Clean room analysis of the PMS150C/PFS154 programming sequence v0.4

Thanks to everyone involved in investigating the Padauk microcontrollers on the EEV and µC.net forums. Please see here for further background and work leading to this analysis:

http://www.eevblog.com/forum/blog/eevblog-1144-padauk-programmer-reverse-engineering/350/

https://www.mikrocontroller.net/topic/461002

Clean Room Analysis Disclaimer

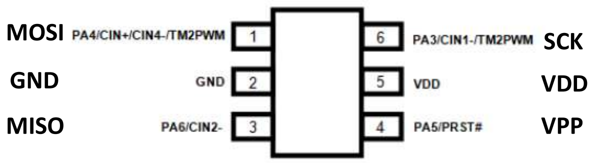
- This report is solely based on analyzing datalogs of the programming sequence as provided here:

  PMC150C: http://www.eevblog.com/forum/blog/eevblog-1144-padauk-programmer-reverse-engineering/msg2096917/#msg2096917
  PFS154C: http://www.eevblog.com/forum/blog/eevblog-1144-padauk-programmer-reverse-engineering/msg2113471/#msg2113471

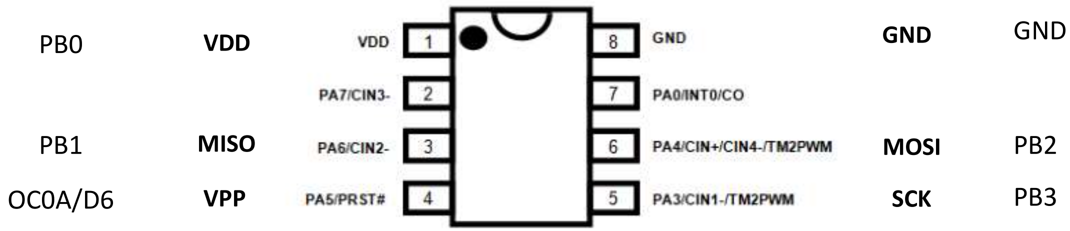
- No attempts have been made by me to reverse engineer any of the software provided by Padauk.

- The information in this document is provided “as is”, without warranty of any kind.

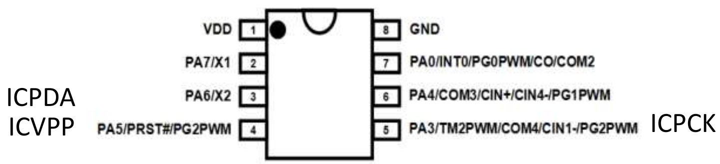
Pinouts



PMS150C-U06 (SOT23-6 60mil)



PMS150C-S08 (SOP8-150mil)
PMS150C-D08 (DIP8-300mil)



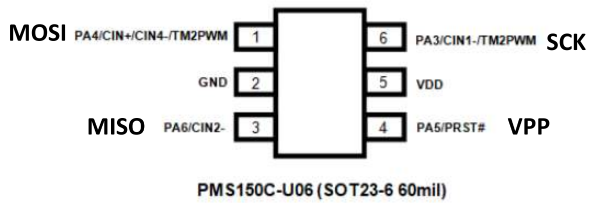
PFS154-S08:SOP8 (150mil)

PMS150C protocol

Initial analysis of data logs.

From a first glance, the programming interface of the PMS150C seems to be a straight forward SPI interface. MSB first, data is valid on rising edge of clk.

The pinout is as follows:



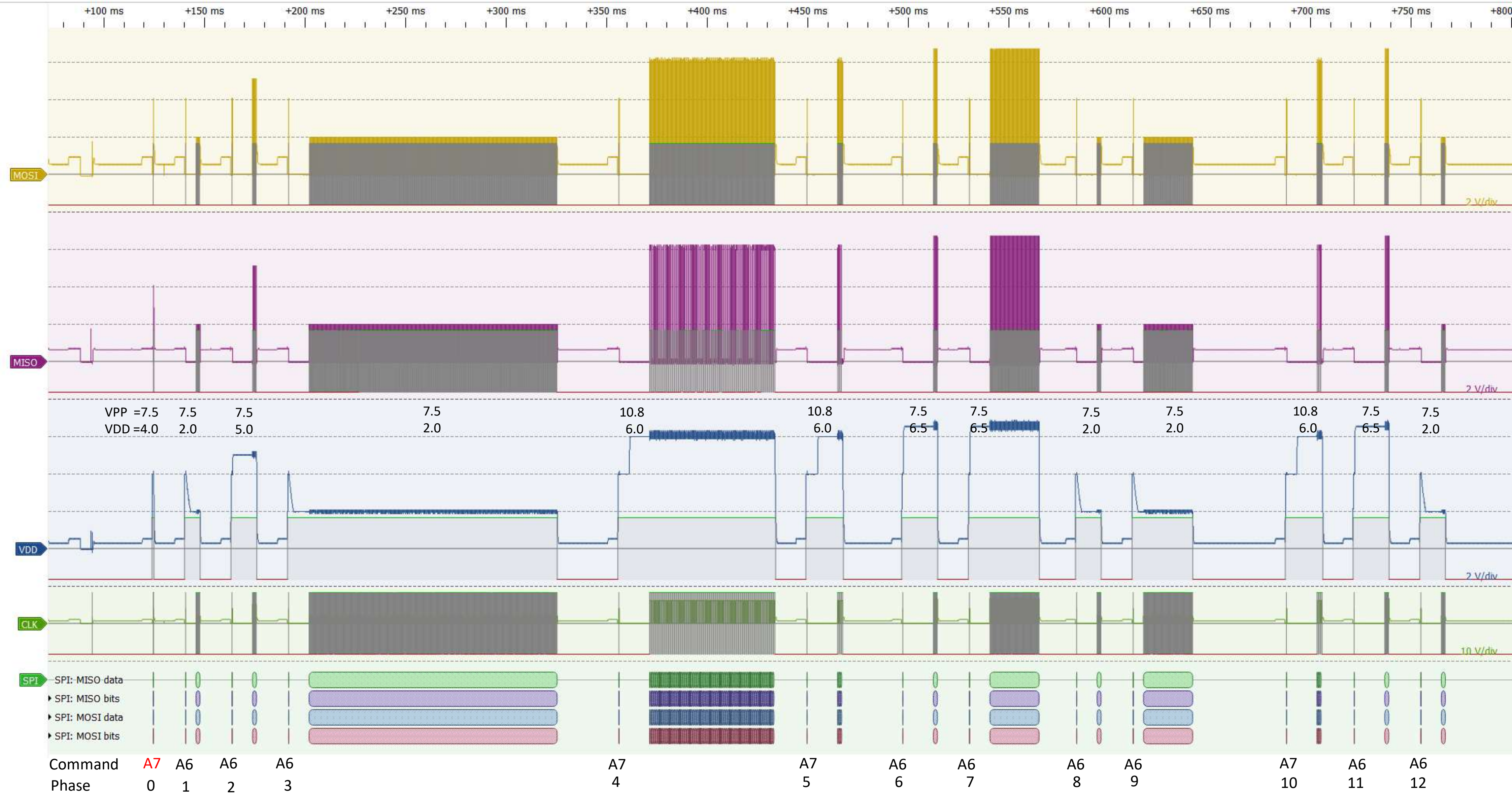
- PA3: SCK/Clock (driven by writer/master)
- PA4: MOSI/Data in (driven by writer/master, data is valid on rising edge of clk. Data is set at arbitrary times due to random timing of writer software)
- PA5: VPP
- PA6: MISO/Data out (driven by MCU, data is valid on rising edge of clk. Data is set on falling edge since the MCU does not have an internal clk)
- Furthermore, the programmer needs to control VDD to reset the MCU.

VPP is 7.5V during read and 10.8 V during writing.

VDD is 6.0 V during programming, 4 V during entry and 6.5 V/2 V for verification. I may be sufficient to keep VDD at 5V if you don't want to verify all corner cases.

Note: The PMS105C is a device with 13x1kbit memory and 13 bit instruction encoding.

