Rev1 Board Communication Firmware Development Report (2)

Gen2 Hardware Call

Bunheng Ty, Kael Hanson

WIPAC

09-14-2017

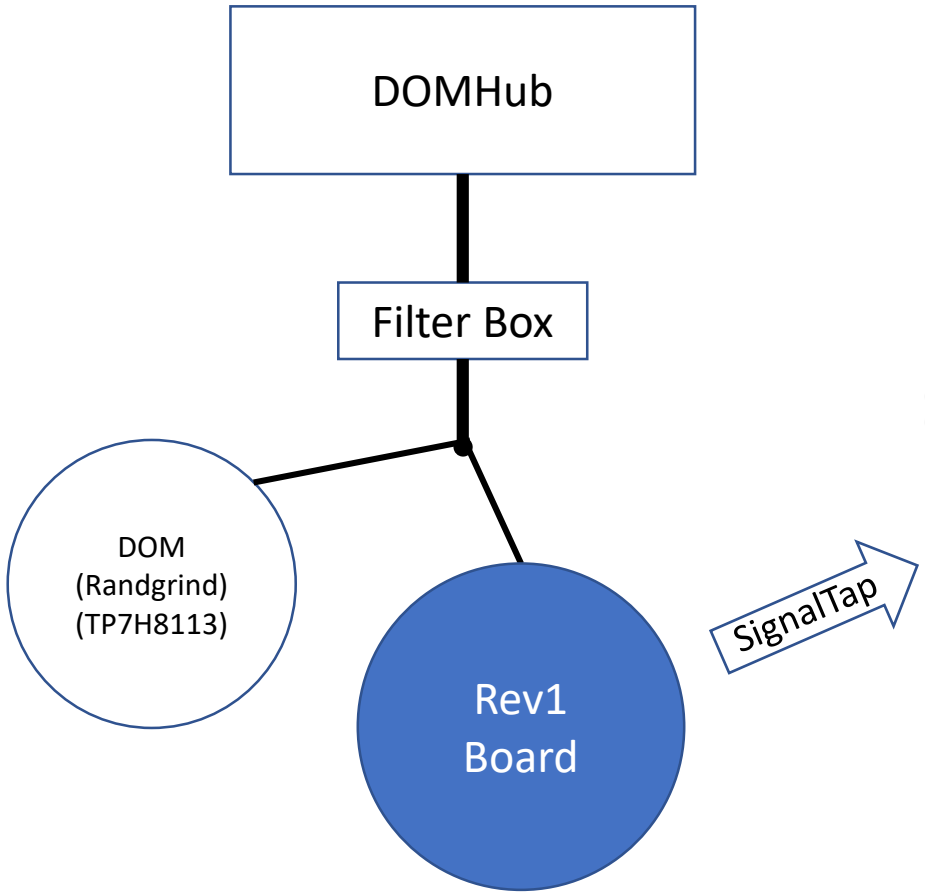
Overview

- The Rev1 COMM firmware can now reliably decode DOMHub-DOM communication. Furthermore, it can stream the decoded bits in real time to a computer via a 2 Mbaud UART port.
- This was accomplished with a new decoding method.
- Handshaking and some data exchange between DOMHub and Randgrind to be presented.

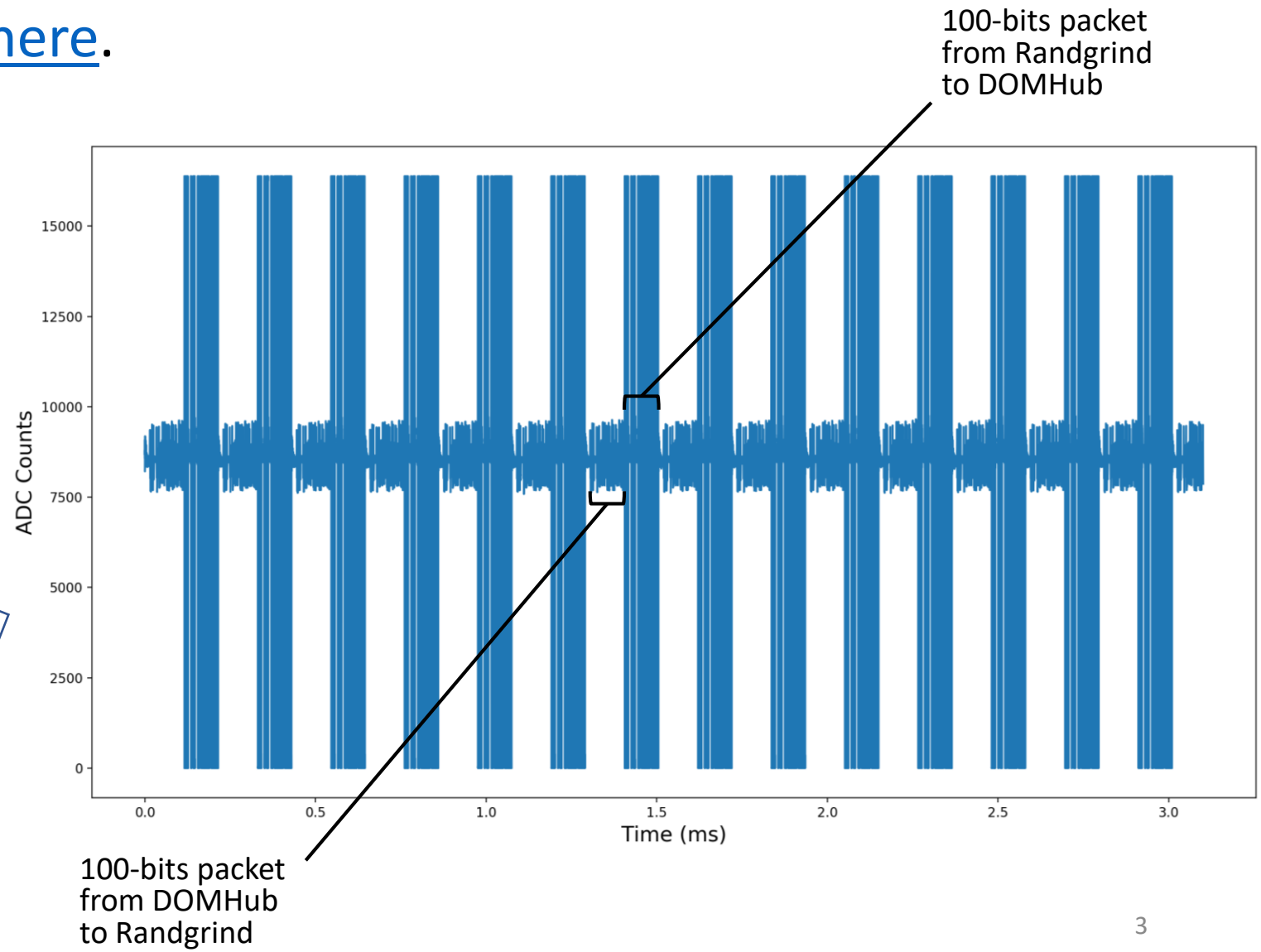
- Began work on the other half of the firmware—comm_transmit. Initial (not so successful) attempt to get the DOMHub to recognize the Rev1 board is presented at the end of this presentation.

