VISORS Propulsion Bootloader Testing 2

Author: Toby Bell, Space Rendezvous Laboratory
Revisions by: Jonathan Vollrath, Space Systems Design Lab
Runner: Jonathan Vollrath, Space Systems Design Lab
Assistants: Ethan Traub & Joseph Gelin, Space Systems Design Lab
Date: 2024-09-18

Contents

Source (Google Doc)
https://docs.google.com/document/d/11FbZCOdbySJJznXFZ-sc717q-xzx7MsjINnGDBYd9Qk

VISORS Propulsion Bootloader Test 1

Author: Toby Bell, Space Rendezvous Laboratory
Runner: Jonathan Vollrath, Space Systems Design Lab
Assistant: Ethan Traub, Space Systems Design Lab
Reviewed: Ethan Traub, Space Systems Design Lab
Date: 2024-09-18
Time: 3:41 PM ET
Duration: TODO

1. Pull CGTC repo `visors-bootloader` branch

1.1. On the `visors-bootloader` branch, run `git pull`

- [x] Record commit hash: abdb8c8fedec418162ba70d741192be794cab2b7
- [x] **Notes:** none

2. Build the updated bootloader

2.1. Run `sh build.sh`

- [x] **Notes:** Size 788

3. Flash the updated bootloader

3.1. Run
`avrdude -C avrdude.conf -v -p atmega128 -c stk500v2 -P usb -U flash:w:boot.hex:i`

- [x] **Notes:** In C:\Users\Jonathan Vollrath\Downloads\avrdude-v7.3-windows-x64, ran

  .\avrdude -C "C:\Users\Jonathan Vollrath\Downloads\avrdude-v7.3-windows-x64\avrdude.conf" -v -patmega128 -cstk500v2 -Pusb -Uflash:w:"C:\Users\Jonathan Vollrath\OneDrive - Georgia Institute of Technology\Desktop\VISORS\Prop Software\cgtc-software\visors-bootloader\boot.hex":i -B 500khz

4. Start telemetry logging on the testbed

4.1. Reconnect the prop board to the testbed

- [x] ~~Done~~
- [ ] **Notes:** Flashed still connected to testbed

4.2. Start telemetry logging to file in COSMOS

- [x] Record filename: 2024_09_18_16_37_54_tlm.bin
- [x] **Notes:** none

4.3. Start prop bootloader decode: run
`tail -f [cosmos-telemetry-file] | python3 visors-prop-boot-decode.py`

- [x] Got START telemetry every 2 seconds

- ☑ ~~Got RESET telemetry every 2 seconds~~ **NO**
- ☑ **Notes:** Didn't get RESET every 2 seconds, see first test. Note that testbed computer doesn't have python on path variables, so actual command run was:

  tail -f "C:\Users\jvollrath6\Downloads\visors-cosmos\outputs\logs\xb1_visors\2024_09_18_16_37_54_tlm.bin" | "C:\Users\jvollrath6\AppData\Local\Programs\Python\Python312\python" visors-propdecode.py

4.4. Leave this process running during subsequent steps

- ☑ OK

5. Build the CGTC application software

5.1. Build the CGTC application in Arduino IDE and get the output **.hex** filename

- ☑ Record filename: cgtc-current.ino.hex
- ☑ **Notes:** Following command will assume app hex is in the visors-bootloader folder in the cgtc-software repo. For naming convention and cleanliness, ensure removal of old hex files

6. Generate page-write commands for upload

6.1. In the `visors-bootloader` directory, run
`python3 visors-prop-boot-encode.py` **`[hex-file]`** `pw`

- ☑ Commands alternate PAGE_WRITE_LO (0x30) and PAGE_WRITE_HI (0x31)
- ☑ Output looks reasonable to run in COSMOS
- ☑ **Notes:** Piped output of command into commands.txt:

  "C:\Users\jvollrath6\AppData\Local\Programs\Python\Python312\python.exe" visors-prop-boot-encode.py cgtc-current.ino.hex pw > commands.txt

7. Run page-write commands in COSMOS

7.1. **Disable bootloader timeout:** payload-write `0x02, 0x31, 0x34`, check telemetry

- ☑ Got TIMEOUT_DISABLE telemetry
- ☑ No longer getting START or TIMEOUT telemetry
- ☑ **Notes:** none

7.2. Run the page-write commands output from script in COSMOS, with a delay of 0.3 seconds in between them, check telemetry parser process

- ☑ Got PAGE_WRITE_LO and PAGE_WRITE_HI telemetry
- ☑ Did not get any PAGE_WRITE_MISMATCH telemetry
- ☑ Got PAGE_WRITE_HI telemetry for all pages
- ☑ **Notes:** FUCK YEAH

7.3. If any pages did not have a PAGE_WRITE_HI telemetry, re-run commands for only those pages.

- ☑ Record number of re-uploaded pages: 0
- ☑ **Notes:** none

7.4. Save any scripts/files created when performing the previous step to the visors-cosmos repo

- ☑ Done
- ☑ **Notes:** Script ota flasher saved as prop_ota_flash.rb in visors-cosmos > procedures

8. Read back page 0

8.1. **Read back page 0:** payload-write `0x02, 0x31, 0x32, 0x00`, check telemetry

- ☑ Got PAGE_READ telemetry for page 0
- ☑ Bytes match the first page of the `cgtc-current.ino` application hex file
- ☑ **Notes:** Hex file has 4 bytes in front a 1 byte appended that aren't included, so need to be included. Each page has 512 bytes.

9. Start the application

9.1. **Exit bootloader:** payload-write `0x02, 0x31, 0x33`, check telemetry

- ☑ Got EXIT telemetry
- ☑ Got prop application telemetry and heartbeat
- ☑ Prop board accepts and responds to commands
- ☑ **Notes:** FUCK YEAH

10. Update the application

10.1. Modify the application (change LED heartbeat frequency)

- ☑ Done

10.2. Compile the modified application in the Arduino IDE

- ☑ Done

10.3. Power-cycle the prop board and load the updated application via bootloader commands within 2 seconds

- ☑ Did not send TIMEOUT_DISABLE command
- ☑ Got PAGE_WRITE telemetry for all pages
- ☑ Got EXIT telemetry
- ☑ Prop application with modified heartbeat frequency
- ☑ Prop application receives and responds to commands
- ☑ **Notes:** FUCK YEAH

SUMMARY

Testing went exactly as planned. Several cleanup changes were made between testing 1 and testing 2, but they did not affect functionality of the bootloader.

1. Bootloader is capable of successfully flashing a new cgtc app over RS-422

2. The newly flashed cgtc application is capable of sending telemetry and receiving commands
3. As in the last test, no TIMEOUT telemetry on watchdog reset: We never received any TIMEOUT telemetry on watchdog reset. It may be because these messages are emitted too close to when the chip reboots, and a power effect ends up interfering with the signaling. Behaviorally, the timeout was still verified to be correct, we just didn't get the telemetry messages confirming it happened. This is a minor issue, as that telemetry was only meant to be used for debugging. It does not interfere with using the bootloader.
