Here is CommandAllSlimsUSB as currently coded

```
[CommandAllSlimsUSB]
    USBwrbuf$ = "A12801"
    for clmn = 0 to 39  'ver113-3c
        USBwrbuf$ = USBwrbuf$ + ToHex$( cmdallarray(thisstep,clmn)+ filtbank )
    next clmn
    if USBdevice <> 0 then CALLDLL #USB, "UsbMSADeviceWriteString", USBdevice as long, USBwrbuf$ as ptr, 43 as short, result as boolean

    USBwrbuf$ = "A10100"+ ToHex$( filtbank )
    if USBdevice <> 0 then CALLDLL #USB, "UsbMSADeviceWriteString", USBdevice as long, USBwrbuf$ as ptr, 4 as short, result as boolean

    pdmcmd = phaarray(thisstep,0)*64 'ver111-39d
    USBwrbuf$ = "A30300"+ToHex$(le1 + fqud1 + le3 + fqud3 + pdmcmd)+ToHex$(pdmcmd + 32)+ToHex$(pdmcmd)
    if USBdevice <> 0 then CALLDLL #USB, "UsbMSADeviceWriteString", USBdevice as long, USBwrbuf$ as ptr, 6 as short, result as boolean

    lastpdmstate=phaarray(thisstep,0)   'ver114-6c
    return
```

The parallel port coding is this

```
[CommandAllSlims]'for SLIM Control and SLIM modules. Old PDM and old Filt Bank can be used 'ver111-31c
    for clmn = 0 to 39  'ver113-3c
       a= cmdallarray(thisstep,clmn)+ filtbank
       out port, a : out control, SELT:out control, contclear 'a is the data, without clock
       out port, a+1:out control, SELT:out control, contclear 'a+1 is data, plus clock
    next clmn
    out port, filtbank 'remove data, leaving filtbank data to filter bank.
    out control, SELT:out control, contclear 'disable buffer. filtbank signals will be latched to filter bank assembly

    pdmcmd = phaarray(thisstep,0)*64 'ver111-39d
    out port, le1 + fqud1 + le3 + fqud3 + pdmcmd 'present data to buffer input'ver111-39d
    out control, INIT: out control, contclear  'latch the buffer, moving the signals to the 5 modules'ver113-2a
    out port, pdmcmd + 32 'remove LEs and Fquds, leaving PDM data, but add a latch signal P2D5 for old PDM if used.'ver111-39d
    out control, INIT: out control, contclear  'sends latch signal to old PDM'ver113-2a
    out port, pdmcmd  'remove the added latch signal to PDM, leaving just the PDM's static data'ver111-39d
    out control, INIT: out control, contclear  'ver113-2a
    out port, 0    'bring all Data lines low. PDM data remains static
    lastpdmstate=phaarray(thisstep,0)   'ver114-6c
    return 'to [CommandThisStep]
```

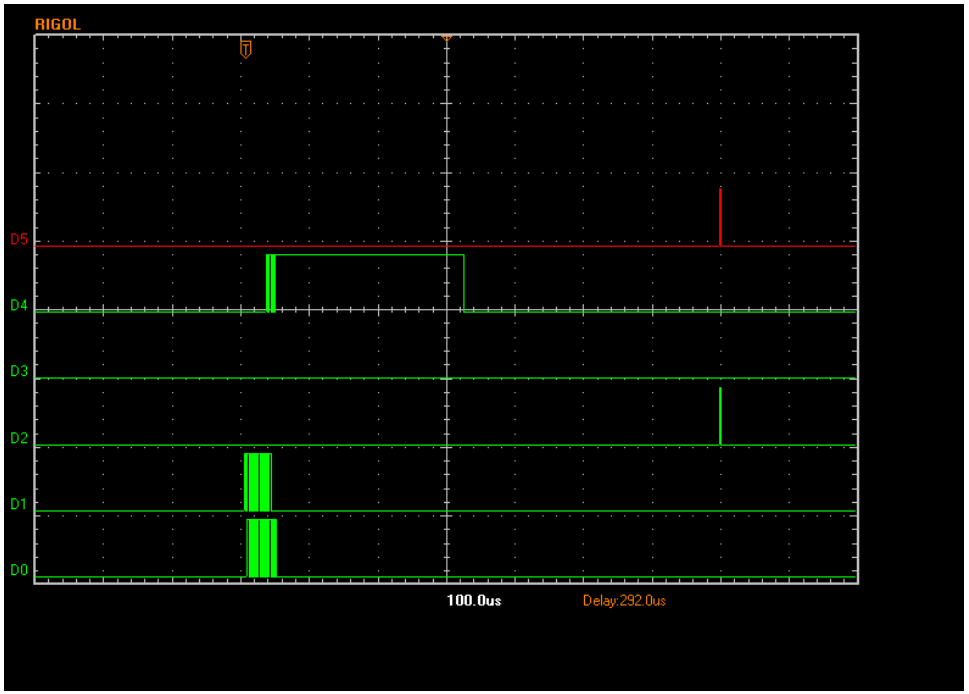
I have tried a few ways to speed it up but have yet to find a good one – anyone with any good ideas let me know. I tried using a struct but I can't find a way to do this for example

```
    ' struct USBwritebuffer, bytes as char[64], dummy as short ' port as char[1], count as char[1], bytes as char[60]
    'USBwritebuffer.bytes[1].struct = 161
```