Previously…

- Report(1) can be found here.

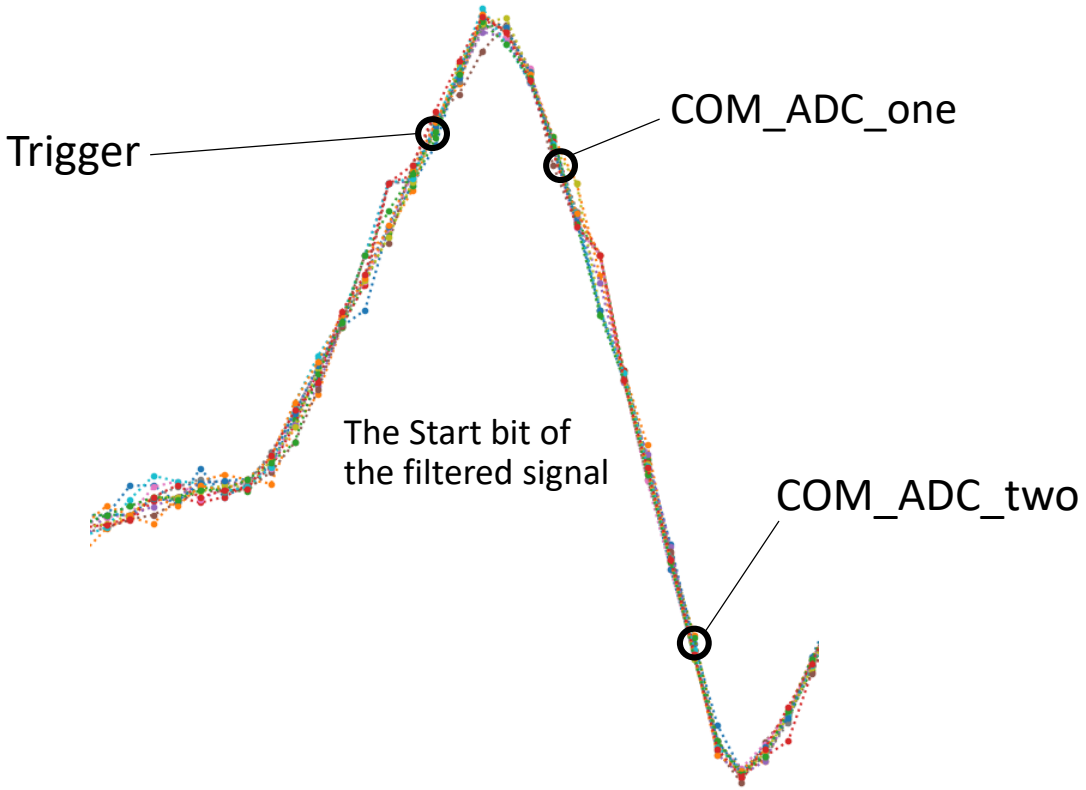


SignalTap



Previously…

- First draft of the bit decoder – decode the bit by looking at the slope of the falling edge.
- The Trigger point is used as the reference point for every bit in the packet: every 20 clock ticks from the trigger point is a start of a new bit.



>> If (COM_ADC_one – COM_ADC_two) > slope_threshold,

then new_bit <= 1

else new_bit <= 0

>> Count 20 clock ticks from the Trigger point to reach the start of the second bit in the packet

