Parallel port drivers

I can't do a full test here as I do not have a slim CB so these are logic analyser traces only.

The command interface provided to the PC differs from the native USB controller in the following ways, but in essence the PC program does not need to care whether it is using the parallel port version or the native USB version – it uses commands A0-A3, A5, and B0-B2 regardless of the version of the firmware running on the USB controller. The USB controller presents the same interface to the PC.

How is the right firmware loaded? Just program the eeprom on the USB controller with the right identity and the code loads automatically.

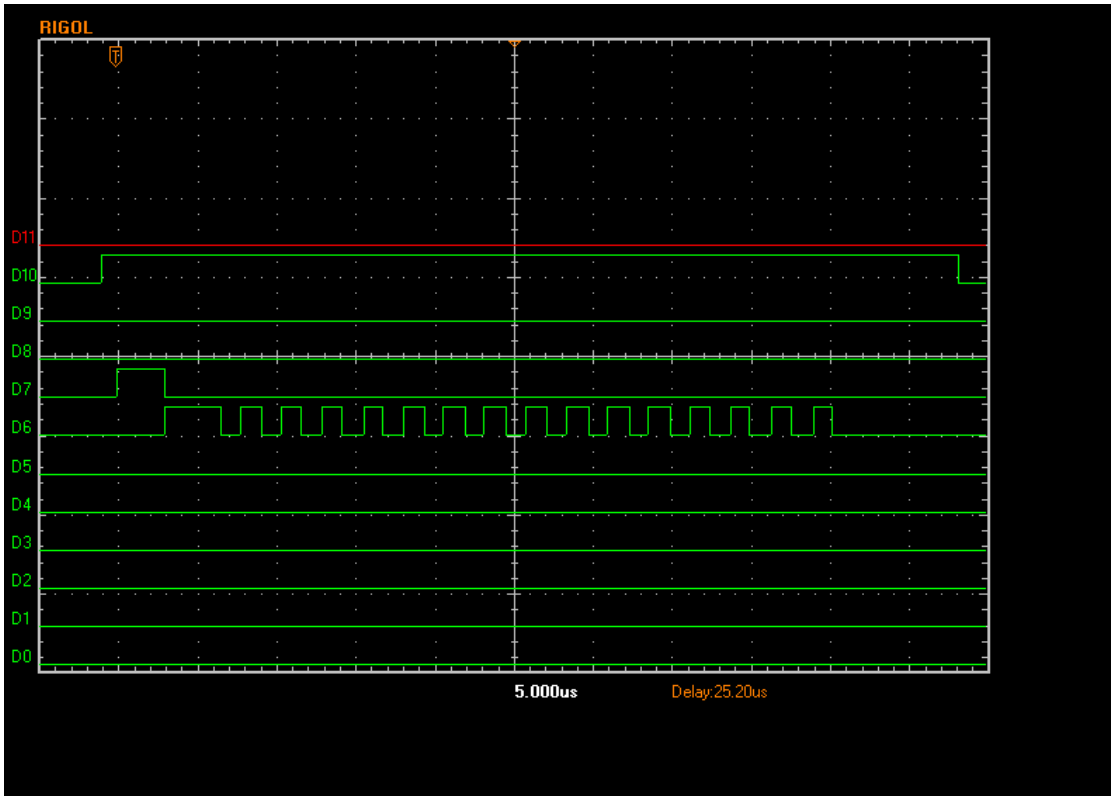
USB Command Code	Function	Comments
A0	Write port A	Writes to parallel port P3 – this accesses the ADC control lines in the same way as native USB controller port PA
A1	Write port B	Writes to parallel port P1 – this accesses the DDA and PLL data and clock lines as well as filter select
A2	Write port C	Writes to parallel port P4 – currently unused
A3	Write port D	Writes to parallel port P2 – this accesses the FQUD and LE lines plus invert PDM
A4	Write port E	Has no parallel port equivalent
A5	Port Reset	Sets the parallel port data lines all low then pulses all 4 latch lines high then low again, setting every output line low
A6	Raw write	Bypasses the parallel port emulation and writes natively to USB ports A-E
B0	Read mag ADC	Writes to parallel port P3 to perform a conversion and clock the result back in via Port A bit 5
B1	Read phase data	Writes to parallel port P3 to perform a conversion and clock the result back in via Port A bit 4
B2	Read both ADCs	Writes to parallel port P3 to perform a conversion and clock the result back in via Port A bits 5 and 4

