VISORS Propulsion Bootloader Testing

Author: Toby Bell
Runner: Jonathan Vollrath
Date: 2024-09-16

Contents

Source (Google Doc)
https://docs.google.com/document/d/1M_s6kFc6XvByFzOGorf_yB_AZIN2VWaidDoJbEy3Ado

VISORS Propulsion Bootloader Test 1

Author: Toby Bell
Runner: Jonathan Vollrath
Date: 2024-09-16
Time: 9:08 AM ET
Duration: 3 h 51 min

1. Pull CGTC repo `visors-bootloader` pull request

1.1. Run `git fetch origin pull/6/head:visors-bootloader`

- ☑ Done
- ☑ **Notes:** *none*

1.2. Run `git checkout visors-bootloader`

- ☑ Done
- ☑ **Notes:** *none*

2. Build bootloader hex file

2.1. Run `make` (may need to adapt for Windows)

- ☑ Done
- ☑ **Notes:** *Adapted for Jonathan's computer; committed to git repo*

3. Check boot fuses using ISP

3.1. Connect prop board to AVR ISP MkII

- ☑ Done
- ☑ **Notes:** *none*

3.2. **Read lock bits:** run
`avrdude -C avrdude.conf -v -p atmega128 -c stk500v2 -P usb -U lock:r:-:h`

- ☑ Record output: 0xFF
- ☑ Record differences from expected value 0xFF: none

3.3. **Read low fuse bits:** run
`avrdude -C avrdude.conf -v -p atmega128 -c stk500v2 -P usb -U lfuse:r:-:h`

- ☑ Record output: 0xBF
- ☑ Record differences from expected value 0xBF: none

3.4. **Read high fuse bits:** run
`avrdude -C avrdude.conf -v -p atmega128 -c stk500v2 -P usb -U hfuse:r:-:h`

- ☑ Record output: 0xC6
- ☑ Record differences from expected value 0xC6: none

3.5. **Read extended fuse bits:** run
`avrdude -C avrdude.conf -v -p atmega128 -c stk500v2 -P usb -U efuse:r:-:h`

- ☑ Record output: 0xFF
- ☑ Record differences from expected value 0xFF: none

4. Write bootloader to flash using ISP

4.1. **Write flash:** run
`avrdude -C avrdude.conf -v -p atmega128 -c stk500v2 -P usb -U flash:w:boot.hex:i`

- ☑ **Notes:** *Wrote and read 712*

5. Write boot fuses using ISP

5.1. **Compute new high fuse bits:** clear bits 0 and 1, set bit 2
`[output from 3.4] & 0xfc | 0x04`

- ☑ Result: 0xC4

5.2. **Write new high fuse bits:** run
`avrdude -C avrdude.conf -v -p atmega128 -c stk500v2 -P usb -U hfuse:w:`**`[result from 5.1]`**`:m`

- ☑ Done
- ☑ **Notes:** *none*

6. Setup testbed

6.1. Connect prop board back to EDU

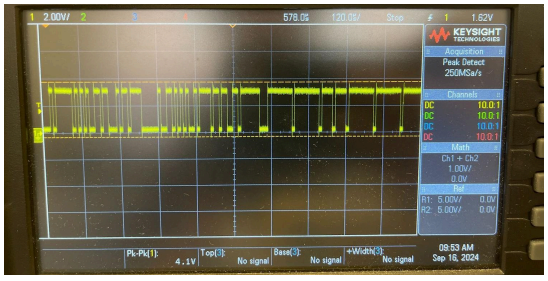
- ☑ ~~Done~~ **SKIPPED**
- ☑ **Notes:** *Never disconnected it in the first place*

6.2. Power on the testbed

- ☑ ~~Done~~ **SKIPPED**
- ☑ **Notes:** *It was already powered on*

6.3. Power on the prop board, check telemetry

- ☑ Got START telemetry
- ☐ Got TIMEOUT telemetry **NO**
- ☑ **Notes:** *Didn't get any telemetry at first, (possibly, also possible Jonathan is just blind) then started receiving telemetry. COSMOS not interpreting b/c it's not configured to do so. While debugging, connected oscilloscope and got the following trace:*



7. Check startup and timeout

7.1. Power-cycle prop board, check telemetry

- ☑ Got START telemetry
- ☐ Got TIMEOUT telemetry **NO**
- ☑ **Notes:** *received 0x80 (START) opcode*

7.2. Power-cycle prop board and send TIMEOUT_DISABLE (`0x02, 0x31, 0x43`) within 2 seconds

- ☑ Got TIMEOUT_DISABLE telemetry
- ☑ Stopped getting TIMEOUT and START telemetry
- ☑ **Notes:** *none*

8. Read page 0 (baseline)

8.1. **Read page 0 lower half:** payload-write `0x02, 0x31, 0x40, 0x00, 0x00`, check telemetry