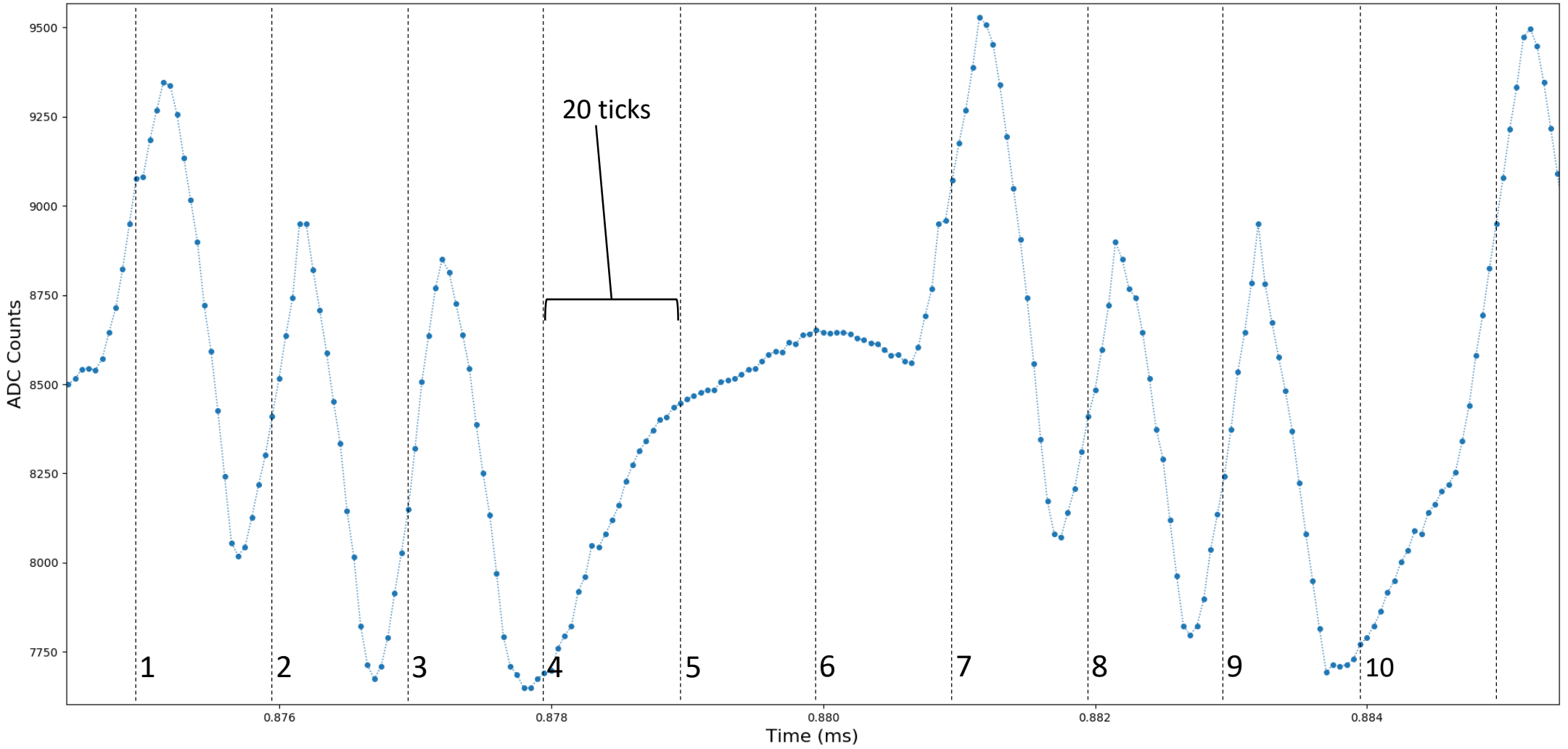
.

.

.

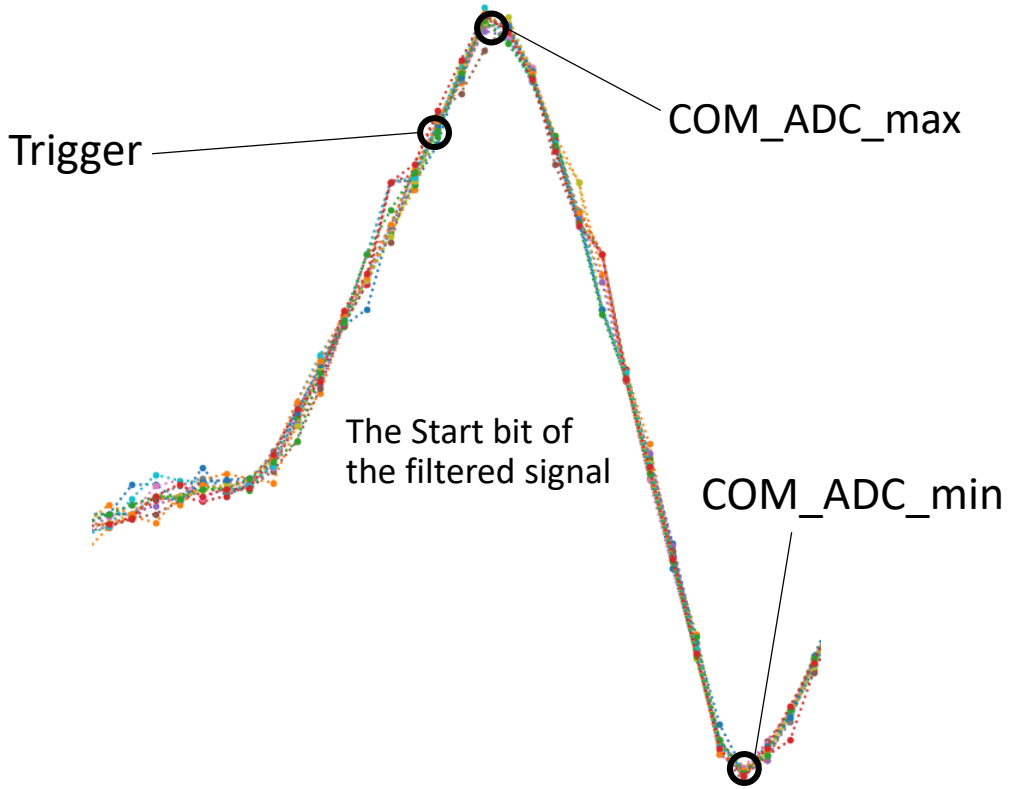
Previously…

- First ten bits – the Start byte (‘1’ + ‘xE3’ + ‘0’)



Now : A new bit decoder

- Two assumptions:
  1. A bit is always 1 µs long
  2. Noise is much smaller than the “1” signal
- The new bit decoder relies on both assumptions in order to work



- Each bit is made up of 20 samples. If the bit is a “1”, the first 10 samples must contain the max value, the remaining 10 contains the min.