ADC example

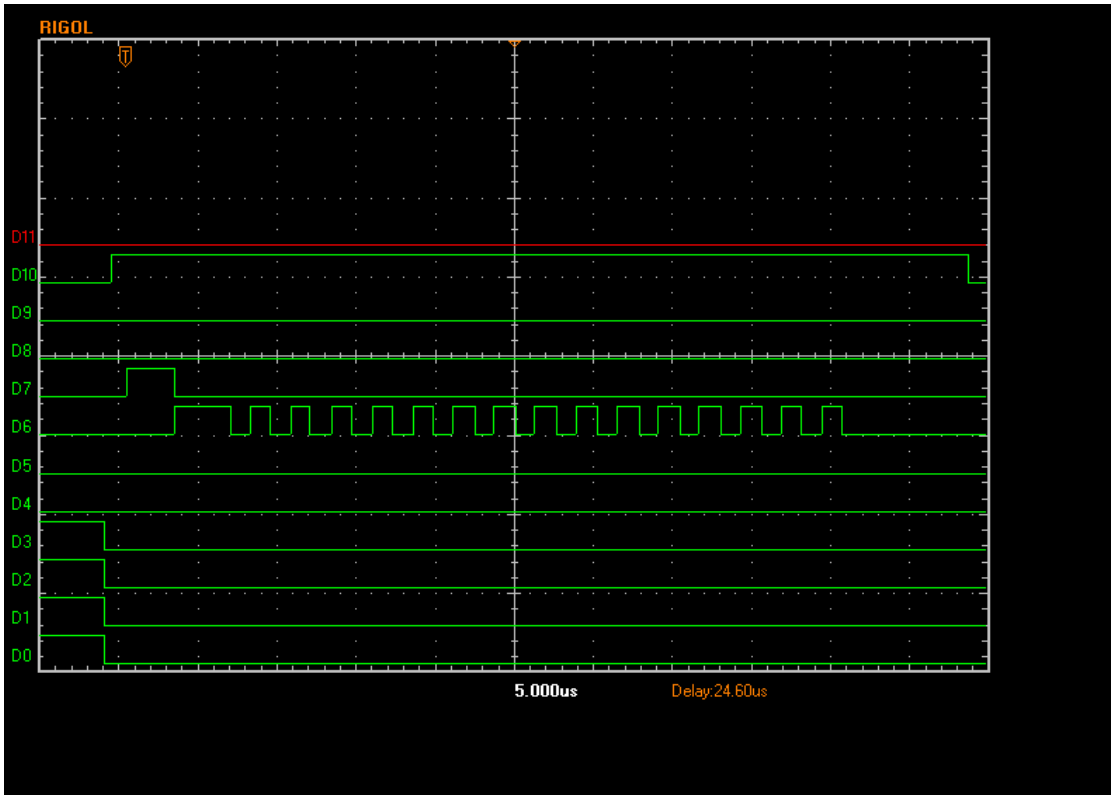
ADC reading; this command to the USB card B1 02 01 10 01 requests conversion of both ADCs and the logic analyser trace is as shown

