off the board*.

With the interactive test program running, press 'r' to enter GPIO readout mode; in this mode all pins are in INPUT_PULLDOWN[1] mode and any changes of state are printed out. Press the BUT button repeatedly and the program should report transitions between HIGH and LOW. Carefully watch the output while pressing and releasing the button a dozen times or so and make sure the state changes every time and that there isn't a significant delay between the physical act and the software update.

To test the RESET button, detach the serial monitoring program and press RESET repeatedly. The blue status LED should turn off and then turn back on and blink every time. For efficiency, this step could be performed a little earlier in the QA assembly line, but be sure both buttons get tested on every board!

Test pins with "Test Shield"

The test shield is a second Maple board which toggles every GPIO pin in a round robin manner and reports any problems. This modified board gets it's power from the board being tested and runs a passive program which responds to the '+' mode of the interactive test program.

The test shield is constructed by removing all the female headers from the top of a Maple and soldering on male connectors to the bottom. All of the GPIO pins plus the Ground and V_{in} pins must be connected; the only pins can NOT be connected are the power select header (which should still be on top and in EXT power mode), RESET, and the V_{cc} pins. It's probably easiest to solder male headers for all the female headers and then break off the V_{cc} and RESET pins with pliers. Here's a photo; this particular shield got a lot of use/abuse and one or two of the male pins have fallen off:

[1] *Note: older versions of the test program used INPUT or INPUT_FLOATING mode and then reported continuous state changes on random pins (especially when the board is touched). Make sure to get the latest version of the test program before uploading it!*

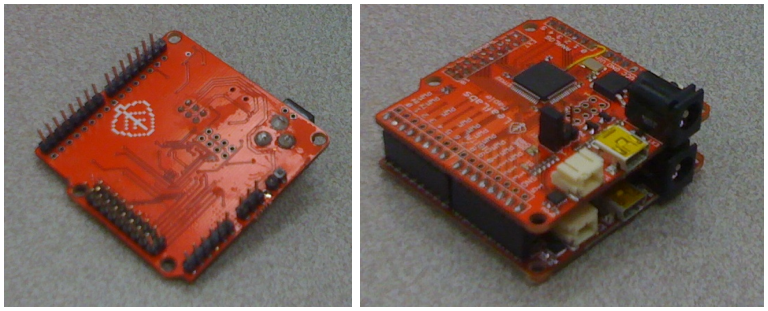

Figure 4: Test Shield underside and attached; note that this shield was constructed from a broken rev1 Maple that needed skywiring and has some missing male pins on the bottom.

Once the board is built and has the bootloader flashed, compile and upload the ./examples/qa-slave-shield.cpp program from the libmaple git repository (you'll put the power select jumper to USB for program upload, then back to EXT for use).

For each board being tested, press on the shield, which should power up and have the status LED blink. Enter the interactive test program and send '+'. It may take a couple seconds to load, but then the program should list all of the pins and a test status for each. If the program fails at any pin, or even pauses for a noticeable period before continuing, there's a problem with that pin.

The most frequent problem is a shorted or open solder connection at the headers or at the STM32 pads. Another problem is that attaching and removing the shield stresses the male header pins (though NOT the female pins)

The '+' program is not 100% perfect and may rarely report false failures; if you think it's not correct or needs tweaking please contact me. The program fails most frequently at pins 15 and 16. Also feel free to come up with your own mechanical design; perhaps thinner male headers or "pogo-pins" a la `http://www.sparkfun.com/commerce/tutorial_info.php?tutorials_id=138`.

Test battery features

The last step is to test the power select and battery charging circuits. The most common failure here is the battery charging status LEDs (surface mount, red and green). This involves moving the power select jumpers around, the following is the quickest method we came up with.

With the Maple powered by USB, close the battery charger jumper with no battery connected: the state of the circuit is undefined in this configuration, but after a second or two the green ("charged") LED usually lights up. The order that the red and green LEDs light up varies from board to board and is unimportant. Next attach a not-fully-charged battery: the red charging LED should light up. Leaving the battery plugged in, put the charging jumper back in the half-attached configuration, then move the power select jumper to BAT: the blue status LED should stop blinking when power is detached, then turn back on and start blinking again. If all is well, move the jumper back to USB and unplug the battery.

Per-board Preparation and Testing Checklist for LeafLabs Maple (rev4)

1. **Brief visual inspection of board** for missing components or silkscreen smudges
2. **Attach 2 jumpers to power select header:** USB (closed) and Battery Charge (hanging)
3. **Flash bootloader** via hardware Serial Bootloader function (attach USB for power, connect RX, TX, and GND to FTDI device, upload and verify with stm32loader.py, reset and check for blinky)
4. **Upload Test Program to FLASH via USB** using perpetual bootloader mode
5. **Test Reset and BUT Buttons with 10 pushes each** by watching the status LED for Reset and using the interactive test program 'r' mode for BUT
6. **Run the GPIO test program** with the test shield attached and the board being tested running the interactive test session in '+' mode
7. **Test Battery power circuit and charging LEDs** by setting the charger jumper with no battery (green LED flickers), attaching a half-charged battery (red LED), insetting charger jumper (no LEDs), moving power select to Battery (blue status blinks), then moving power select back to USB.
8. **Put board in a static bag** and update any database/spreadsheet. It's ready to ship!