>> If (COM_ADC_max – COM_ADC_min) > half of Start bit’s,

then new_bit <= 1

else new_bit <= 0

>> Count 20 clock ticks from the Trigger point to reach the start of the second bit in the packet

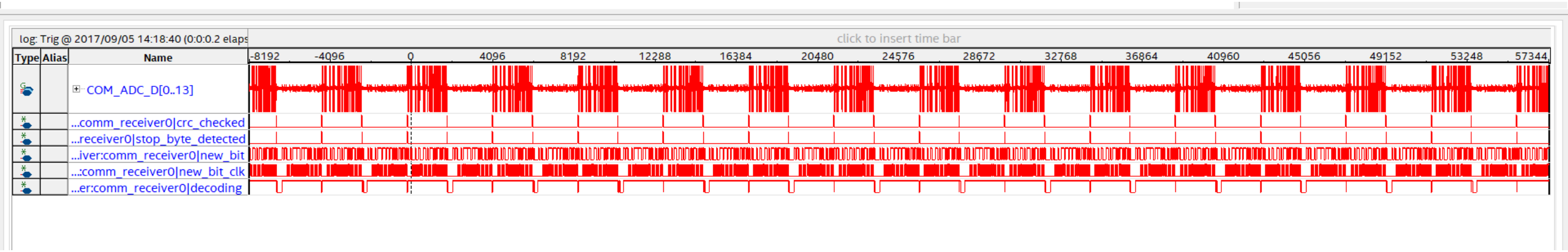
.

.

.

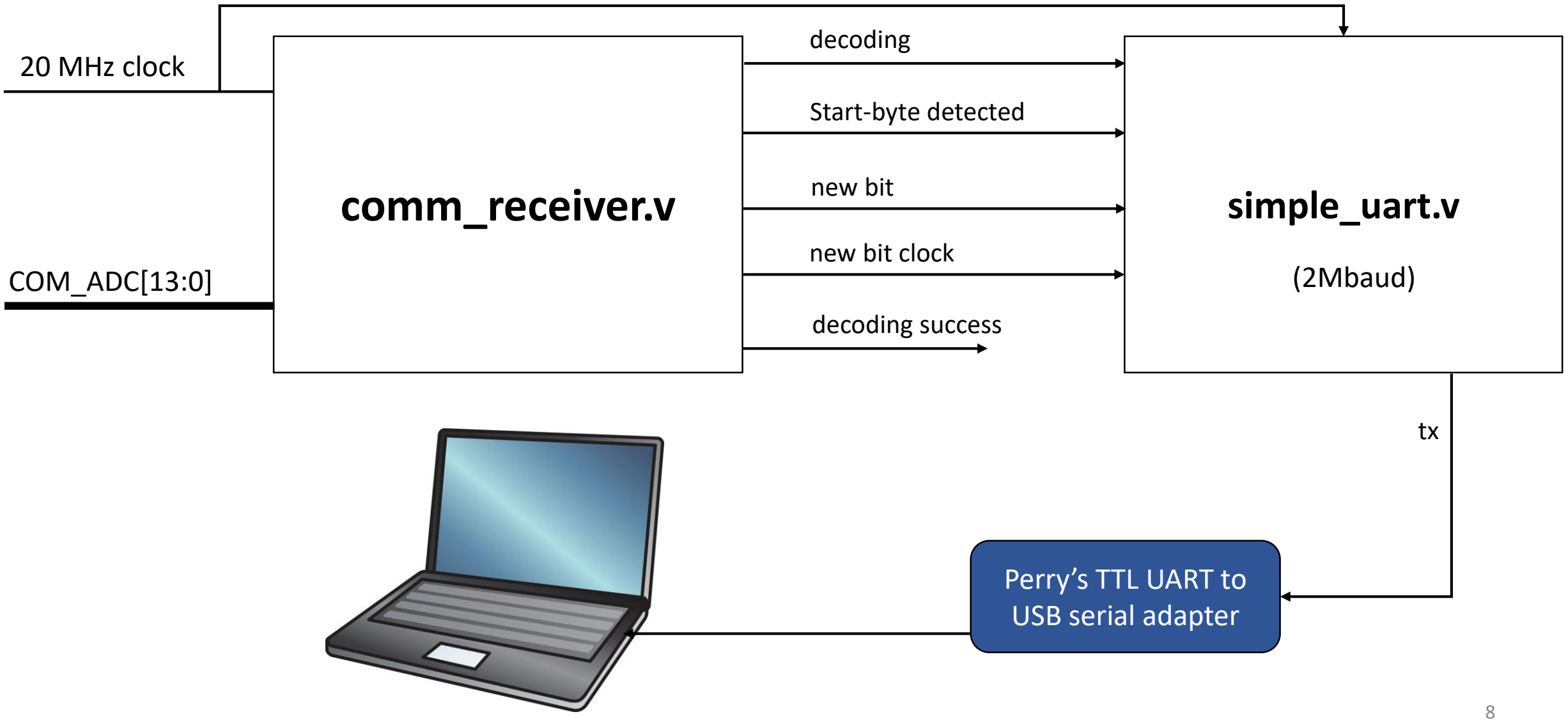
comm_receiver.v

- The new bit decoder now used in the comm_receiver Verilog module.
- Also implemented are: start-byte detection, stop-byte detection, and CRC32 check
- The module was successfully tested in hardware:



64K samples of COM_ADC and status registers in comm_receiver.v, captured in SignalTap II.