- ☑ Got PAGE_READ_LO telemetry (0x83)
- ☑ Record contents:
  ```
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffff
  ```
- ☑ **Notes:** *none*

8.2. **Read page 0 upper half:** payload-write 0x02, 0x31, 0x41, check telemetry

- ☑ Got PAGE_READ_HI telemetry (0x84)
- ☑ Record contents:
  ```
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffff
  ```
- ☑ **Notes:** *none*

9. Read page 10 (baseline)

9.1. **Read page 10 lower half:** payload-write `0x02, 0x31, 0x40, 0x0a, 0x00`, check telemetry

- ☑ Got PAGE_READ_LO telemetry (0x83)
- ☑ Record contents:
  ```
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  ffffffffffffffffffffffffffffffffffff
  ```
- ☑ **Notes:** *none*

9.2. **Read page 10 upper half:** payload-write 0x02, 0x31, 0x41, check telemetry

- ☑ Got PAGE_READ_HI telemetry (0x84)

- [x] Record contents:
```
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffff
```
- [x] **Notes:** *none*

10. Write to page 0

10.1. Write page 0 lower half: payload-write (LENGTH 131), check telemetry
```
0x02, 0x31, 0xf0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7,
0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2,
0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5,
0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8,
0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3,
0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6,
0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8
```
- [x] Got PAGE_WRITE_LO telemetry (0x81)
- [x] Record response: 0B 32 C0 00 00 08 00 00 00 00 00 00 81 E1 E5
- [x] **Notes:** *none*

10.2. **Write page 0 upper half:** payload-write (LENGTH **133**), check telemetry
```
0x02, 0x31, 0xf1, 0x0, 0x0, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd,
0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe,
0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff,
0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8,
0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9,
0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa,
0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb,
0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc,
0xfb, 0xfa, 0xf9, 0xf8, 0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8
```
- [x] Got PAGE_WRITE_HI telemetry (0x82)
- [x] Record response: 0B 32 C0 00 00 08 00 00 00 00 00 00 82 D1 86
- [x] **Notes:** *none*

11. Write to page 10

11.1. Write page 10 lower half: payload-write (LENGTH 131), check telemetry
```
0x02, 0x31, 0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4,
0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3,
0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2,
0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1,
0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8,
0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7,
0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6,
0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5,
0xf6, 0xf7, 0xf8, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8
```
- [x] Got PAGE_WRITE_LO telemetry (0x81)
- [x] Record response: 0B 32 C0 00 00 08 00 00 00 00 00 00 81 E1 E5
- [x] **Notes:** *none*

11.2. **Write page 10 upper half:** payload-write (LENGTH **133**), check telemetry

```
0x02, 0x31, 0xf1, 0xa, 0x0, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb,
0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8,
0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd,
0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa,
0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf,
0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc,
0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9, 0x8, 0xf, 0xe, 0xd, 0xc, 0xb, 0xa, 0x9,
0x8
```

- ☑ Got PAGE_WRITE_HI telemetry (0x82)
- ☑ Record response: 0B 32 C0 00 00 08 00 00 00 00 00 00 82 D1 86
- ☑ **Notes:** *none*

ADJOURNED

We adjourned at this point, since Jonathan had to go for a class. We did not complete the remainder of the testing in this procedure, since Toby made some changes after this point in preparation for a second round of testing in the afternoon.

SUMMARY

Overall, testing went very smoothly. Almost all requirements/expectations were met, and most steps worked on the first try. The only significant deviations were:

- We never completed the procedure, because Jonathan had to go for a class, and Toby elected to make changes to the software during midday based on feedback from the first round of testing.
- 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.

12. Read-back page 0 after writing

12.1. **Read page 0 lower half:** payload-write (LEN 5) `0x02, 0x31, 0x40, 0x00, 0x00`, check telemetry

- ☐ Got PAGE_READ_LO telemetry (0x83)
- ☐ Record contents:
- ☐ Contents start with 01 02 03 04 05 06 07 08 01 02 03…
- ☐ Contents differ from baseline

12.2. **Read page 0 upper half:** payload-write (LEN 3) `0x02, 0x31, 0x41`, check telemetry

- ☐ Got PAGE_READ_HI telemetry (0x84)
- ☐ Record contents:
- ☐ Contents start with FF FE FD FC FB FA F9 F8 FF FE FD…
- ☐ Contents different from baseline

13. Read-back page 10 after writing

13.1. **Read page 10 lower half:** payload-write (LEN 5) `0x02, 0x31, 0x40, 0x0a, 0x00`, check telemetry

- ☐ Got PAGE_READ_LO telemetry (0x83)