Overview of dump2 – writing to previously programmed device

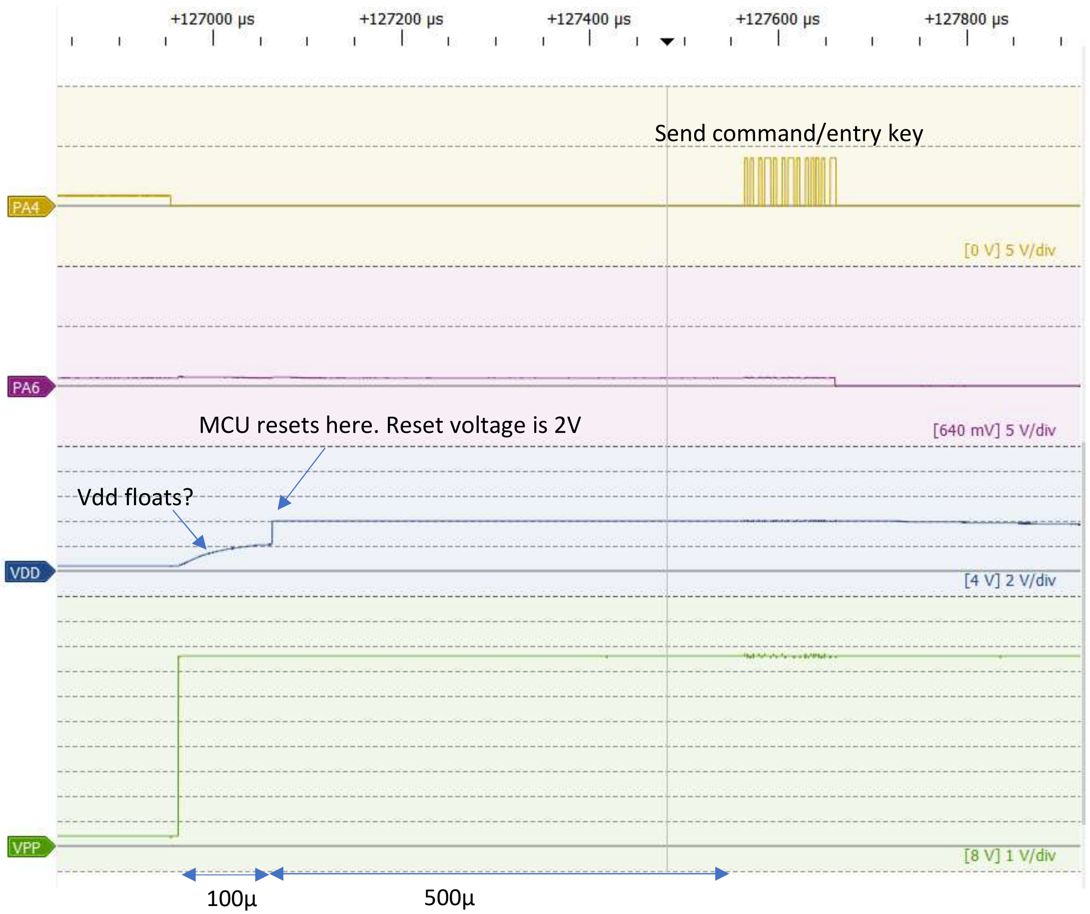


Summary of all phases of the programming sequence (Dump 2)

Phase	Command	VDD	VPP	Description
0	A7 (Write)	4.0 V	7.5 V	Read device ID. This is achieved by initiating a dummy write that is aborted before starting the actualy OTP programming
1	A6 (Read)	2.0 V	7.5 V	Read instruction memory words 0x03F0-0x3FF at low VDD voltage corner. This region contains calibration data.
2	A6 (Read)	5.0 V	7.5 V	Read instruction memory words 0x03F0-0x3FF at standard voltage corner. This region contains calibration data. (Should this be VDD=6.5V? May be a bug)
3	A6 (Read)	2.0 V	7.5 V	Read instruction memory words 0x000-0x3EF at low VDD voltage corner. Dump of full memory.
4	A7 (Write)	6.0 V	10.8 V	Write main memory region between 0x000-0x3EF. Only memory cells used by the program are written to.
5	A7 (Write)	6.0 V	10.8 V	Write to 0x3F6/0x3F8/0x3FC/0x3FE. Housekeeping?
6	A6 (Read)	6.5 V	7.5 V	Read instruction memory words 0x03F0-0x3FF at high voltage corner for verification.
7	A6 (Read)	6.5 V	7.5 V	Read main memory region between 0x000-0x3EF at high voltage corner for verification. Only previously written memory is read.
8	A6 (Read)	2.0 V	7.5 V	Read instruction memory words 0x03F0-0x3FF at low voltage corner for verification.
9	A6 (Read)	2.0 V	7.5 V	Read main memory region between 0x000-0x3EF at low voltage corner for verification. Only previously written memory is read.
10	A7 (Write)	6.0 V	10.8 V	Write to 0x3F6/0x3F8/0x3FC/0x3FE to store clock calibration data and code checksum.
11	A6 (Read)	6.5 V	7.5 V	Read instruction memory words 0x03F0-0x3FF at high voltage corner for verification.
12	A6 (Read)	2.0 V	7.5 V	Read instruction memory words 0x03F0-0x3FF at low voltage corner for verification.

- Note: For a fresh device, clock calibration takes place between steps 9 and 10.
  Two additional phases are inserted (see dump 4).

Enter programming mode



Each phase of the programming sequence is as follows:

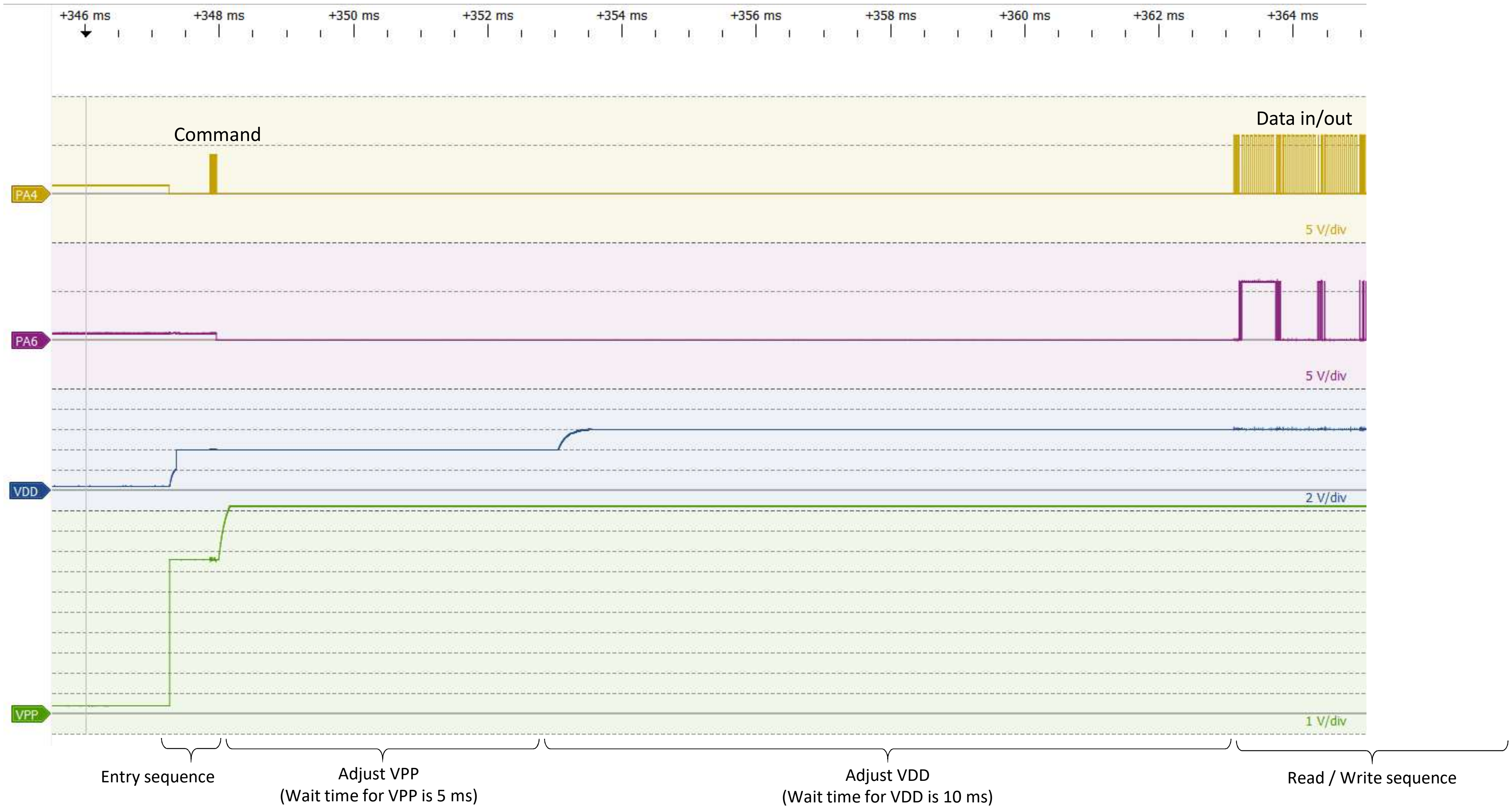
1. Set all pins to GND
2. Drive VPP to 7.5V (possibly >6V is threshold?)
3. wait 100µs
4. Drive VDD to ~4V
5. wait 500µs
6. Send key/command 0xA5A5A5AX (X=6 for read, X=7 for write)
7. Ramp to target VPP (7.5 V for reading, 10.8V for writing)
8. Wait for 5 ms
9. Ramp to target VDD
10. Wait for 10 ms
11. *Perform read or write operation (see later slides)*
12. Pull VDD and VPP to GND