Here the signal lines are as follows

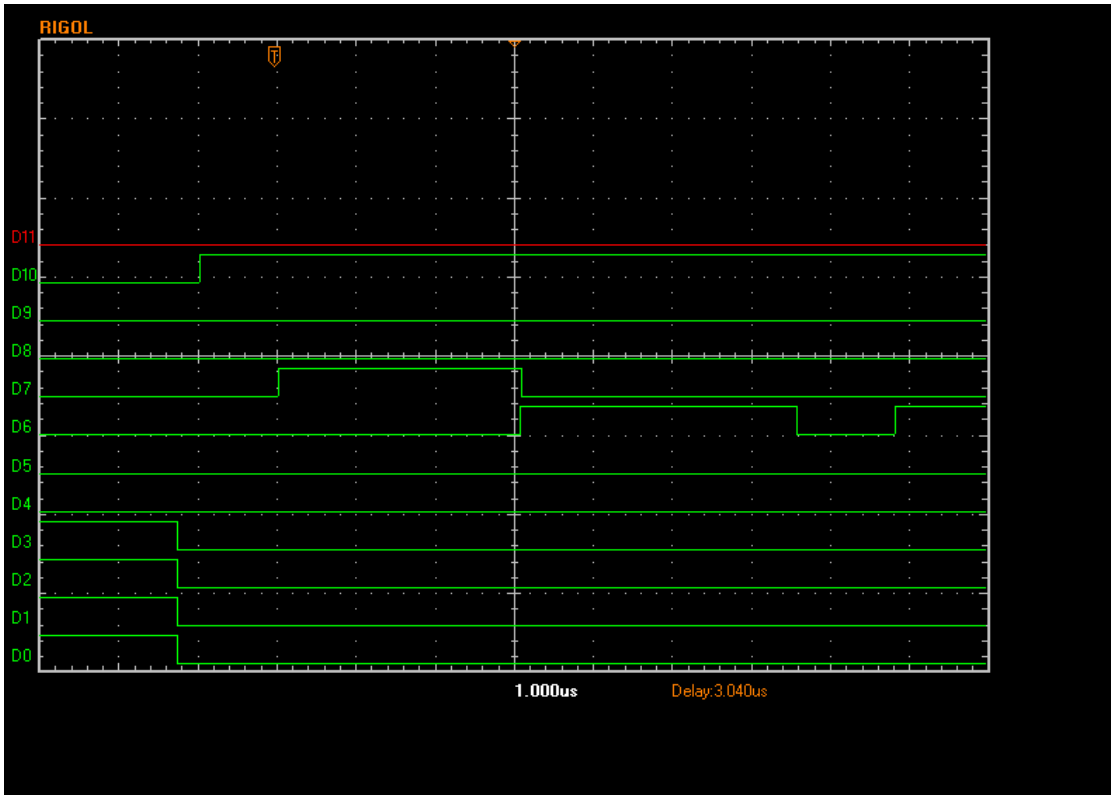
LA line	Connects to	Comment
D0-7	D0-7	These are the 8 data pins into the parallel port
D8	SELT	Latches bus lines into P1
D9	INIT	Latches bus lines into P2
D10	AUTO	Latches bus lines into P3
D11	STROBE	Latches bus lines into P4



The data bus is set to zeros, then D10 is raised. You can't see the bus being cleared so here the low 4 bits of the bus have been forced high before the convert function is called



And zooming in on the start



The bus is forced low (D0-7). D10 is then raised (AUTO, opening the P3 latches and giving write access to ADConv and ADSerClk). 1us later, ADConv is raised (D7) and