- [ ] Record contents:
- [ ] Contents start with F1 F2 F3 F4 F5 F6 F7 F8 F1 F2 F3…
- [ ] Contents differ from baseline

13.2. Read page 10 upper half: payload-write (LEN 3) `0x02, 0x31, 0x41`, check telemetry

- [ ] Got PAGE_READ_HI telemetry (0x84)
- [ ] Record contents:
- [ ] Contents start with 0F 0E 0D 0C 0B 0A 09 08 0F 0E 0D…
- [ ] Contents different from baseline

VISORS Propulsion Bootloader Test 2

Author: Toby Bell
Runner: Jonathan Vollrath
Date: 2024-09-16
Time: 7:24 PM ET
Duration: 3 h 56 min

NOTE Toby changed command and telemetry opcodes since test 1 above.
NOTE Toby removed outputting the CCSDS secondary header from the bootloader telemetry.

1. Pull CGTC repo `visors-bootloader` branch

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

- ☑ Record commit hash: dc6721f
- ☑ **Notes:** *Cherry-picked build.sh commit on top of this, resulting in 10055a6*

2. Build the updated bootloader

2.1. Run `sh build.sh`

- ☑ **Notes:** *Size 790 bytes*

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`

- ☑ **Notes:** *We did not get prop telemetry at first, because Toby had removed the secondary header from the booarloader telemetry but forgot to clear the "has secondary header" bit in the primary header. We modified the bootloader to remove the "has secondary header" bit and reran step 2.*

4. Start telemetry logging on the testbed

4.1. Reconnect the prop board to the testbed

- ☑ Done
- ☑ **Notes:** *none*

4.2. Start telemetry logging to file in COSMOS

- ☑ Record filename: C:/Users/jvollrath6/Downloads/visors-cosmos/outputs/logs/xb1_visors/2024_09_16_19_57_15_tlm.bin
- ☑ **Notes:** *none*

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

- ☑ Got START telemetry every 2 seconds

- ☐ Got RESET telemetry every 2 seconds **NO**
- ☑ **Notes:** *Changed decode script to look for the new CCSDS header pattern since removing the secondary header bit*

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:** *none*

6. Generate page-write commands for upload

6.1. In the `bootloader` directory, run
`python3 visors-prop-boot-util.py` **`[hex-file]`** `--wrap=pw --format=cosmos`

- ☑ Commands alternate PAGE_WRITE_LO (0x30) and PAGE_WRITE_HI (0x31)
- ☑ Output looks reasonable to run in COSMOS
- ☑ **Notes:** *Had to delete last three commands from output because Toby forgot to remove those after debugging his script*

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 0.1 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:** *We ultimately ran with a delay of 0.3 seconds instead of 0.1 seconds. We never tried 0.1 seconds, however we chose 0.3 seconds because BCT had a note in their ICD saying that the payload-write command had a rate limit of 5 Hz.*
  We did originally try with no delay at all, and this caused many BAD_CRC, WRITE_MISMATCH, and missing PAGE_WRITE messages.

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 to the COSMOS repo

- ☐ ~~Done~~ **WON'T DO**
- ☑ **Notes:** *We didn't create or save any files. We did copy/paste the generated commands into the COSMOS script editor and run them from there, but it was an ad-hoc script containing the specific data for our upload, and not worth saving.*

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:** *We had a big issue with this for a little while. It seemed like some of the bytes were not written correctly to flash, resulting in a corrupted application. Ultimately, we noticed that it was consistently bytes 1 and 129 of each page, and more so that they were always off by 2 (the bytes written to flash were 2 greater than they should have been). This was curious, since bytes 1 and 129 are both at the same offset (offset 5) in the PAGE_WRITE_LO and PAGE_WRITE_HI commands respectively. We spent a little while looking for bugs in the bootloader, but couldn't find the problem. Eventually, we realized that the bug was not in the bootloader, but in what BCT was sending to prop—it turns out BTC was thinking that byte 5 in the command was the last byte of a CCSDS header, so they added two to it for each packet when they appended their CRC. For now, we are going to work around this issue by simply subtracting two from byte 5 in each command we ask BCT to send to prop. Doing that ultimately achieved the same result.*

9. Start the application

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

- ☑ Got EXIT telemetry
- ☑ Got prop application telemetry and heartbeat
- ☑ **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
- ☑ **Notes:** *FUCK YEAH*

SUMMARY

Overall, testing went smoothly. Almost all requirements/expectations were met. The only two significant deviations were:

1. 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.
2. We had to work around the fact that BCT always adds `2` to byte 5 of the payload, which for the bootloader is inside the application payload range, resulting in corrupted software uploads. We solved this by simply decrementing byte 5 of all commands by 2 to counteract the effect of the BCT bus. This workaround was implemented and documented in ground software (the Python script that generates the bootloader commands), and did not require any modification to the bootloader itself.