Programming mode is always entered with
Vpp=7.5 V
Vdd=4 V

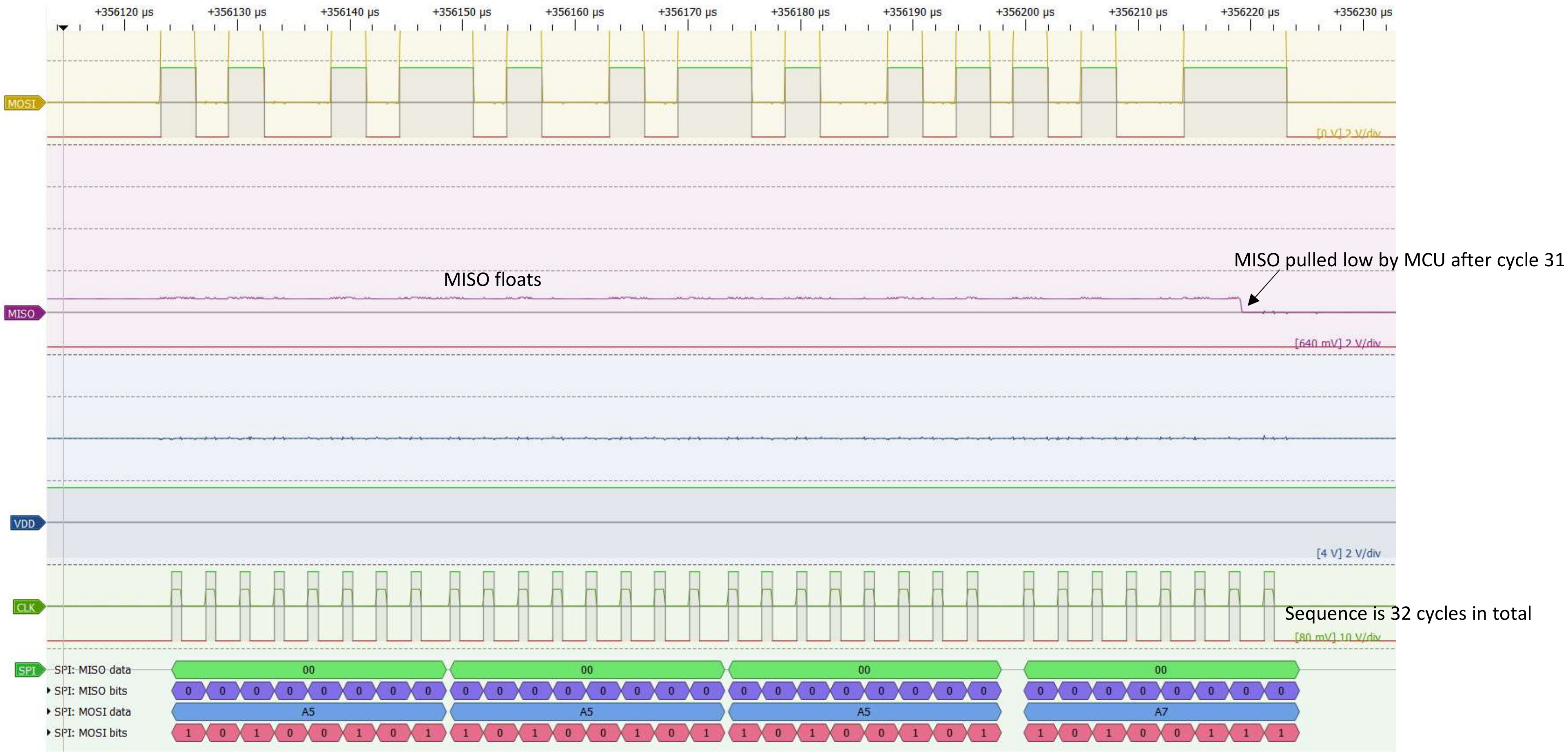
Voltages are only adjusted to final target after sending command (step 6).
Steps 7-10 can be skipped if initial voltages are kept.

Note: Logic levels scale with Vdd. If varying Vdd is implemented, this needs to be considered in the SPI interface hardware (buffer).

Enter programming mode with voltage adjustment and read/write phase

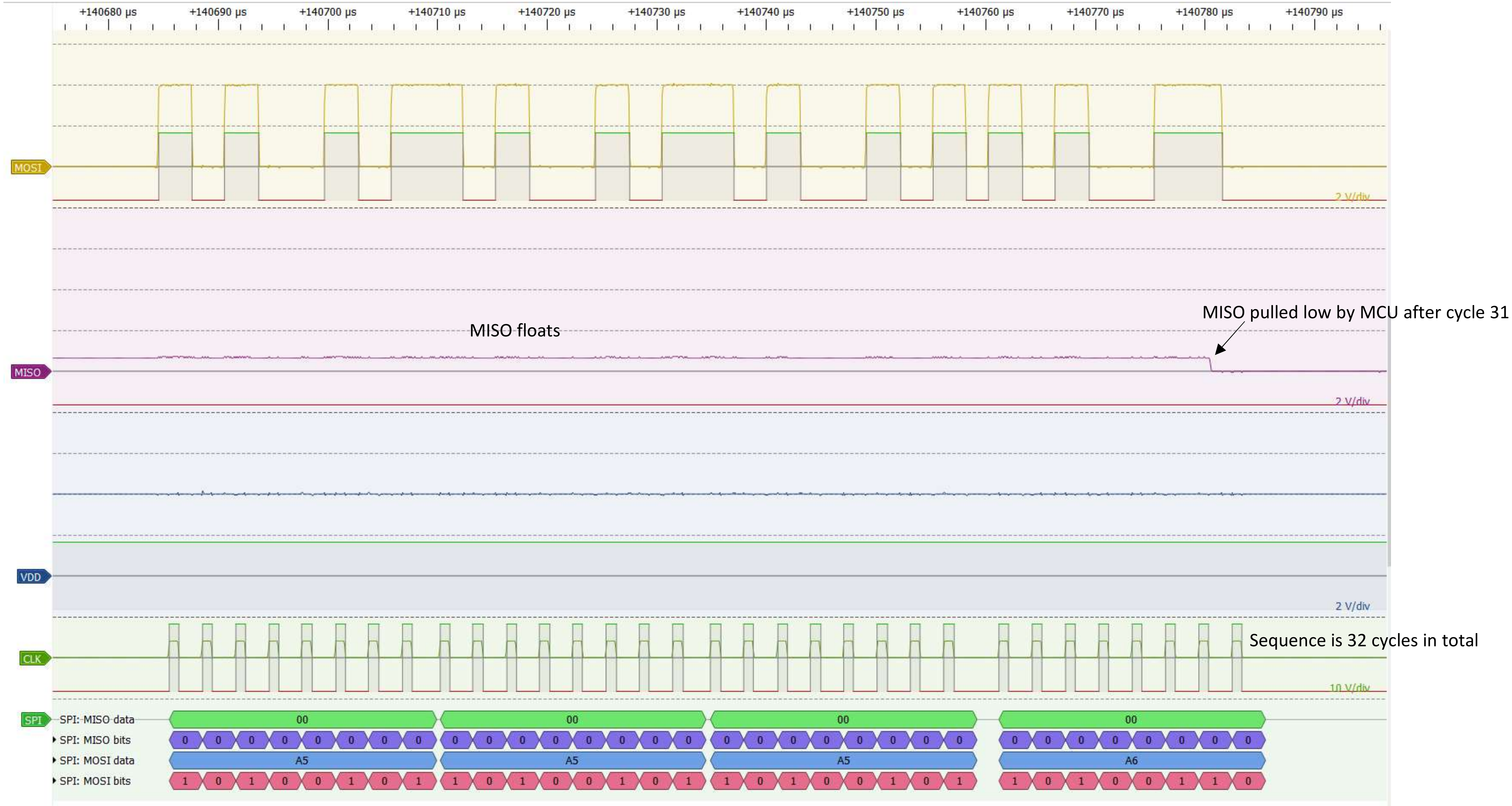


0xA5A5A5A7 key/command – write



Not on PulseView: SPI decoder uses VDD as CS (Active high). This will reset the bitcount when the MCU is reset and ensures proper decoding for magic word
Analog signal were converted to logic by using a threshold of 1.8V (3.3V logic) to also capture the regions with Vdd=2V