Liberty basic throws an error as it won't let me index into the array – I guess I am doing something stupid…

I also tried forming the array in the dll with multiple DLL calls but that was slow too.

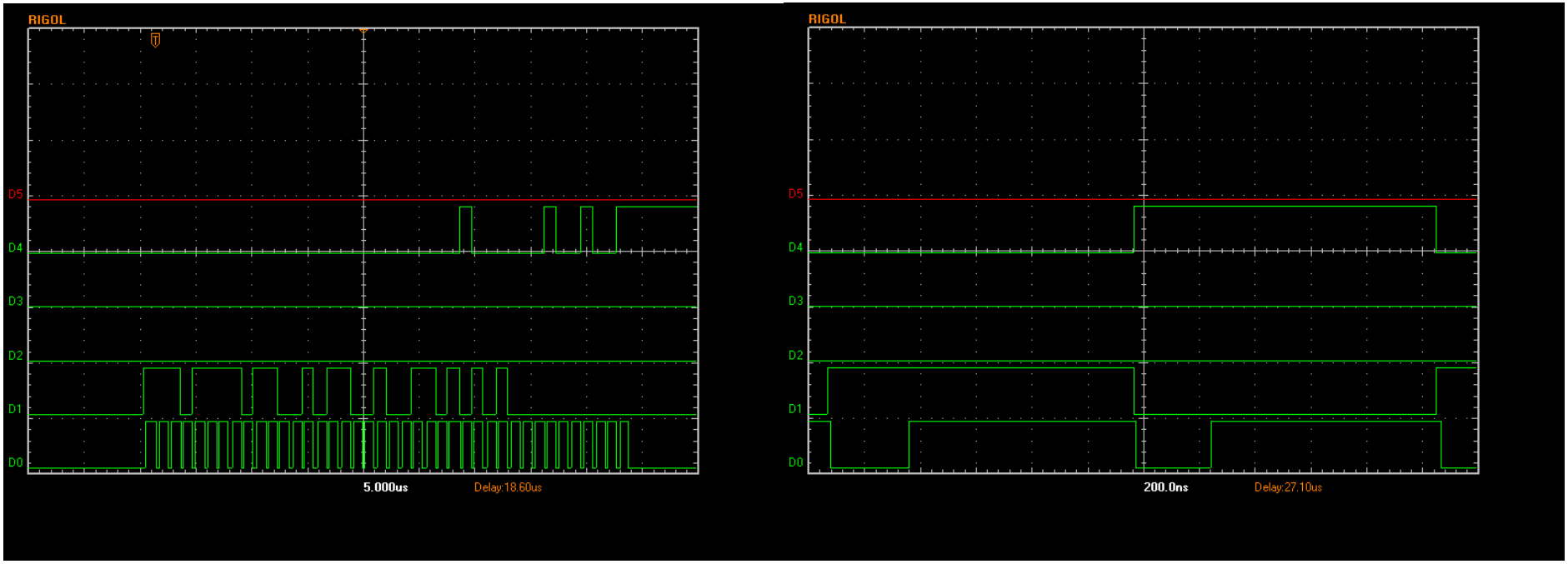
Anyway, here a trace – the time delay before the first write is not shown – at the moment it seems to take 5 ms for Liberty Basic to form the string in the first part of the function ! There has GOT to be a better way.



The traces are:
D0 – clock
D1 – DDS1 data
D2 – DDS1 FQUD
D4 – PLL1 data
D5 – PPL1 LE

The sequence after the 3rd division is the first USB command – clocking the 40 bits of data. The event just after the centre line is the second USB event – writing filtbank – this seems pretty pointless so I will comment it out, The two pulses are the final event – pulsing FQUD and LE

Zooming in on the clocking of the data we get this (the second trace zooms in on part of the write)


RIGOL
D5
D4
D3
D2
D1
D0
5.000us
Delay:18.60us
RIGOL
D5
D4
D3
D2
D1
D0
200.0ns
Delay:27.10us