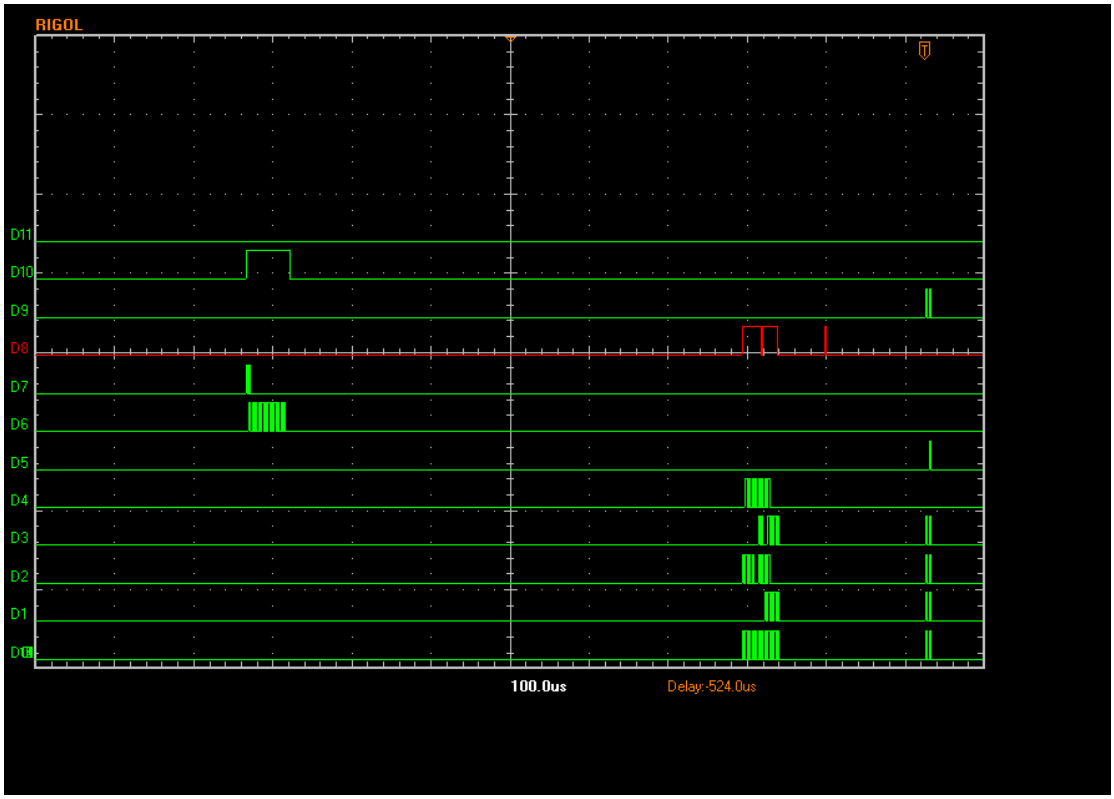
kept high for just over 3 usec. It is then lowered and ADSerClk is raised to start clocking out the result.

The result data is not shown above – it is on ACK and WAIT which connect to USB controller port A bits PA4 and PA5 respectively. This data is clocked in and presented as the result in the same way as it is handled in the native USB case (see the ADC conversion main document)