0xA5A5A5A6 key/command - read



Command / Entry key encoding

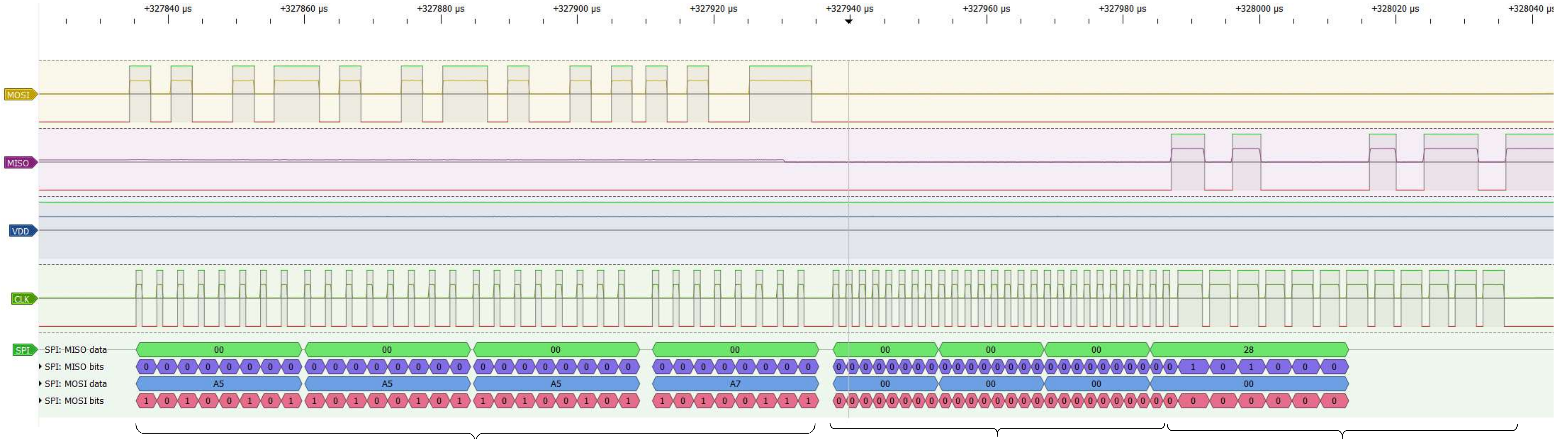
- The entry key is 32 bit and is sent by the master directly after entering programming mode.

- A6 key: 0XA5A5A5A6 -> Initiate reading
- A7 key: 0XA5A5A5A7 -> Initiate writing

- No other codes were found.

- The slave (MCU) pulls MISO down after clock 31. The pin floats before, which could suggest that the programming logic is activated after 31 clocks. This may also suggest that only the LSB is actually used for commands.

Phase 0 – Check device ID - Key-A7, Vdd=4 V, Vpp=7.5 V



32 bits magic key

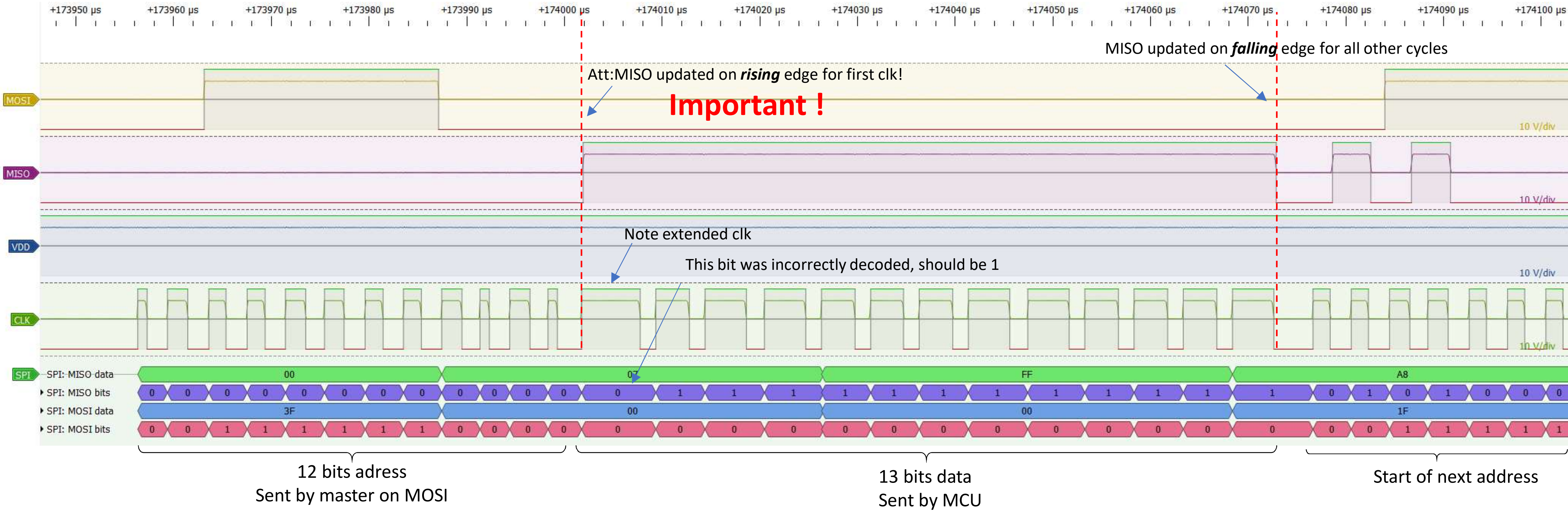
Master sends 26 Bits „0“
=> 2x13 bit empty data word

Master sends 12 bit „0“
=> Adress 0x000
Slave sends 12 bit response
0b101000010110
= 0xA16 device ID

Not that the write execution sequence is omitted so the actual memory write is not started.

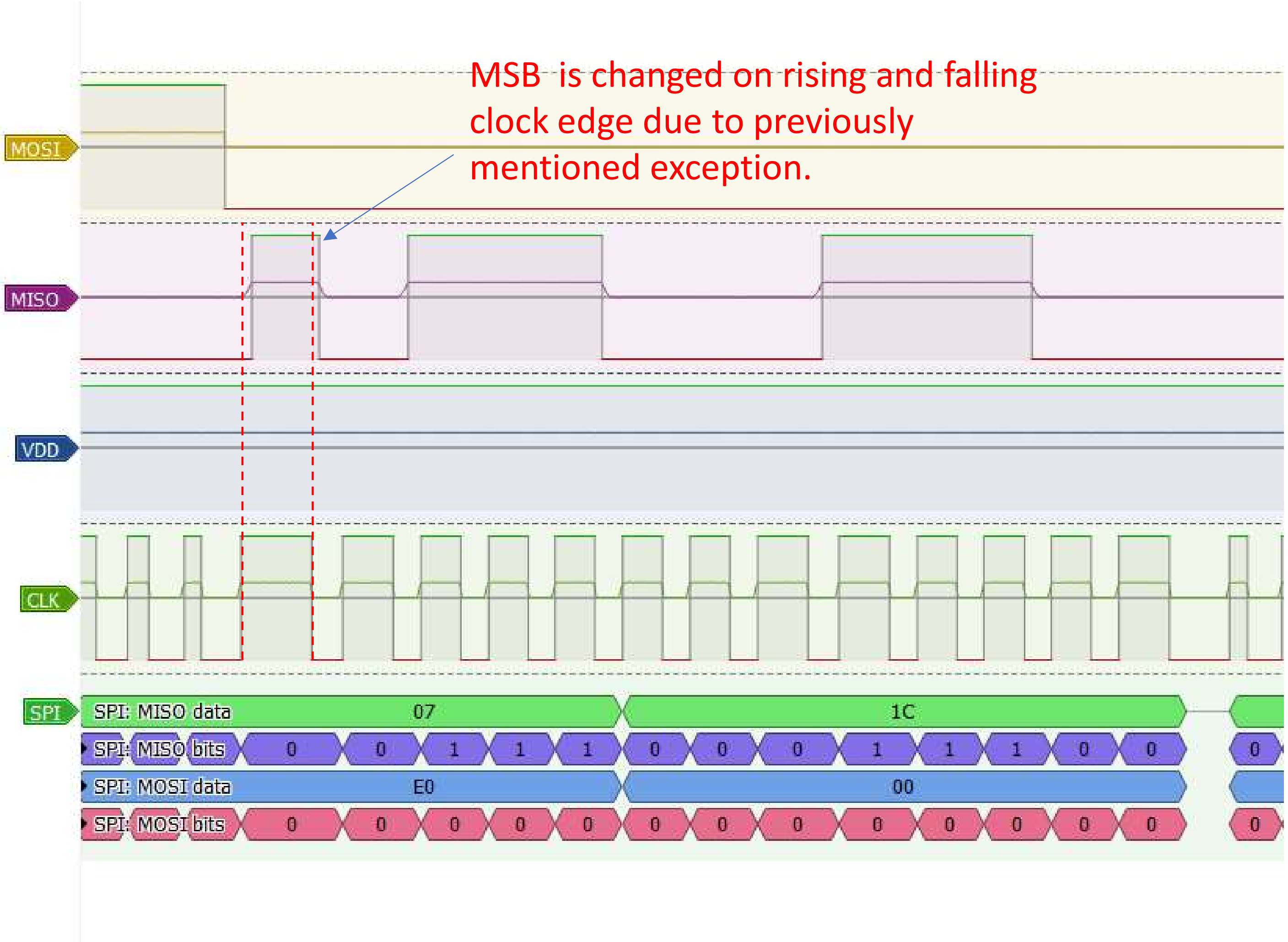
- Phase 0 is basically an incomplete write of 0x0000/0x0000 to 0x000. The write exection cycle is omitted to prevent the dummy data from actually being written.
  The device ID is clocked out of MISO during the adress phase of the write.
- In principle, the device ID could also be read during a read phase.
- Sequence is identical for both previously programmed and clean device (logs 2 and 4)
- Idcode is updated on falling edge! Delay 240-320ns.