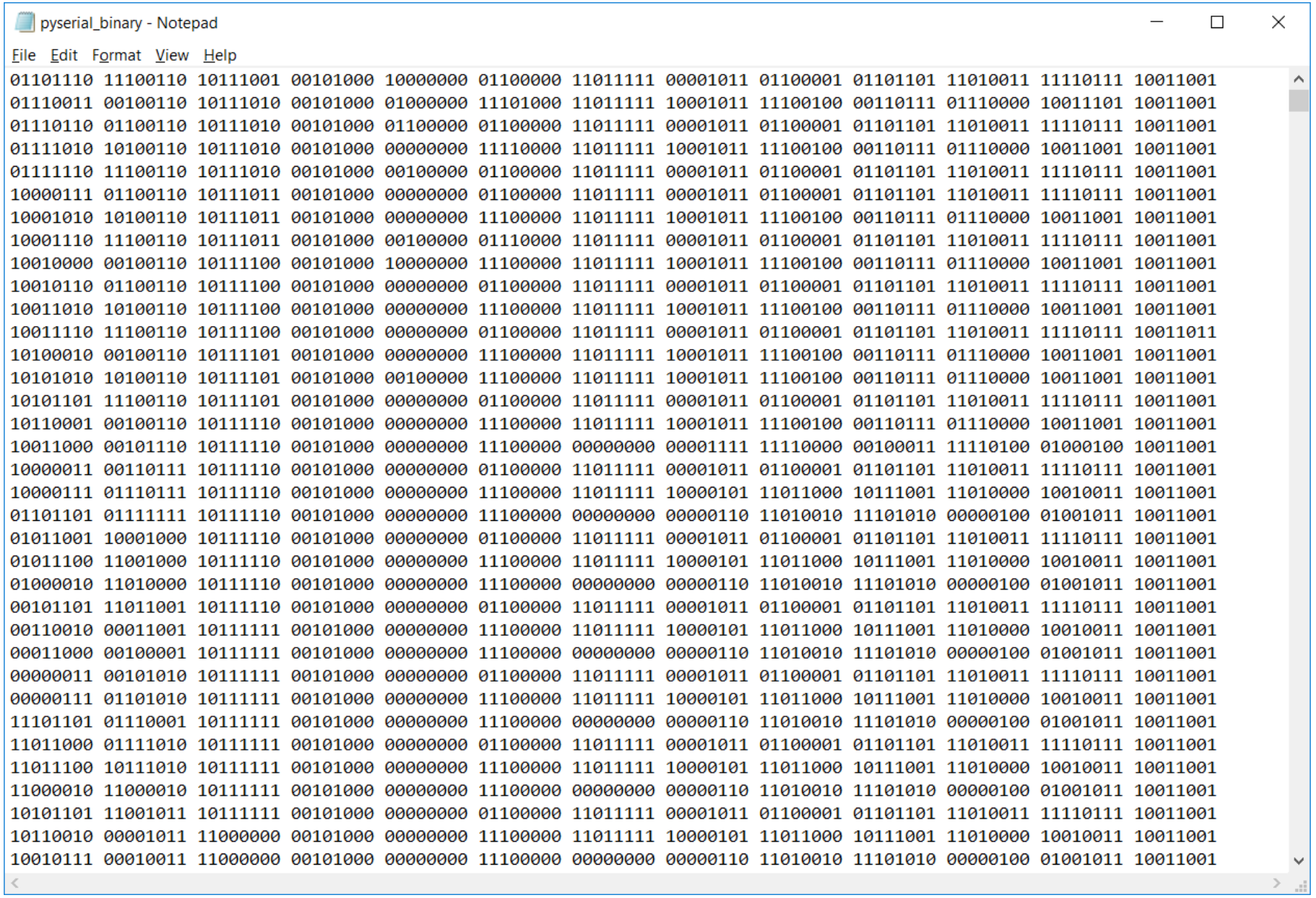
comm_receiver.v



DOMHub-DOM Conversation on Start Up

Raw Data:

*Each line represents a packet of data. The first four bytes give the timestamp of the start of the packet. The rest of the line is the actual data with the first ten bits and all paddings removed.



DOMHub-DOM Conversation during Initial Starting Up

- Used Karl-Heinz Sulanke's document to write a python script to interpret the binary data.

Command Prompt - python comm_decode.py

```
C:\Users\Bunheng\Desktop>python comm_decode.py
```

Timestamp(ms)	Msg #	CRC32	DOM	Packet Type	Data len	Sequence field	Boot State	Vol Req	DOR Control Message
190421.4016	1	Error	A	undefined	0	1000101111011111			
34161.95075	2	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34162.7701	3	Error	A	undefined	0	1000101111011111			
34163.5895	4	Error	B	DOR contr	128	0000101111011111	ConfigBoot	00	comm chan reset
34164.40895	5	Error	A	DOR contr	2112	1000101111011111	ConfigBoot	Up	comm chan reset
34165.2283	6	Error	B	DOR contr	96	0000101111011111	ConfigBoot	00	comm chan reset
34166.0477	7	Error	A	undefined	0	1000101111011111			
34166.8671	8	Error	B	DOR contr	32	0000101111011111	ConfigBoot	00	comm chan reset
34168.50595	9	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34169.3253	10	Checked	A	DOR contr	0	1000101111011111	ConfigBoot	Up	comm chan reset
34170.1447	11	Error	B	undefined	32	0000101111011111			
34170.964	12	Error	A	DOR contr	128	1000101111011111	ConfigBoot	Up	comm chan reset
34171.7835	13	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34172.6029	14	Checked	A	DOR contr	0	1000101111011111	ConfigBoot	Up	comm chan reset
34173.4223	15	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34174.2417	16	Checked	A	DOR contr	0	1000101111011111	ConfigBoot	Up	comm chan reset
34175.8805	17	Error	A	DOR contr	32	1000101111011111	ConfigBoot	Up	comm chan reset
34176.69985	18	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34177.51925	19	Checked	A	DOR contr	0	1000101111011111	ConfigBoot	Up	comm chan reset
34177.6204	20	Checked	A	DOR contr	0	0000111100000000	ConfigBoot	00	idle
34177.73455	21	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34178.55395	22	Checked	A	DOR contr	0	1000010111011111	ConfigBoot	Up	data read request
34178.65505	23	Checked	A	DOR contr	0	0000011000000000	ConfigBoot	00	data read ack, no data
34178.76925	24	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34179.5886	25	Checked	A	DOR contr	0	1000010111011111	ConfigBoot	Up	data read request
34179.6897	26	Checked	A	DOR contr	0	0000011000000000	ConfigBoot	00	data read ack, no data
34179.80385	27	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34180.6233	28	Checked	A	DOR contr	0	1000010111011111	ConfigBoot	Up	data read request
34180.7244	29	Checked	A	DOR contr	0	0000011000000000	ConfigBoot	00	data read ack, no data
34180.83855	30	Checked	B	DOR contr	0	0000101111011111	ConfigBoot	00	comm chan reset
34181.65795	31	Checked	A	DOR contr	0	1000010111011111	ConfigBoot	Up	data read request
34181.75905	32	Checked	A	DOR contr	0	0000011000000000	ConfigBoot	00	data read ack, no data