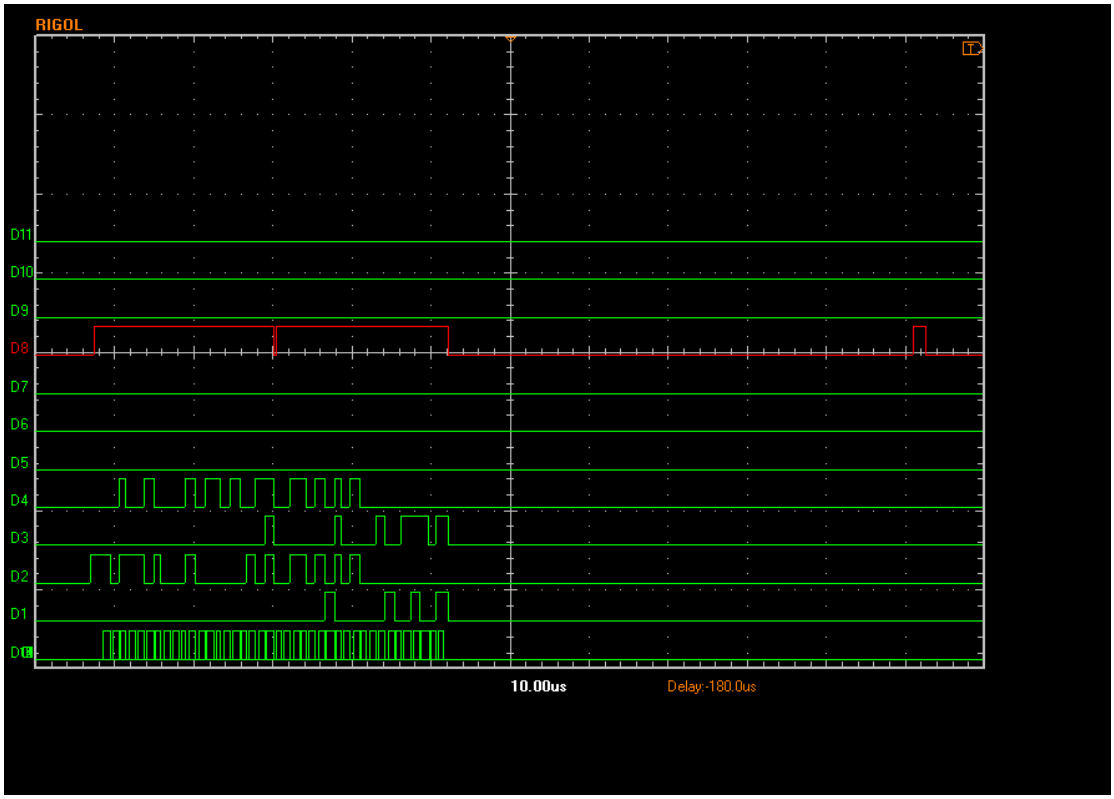
After clocking out the data, the latch signal (D10) is set low.

Command All Slims

Here is the trace for a section between steps. D5/D7/D10 activity is the ADC conversion as part of a step in the scan. D0-4 & D7 show the CommandAllSLims code setting the DDS and PL devices for the next step in the scan, followed in D8 alone by the dll code latching the filtbank data. Finally at the far right is the FQUD & LE lines of the slims.