Read sequence



- Read sequence is straight forward:
  - Master sends 12 bit adress, slave sends 13 bits of data in subsequent cycles.
- There is one apparent oddity:
  Usually the data on MISO is always updated on the falling clk cycle. However, the MSB of the data is updated with a delay of half a clock cycle, on the rising edge of the clock.
- => For the MSB, it is therefore necessary to set clk high and read MISO after ~2µs before setting clk low again.
- What is the reason for this behavior? No idea, it could be remnant of a direction switching sequence for a bidirectional port.

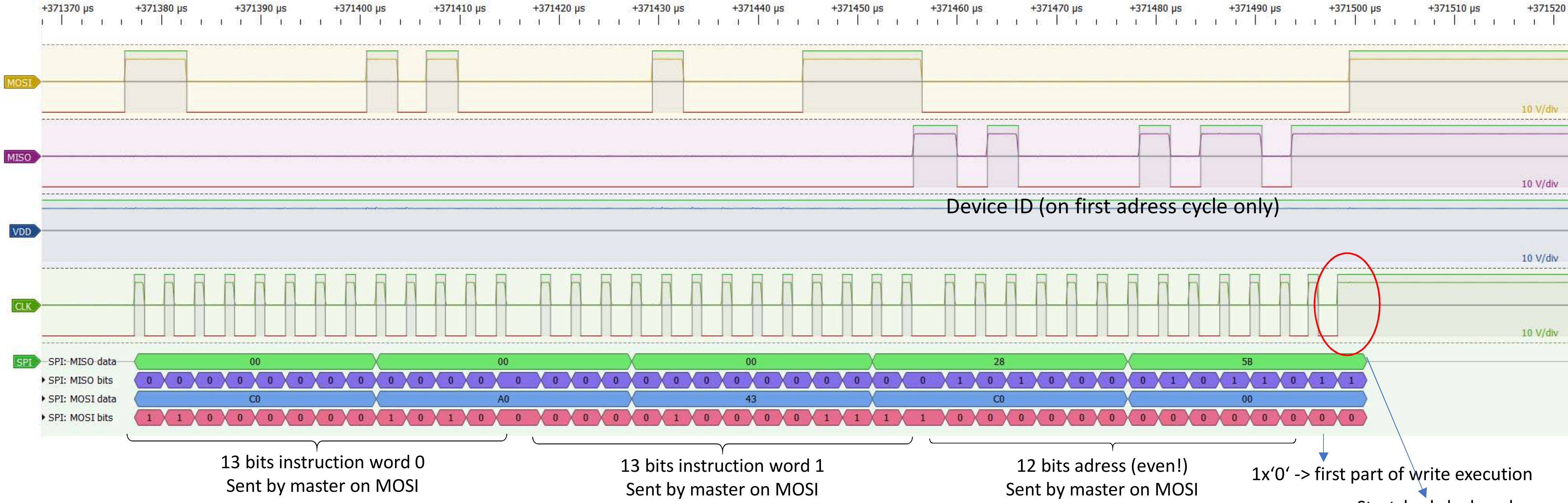
Another example of MISO MSB exception during the read sequence.



Phase 1 – overview – Read 0x3F0-0x3FF (system area)

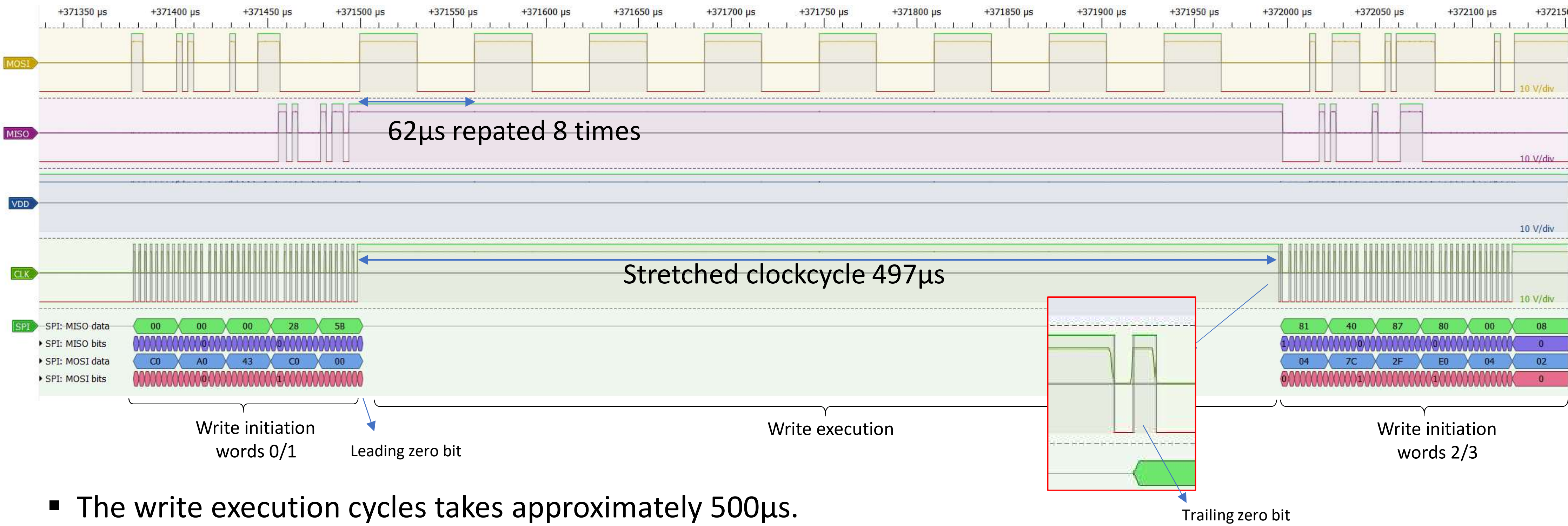
- The MCU seems to send out information during adress cycles as well:
  - During the second adress cycle the device ID is sent.
  - Occasionally the MSB is set on MISO in subsequent adress cycles. Glitch? Parity information?