First response from Randgrind

DOMHub-DOM Conversation during Initial Starting Up

Time(ms)	Event
--	DOMHub keeps sending “communication channel reset” messages, with no response
0	Randgrind finishes booting up and responds to one of the “comm chan reset” with an “idle”
0.93355	DOR sends, for the first time, “data read request” to Randgrind
1.03465	Randgrind responds to above with “data read request acknowledged, no data”
	(for a while, a repeat of the two lines above)
18.6238	Randgrind responds with “initiate connection”
422.1399	Randgrind again responds with “initiate connection”.

*Most of the time, Randgrind responds to “data read request” with “data read request acknowledged, no data”.

*But every ~400 ms, it responds with “initiate connection”.

DOMHub-DOM Conversation during Initial Starting Up

Time(ms)	Event
--	DOMHub keeps sending “communication channel reset” messages, with no response
0	Randgrind finishes booting up and responds to one of the “comm chan reset” with an “idle”
0.93355	DOR sends, for the first time, “data read request” to Randgrind
1.03465	Randgrind responds to above with “data read request acknowledged, no data”
	(for a while, a repeat of the two lines above)
18.6238	Randgrind responds with “initiate connection”
422.1399	Randgrind again responds with “initiate connection”.

In summary, the DOMHub-DOM handshaking process is:

DOMHub sends “comm_chan_reset”

→ DOM responses with “idle”

DOMHub-DOM Conversation during a DOMTerm Session

* I then ran domterm from the DOMHub terminal. And that caused these messages:

Time(ms)	Event
7372.36525	DOMHub, for the first time, sends “connection initiated” to Randgrind
7372.4663	Randgrind responds to above with “message received, more Rx buffer”
	(the two lines above happens nine more times, but not back-to-back)
7381.72185	Randgrind responds to “data read request” with “connection initiated”
7383.87445	Randgrind responds to “data read request” with a data_end packet with a 20 byte payload. The content of the payload is: **\r\nconfigboot v2.71\r\n**
7384.2897	Randgrind responds to “data read request” with a data_end packet with a 17 byte payload. The content of the payload is: **type ? for help\r\n\x00\x00\x00**
7384.70495	Randgrind responds to “data read request” with a data_end packet with a 2 byte payload. The content of the payload is: **# \x00\x00**

DOMHub-DOM Conversation during a DOMTerm Session

Time(ms)	Event
7422.314	DOMHub sends “ack” (instead of the usual “data read request”). Sequence field reads: 0.
7422.41505	Randgrind responds to above with “message received, more Rx buffer”
7422.7445	DOMHub sends “ack” (instead of the usual “data read request”). Sequence field reads: 1.
7422.8455	Randgrind responds to above with “message received, more Rx buffer”
7423.1749	DOMHub sends “ack” (instead of the usual “data read request”). Sequence field reads: 2.
7423.27595	Randgrind responds to above with “message received, more Rx buffer”

Typing the character “?” on the keyboard then caused the following:

8737.10985	DOMHub sends a data_end packet with a 1-byte payload.
	But the actual content is: **?\x00f÷**
8737.25095	Randgrind responds to above with “message received, more Rx buffer”
8737.68145	Randgrind responds to a “data read request” with an ack packet

*Randgrind then proceeds to send a series of data_end packets containing various characters…