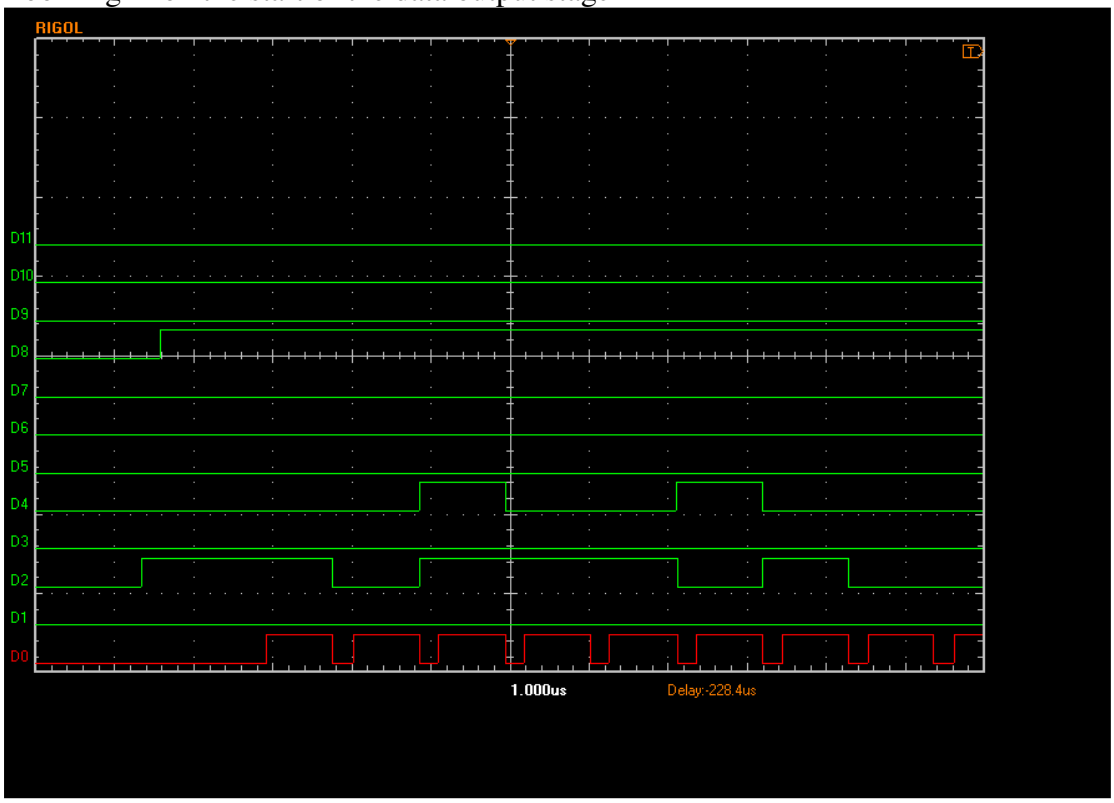


Zooming in on the writing of data to the SLIMs shows this

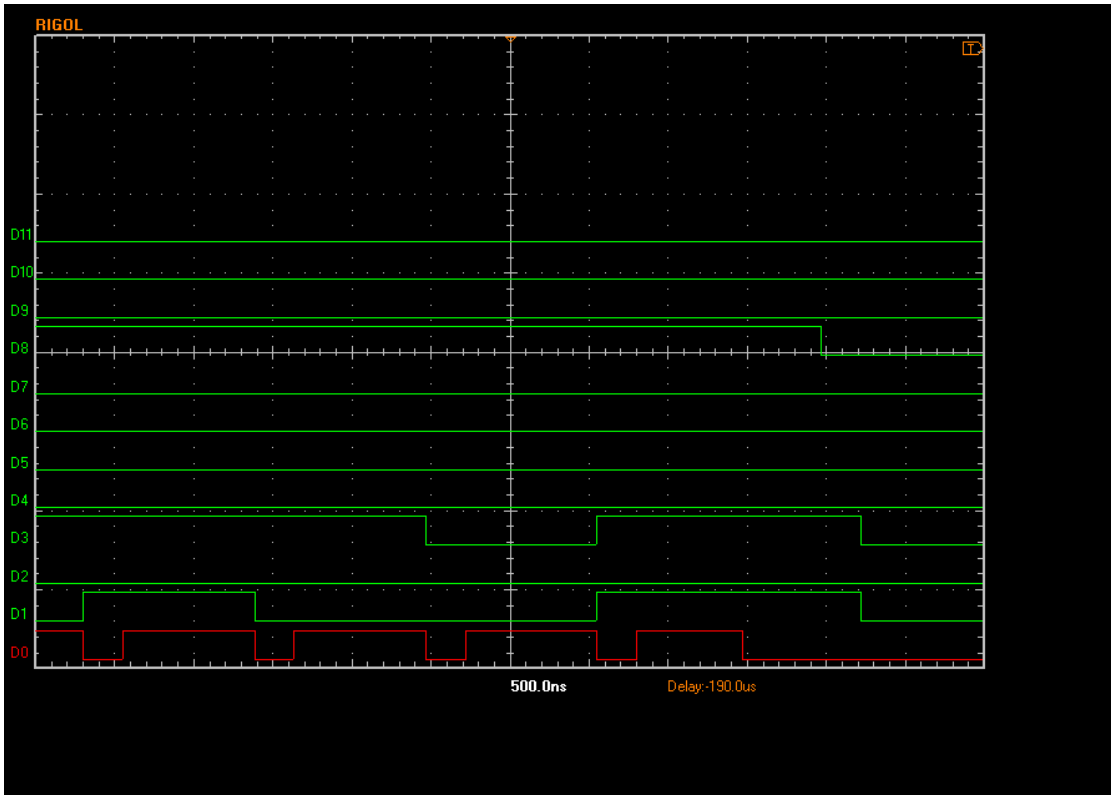


Ignore the transient in the middle of the signal high of D8 – this is an analyser glitch. The data lines are set, the SLIM-CB latch line raised and the data is clocked out into the devices. When it is over, the latch line is lowered, and raised again briefly 60 usec later to latch filtbank.