Write Sequence Part I



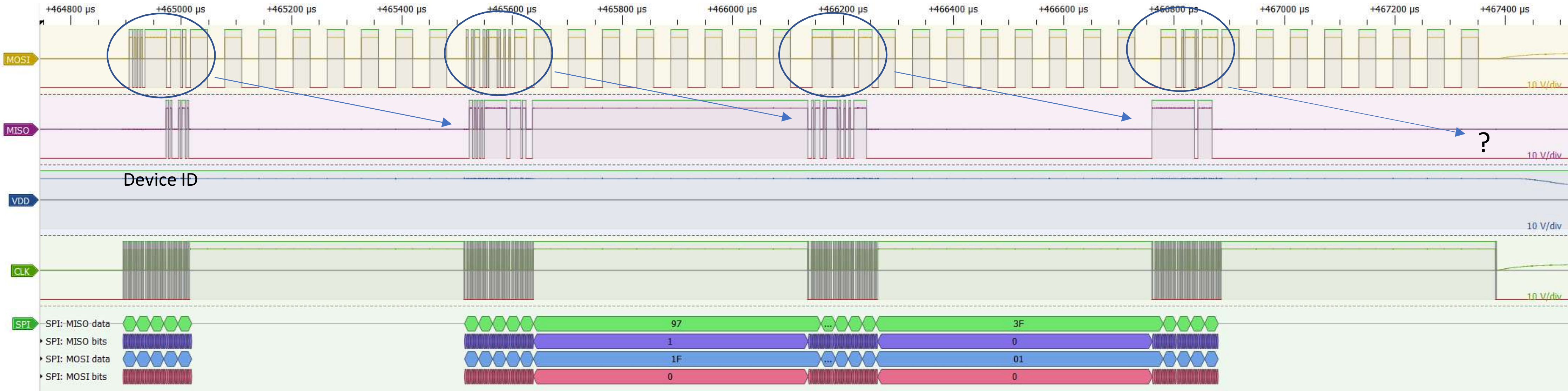
- The PMS150C seems to write two words of memory per write cycle.
- Each write cycle consists of the the following initiation sequence:
  - Send 2x 13 bit instruction words
  - Send 12 bit adress word. (Needs to be dividable by two ?).
  - Send a single „0“ bit. The write cycle seems to be aborted if the device is powered down before sending this bit.
  - The next low-> high transition of the clk seems to initiate the write.

Write Sequence Part II



- The write execution cycles takes approximately 500µs.
  - The sequence consists of three steps:
    - 1) Send 1x‘0‘ at normal clk – „Leading zero bit“
    - 2) Stretched clock cycle (497µs) while MOSI is clocked at 16 kHz
    - 3) Send 1x‘0‘ at normal clk – „Trailing zero bit“
  - It appears a secondary clock signal of 16kHz is provided on MOSI. Very odd.

Write Sequence Part III – full sequence overview of phase 5

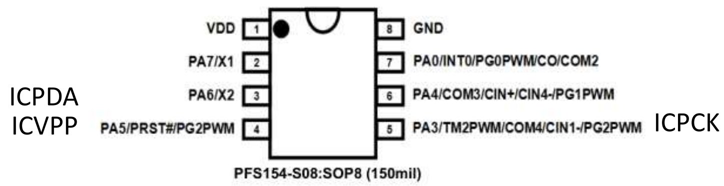


- The last write execution cycle simply ends with the H->L transition of SCLK.
- The MCU will also output data on MISO during the write initiation cycle.
  - First adress cycle: The device ID is clocked out.
  - Subsequent cycles: The data and adress words of the previous cycle are repeated.
    This may be a feature to verify the correctness of the written data without a separate read phase. However, it seems that the current software is not making use of it since the information of the last write cycle is discarded.
- Edit: As noted in the EEV forum, this is most likely just information that was clocked out from the SPI register in the the previous cycle.

PFS154 protocol

Overview

PFS154



PA3 = ICPCK
PA5 = ICVPP
PA6 = ICPDA
VDD

The programming interface of the PFS154C as based on a serial interface with bidirectial data line. MSB first, data is valid on rising edge.

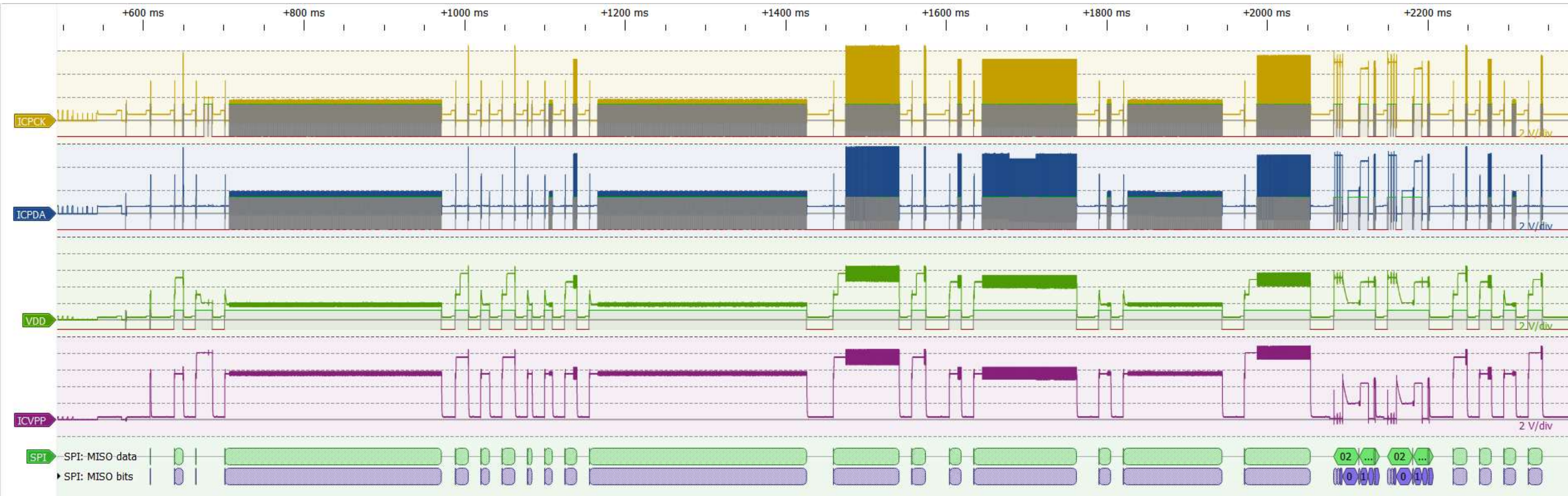
The pinout is as follows:

- PA3: ICPCK/Clock (driven by writer/master)
- PA5: VPP
- PA6: ICPDA/Data InOut (driven by slave/MCU or master, depending on bus direction)
- Furthermore, the programmer needs to control VDD to reset the MCU.

ay be sufficient to keep VDD at 5V if you don't want to verify all corner cases.

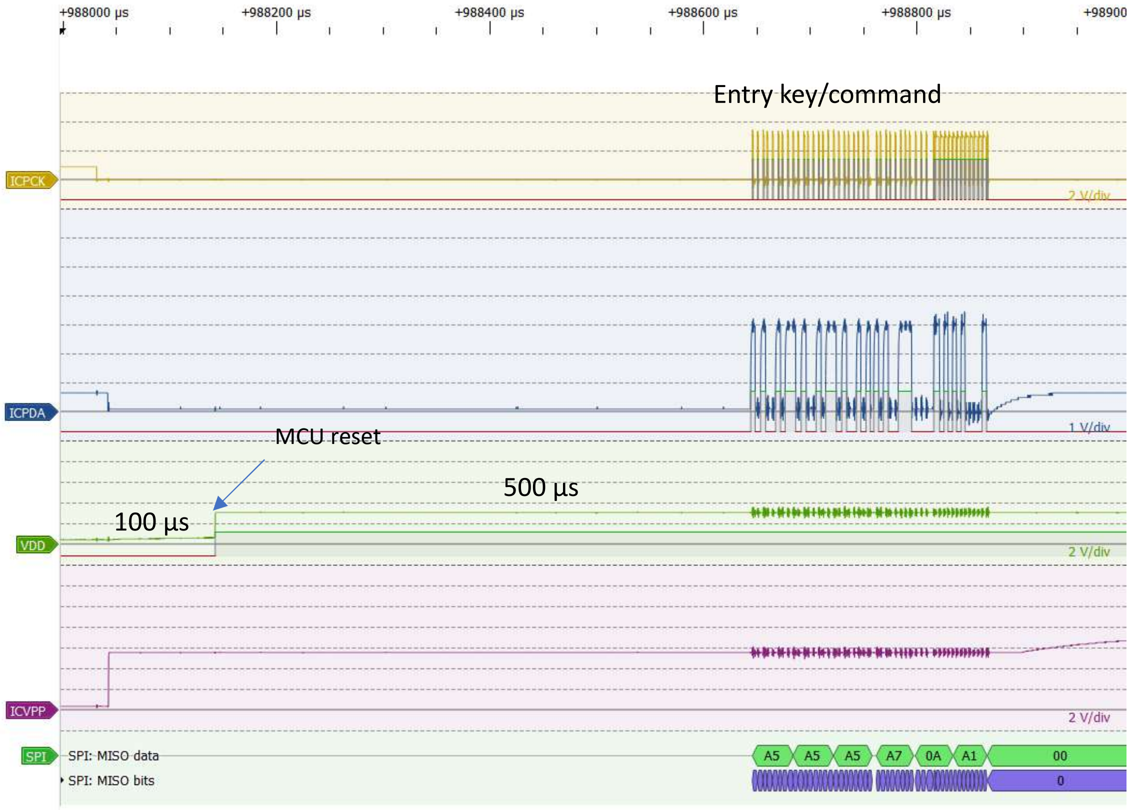
The PFS154C is a device with 14x2kbit memory and 14 bit instruction encoding.