DOMHub-DOM Conversation during a DOMTerm Session

testdaq@labhub01:~

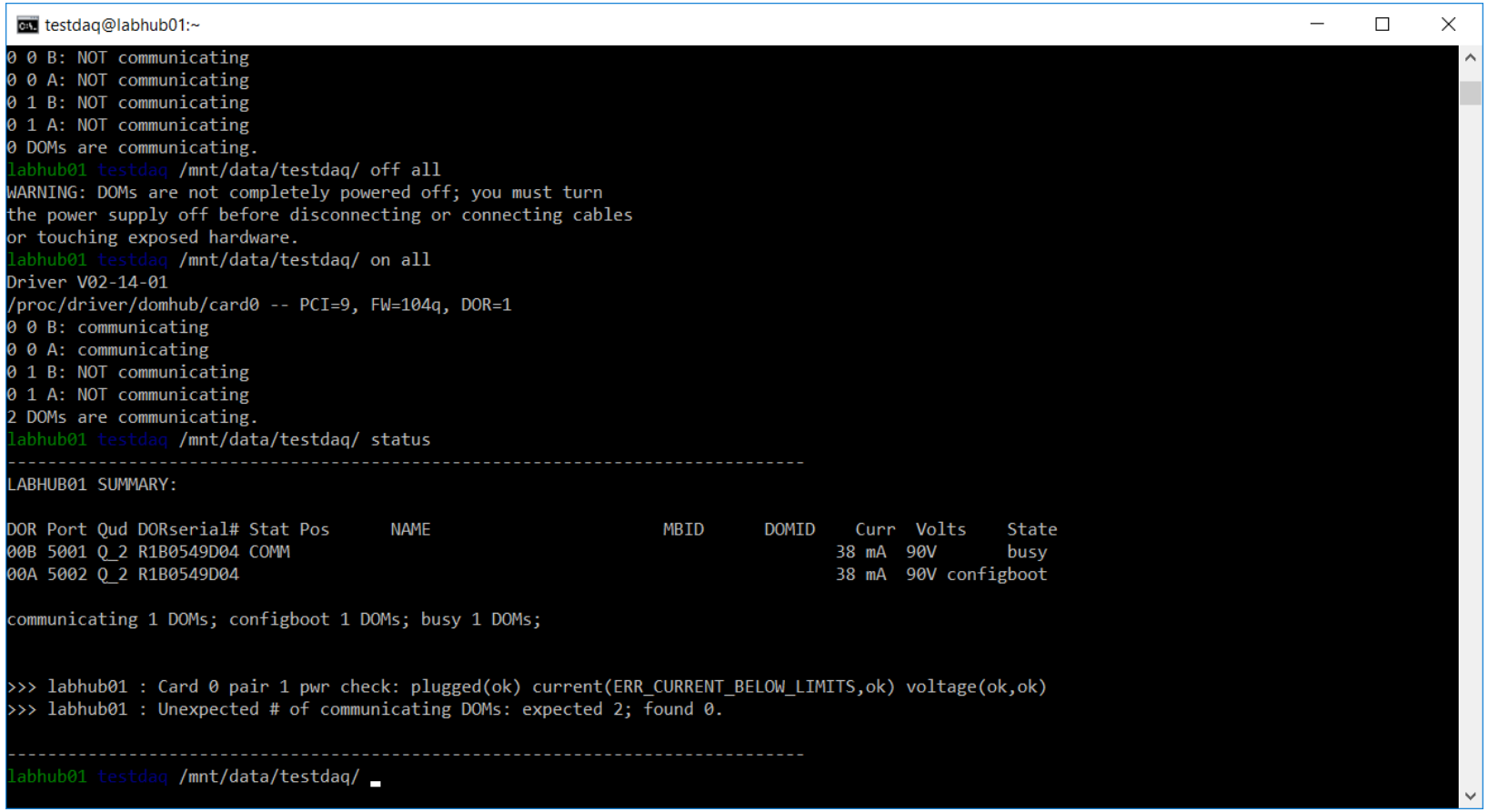
```
labhub01 testdaq /mnt/data/testdaq/ domterm 00a
connecting... OK

configboot v2.71
type ? for help
# ?
Commands:
  r               : reboot
  d               : display current parameters
  f               : set boot from flash mode
  s               : set boot from serial mode
  b               : set swap flash A and B mode
  a               : set canonical flash A and B mode
  p [schip echip] : program flash starting at schip (0 or 1) to echip (0 or 1)
  ?               : show commands
# d
Parameters:
  Boot from flash
  Don't swap flash A and B
  0 flash unlock errors
  0 flash lock errors
  0 flash erase errors
  0 flash write errors
  0 flash checksum errors
  0 flash parse errors
# 
```

domterm as seen on the DOMHub terminal

comm_transmitter

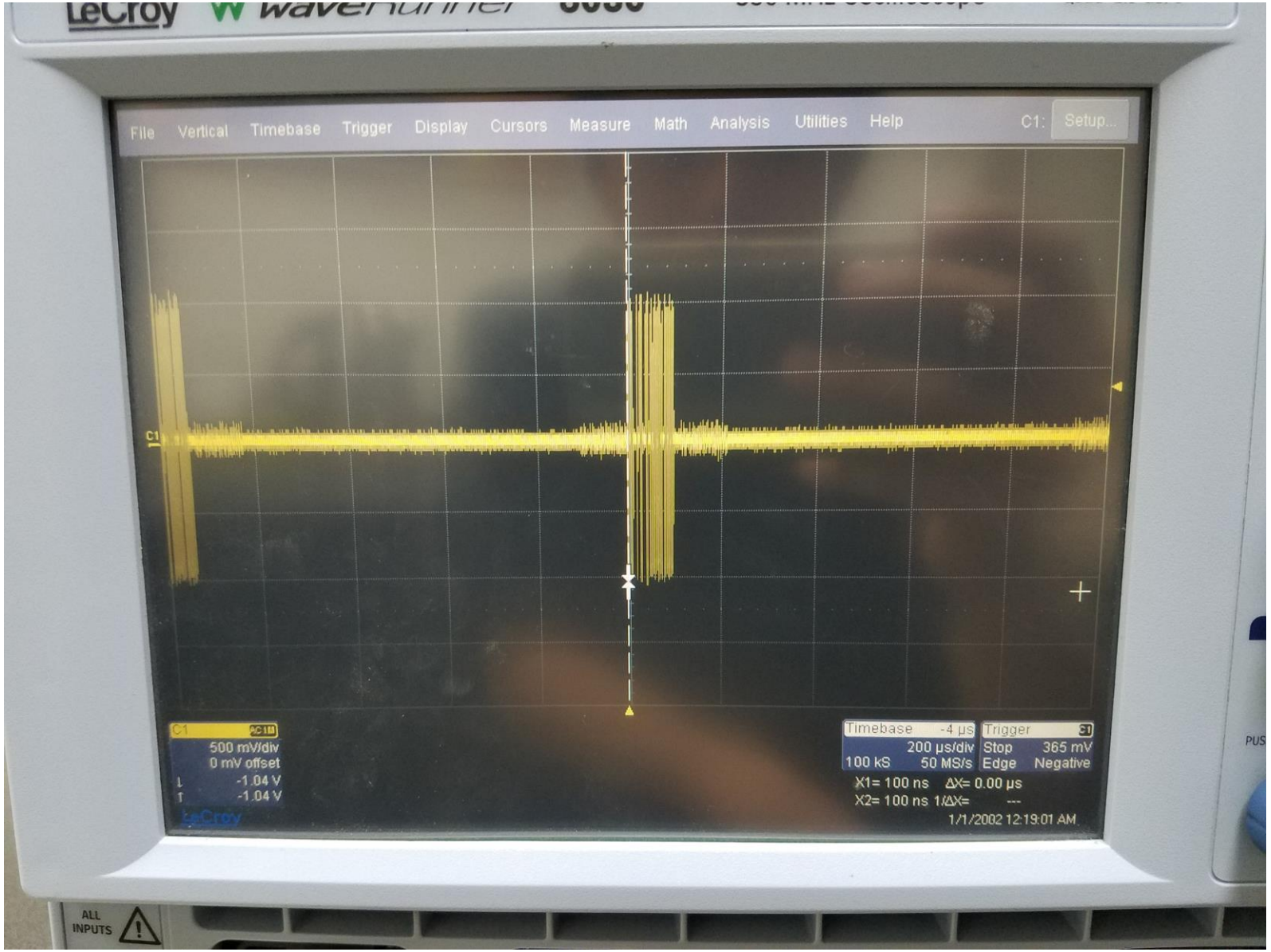
- Now working on comm_transmitter.v. Currently it sends prerecorded messages to the DOMHub. This should be enough to handle the initial handshaking process.