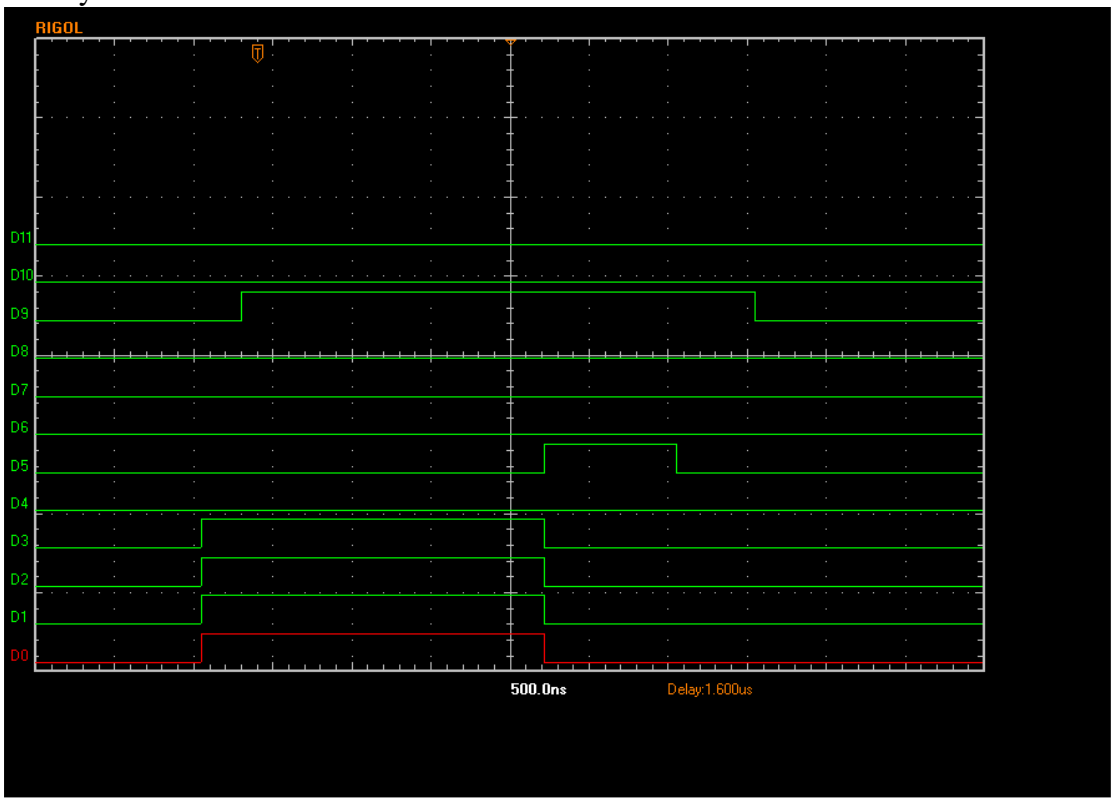
Zooming in on the start of the data output stage



The cloking cycle for the data is just over 1usec per bit, and there are 40 bits that are clocked. At the end of the 40 bits (below) the latch line is lowered again.



Finally here is the latch event



The data lines are set and the latch line raised, which latches data into the DDS and PLL chips. Then pdm data is latched by D5 and the bus latch closed again