PFS154 full programming sequence overview



Phase	0	1	2	3	4	5	6	7	8
Command	A6	A6	A3	A6	A7	A6	A7	A6	A6
Vdd	3.0	5.0	2.0	2.0	5.5	2.0	5.8	2.0	5.0
Vpp	5.5	5.5	8.0	5.5	7.5	5.5	7.5	5.5	5.5

PFS154 entry sequence



Each phase of the programming sequence is as follows:

1. Set all pins to GND
2. Drive VPP to 5.5V (threshold is relative to VDD)
3. wait 100µs
4. Drive VDD to ~3V
5. wait 500µs
6. Send key/command 0xA5A5A5AX (X=6 read, X=7 write, X=3 erase)
7. Receive response (see later slides)
8. Ramp to target VPP (5.5 V for reading, 7.5V for writing, 8V for erasing)
9. Wait for 5 ms
10. Ramp to target VDD
11. Wait for 10 ms
12. *Perform read or write operation (see later slides)*
13. Pull VDD and VPP to GND

Programming mode is always entered with
Vpp=5.5 V
Vdd=3 V

Voltages are only adjusted to final target after sending command (step 6).
Steps 8-11 can be skipped if initial voltages are kept.

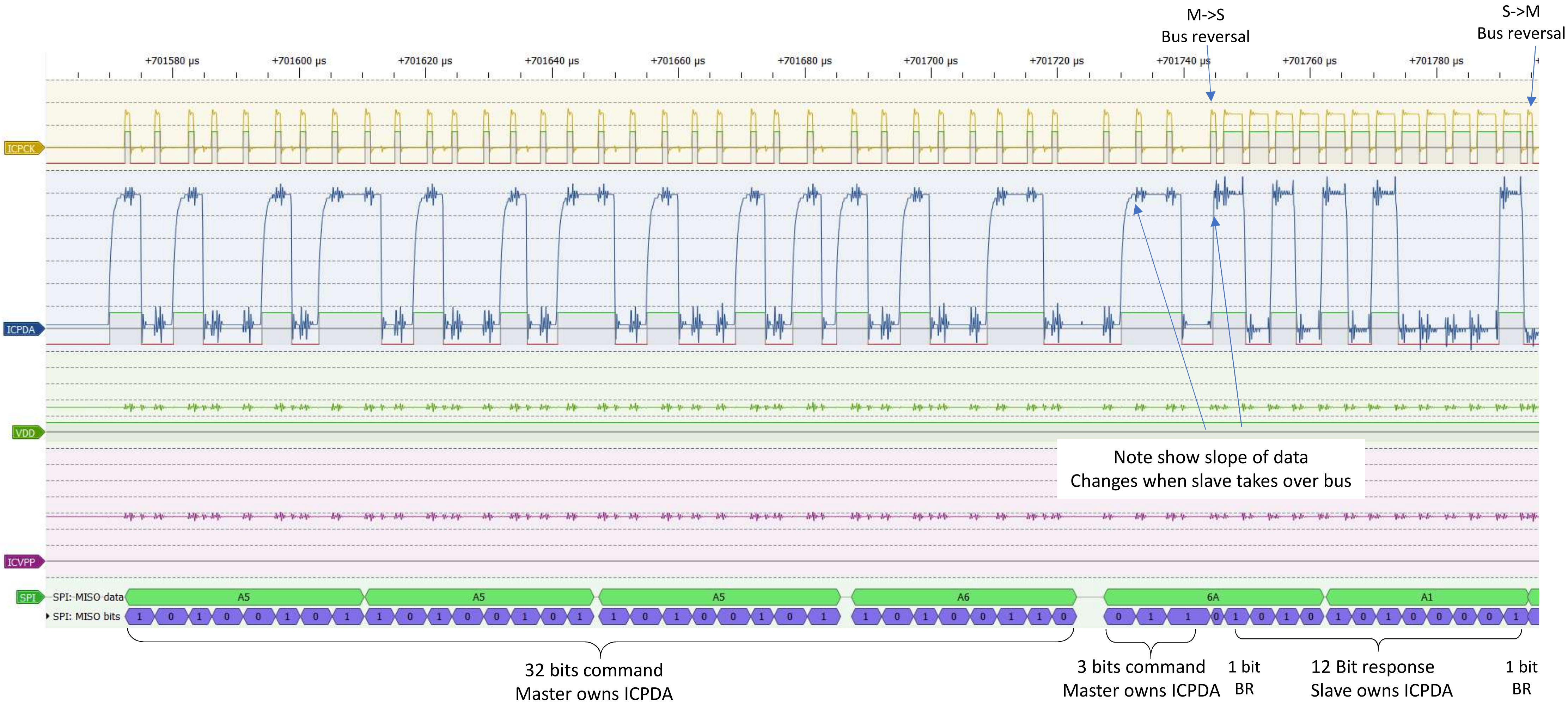
Note: Logic levels scale with Vdd. If varying Vdd is implemented, this needs to be considered in the SPI interface hardware (buffer).

- Entry mode is very similar to PMS150C. Main differences are marked in red.
- Note: The most important factor to activate the programming mode is the difference between VPP and VDD during MCU reset.
  -> VPP-VDD>=2V!   If a higher VDD is used, also VPP needs to be increased accordingly.
  E.g. VDD=5V and VPP=6V will fail to enter programming mode. The difference between VPP and VDD can be reduced after entry.

Full sequence

- Same as for PMS150C

Command sequence (Phase 3)



Command / Entry key encoding

- Full command sequence:
  - 32x1 bit command (Master drives data line)
  - 3x1 bit dummy* (Master drives data line)
  - 1x1 bit bus reversal (Master releases data line)
  - 12x1 bit DeviceID (Slave drives data line)
  - 1x1 bit bus reversal (Slave release data line)
  - 48 clock cycles in total.

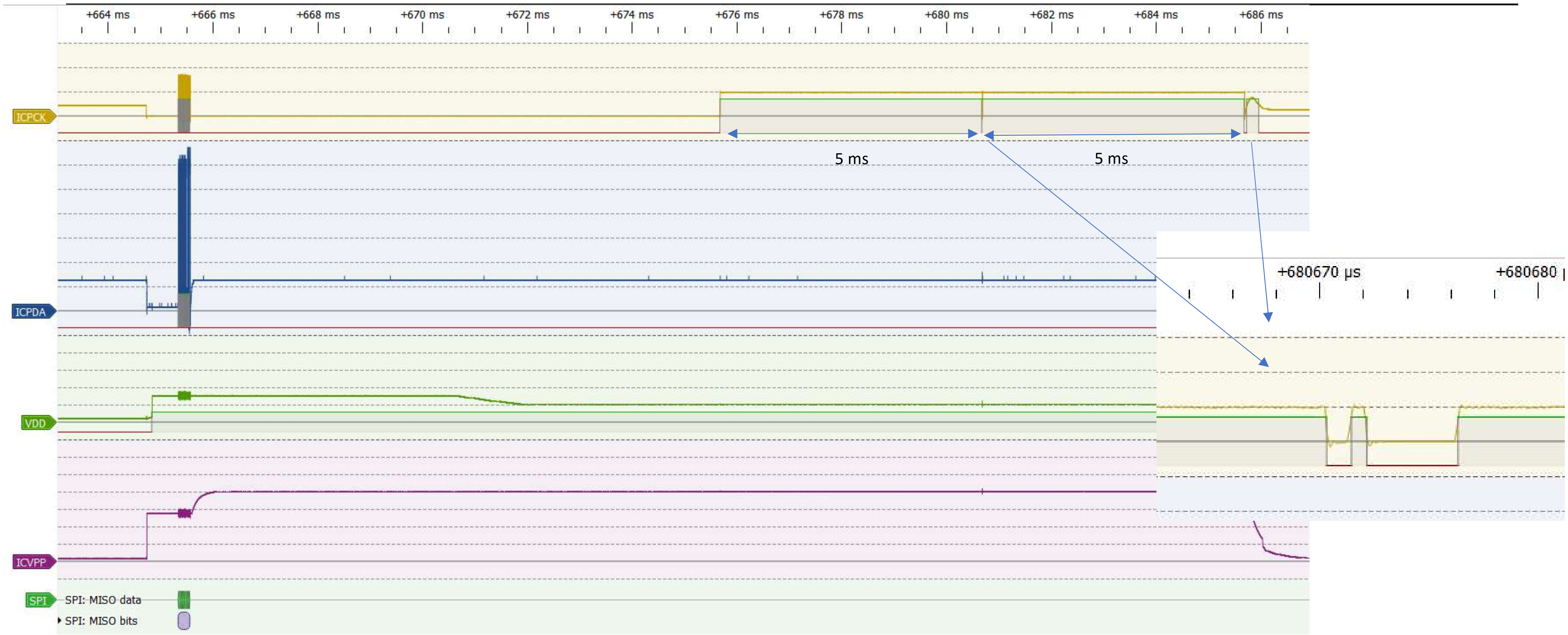
- The entry key is 32 bit and is sent by the master directly after entering programming mode.