DOMHub seems to recognize the Rev1 board (B), but it gets stuck on the ‘status’ command.

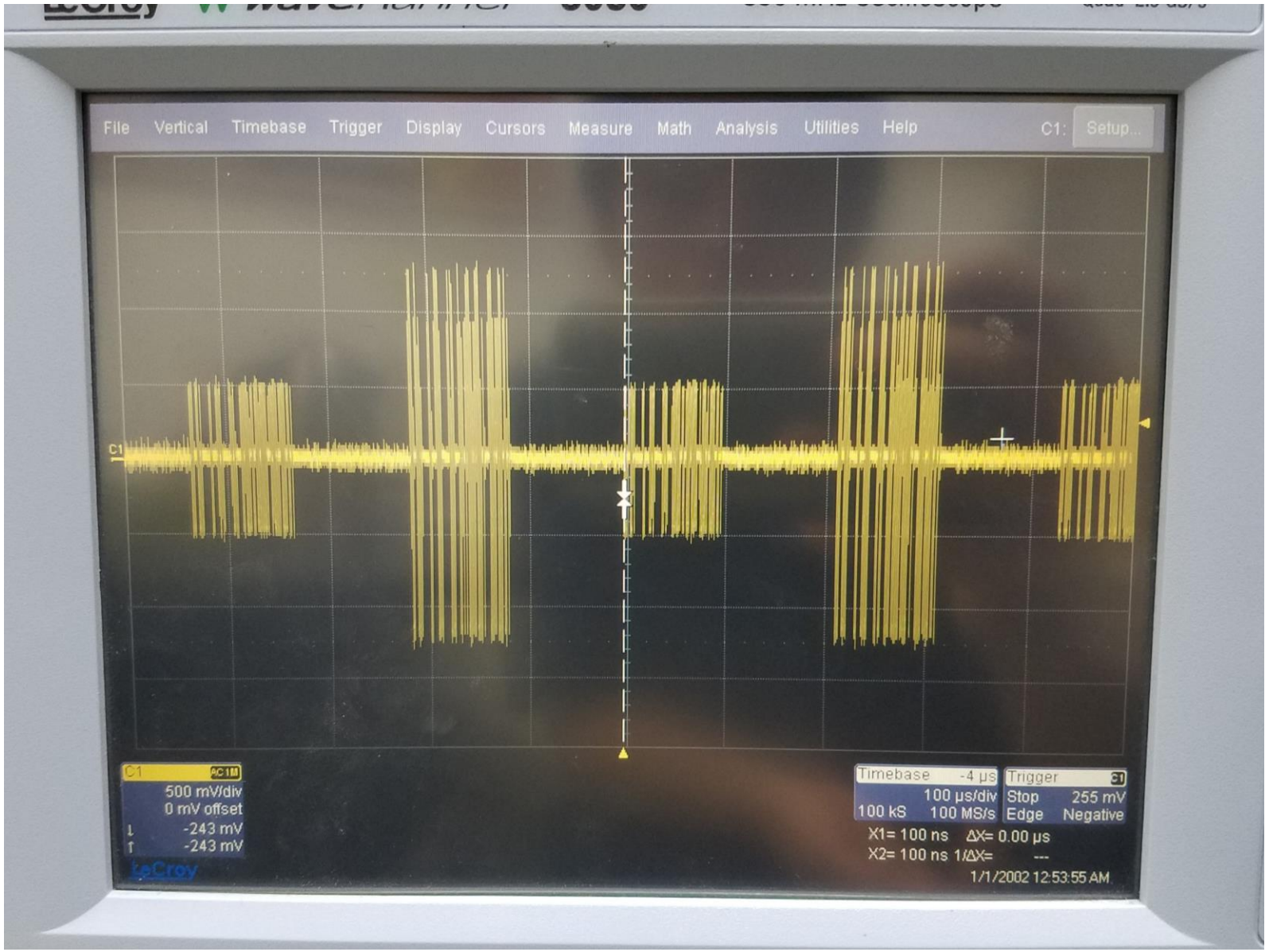
My guess is that the ‘status’ command causes the DOMHub to send extra packets to the Rev1 board, which at this point can not respond to them properly.

comm_transmitter



- DOMHub-Randgrind communication with the Rev1 board disconnected (physically).

comm_transmitter



- DOMHub-Randgrind + DOMHub-Rev1 communication
- Problems:
  - Rev1's signal too weak
  - Randgrind's signal got distorted (see next slide)

comm_transmitter

- Zoom-in picture of last slide.

- Randgrind's single gets distorted when Rev1 is hooked up on the same communication line (even when Rev1 is powered off).

Conclusions

- The receiver part of the COMM firmware now works in actual hardware, in real time. Additional capabilities will still need to be added: packet type recognition, automatic response, interfacing with Nios II, etc…
- Firmware is fast, software is slow. The DOM communication firmware solves the problem of slow software by having automatic responses made by the firmware, until the software is ready to give an actual response.
- Now working also with the other half, the transmitter part.
- Have not yet worked with the NiosII.

Thank You!