- A6 key: 0XA5A5A5A6 -> Initiate reading
- A7 key: 0XA5A5A5A7 -> Initiate writing
- A3 key: 0xA5A5A5A3 -> Initiate Erase

- DeviceID of PFS154 is 0xAA1

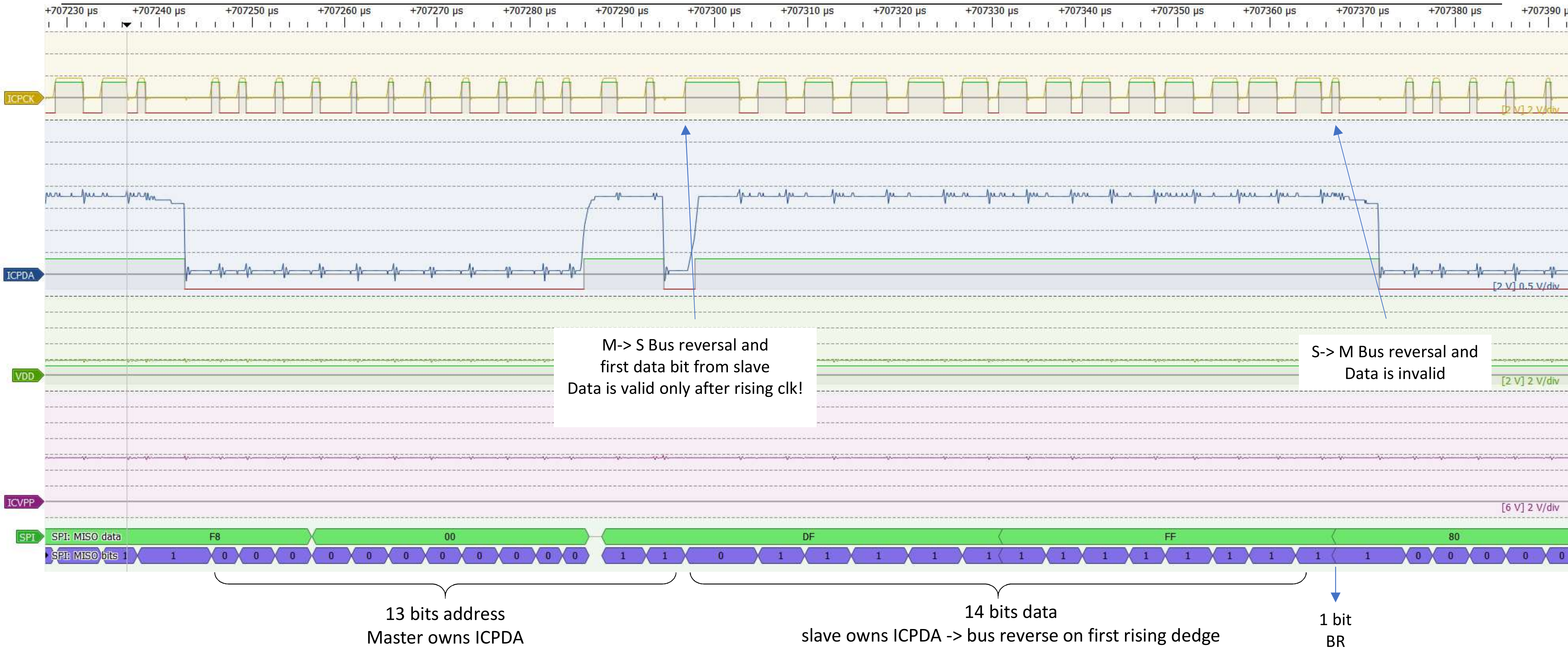
- *The three dummy bits are usually „000“, except in phase 3 where they are „011“. Bug? No impact was observed when changing these bits.

Erase Sequence



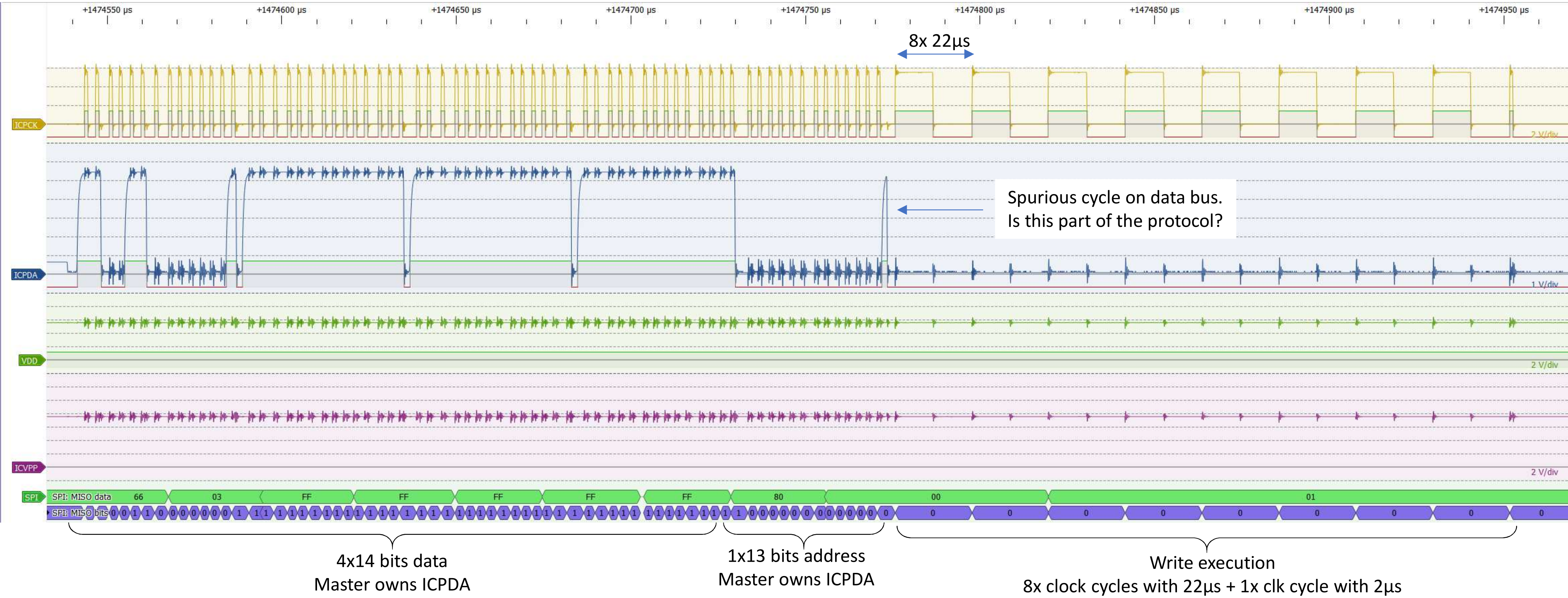
- Erase sequence:
  - Entry key is 0XA5A5A5A3
  - Ramp to VPP=8V, Vdd=2V
  - Repeat 2x: (stretched clock 5ms, normal clock 2µs)   -> 4 clock cycles in total

Read Sequence



- Read Sequence
  - Command: 0XA5A5A5A6
  - Repeat for every word: write 13 bits adress + read 14 bit data + 1 bit bus reversal

Write Sequence



- Write Sequence
  - Command: 0XA5A5A5A7
  - Ramp VDD=5.8V VPP=7.5V
  - Repeat for every page of four words: write 4x14 bit data, 1x13 bit adress, write execuction sequence

Revisions

▪ V0.1 – Jan 7th, 2019 – cpldcpu.	Initial report.
▪ V0.11 - Jan 7th, 2019 – cpldcpu.	Updated clean room disclaimer and front matter.
▪ V0.2 - Jan 8th, 2019 – cpldcpu.	Corrected device ID, added pinouts, corrected write mode description.
▪ V0.3 – Jan 9th, 2019 – cpldcpu	More pinouts, corrected write execution sequence (p.17), updated p.18
▪ V0.4 – Jan 13th, 2019 – cpldcpu	Added PFS154 section
▪ V0.5 – April 27th, 2019 – cpldcpu	Corrected PFS154 voltages