De USB ACK/NAK handshake

1796	37.311602450	host	4.3	USBLL	3 OUT
1797	37.311605700	host	4.3	USBLL	67 DATA0
1798	37.311651783	4.3	host	USBLL	1 ACK
1799	37.311653783	host	4.3	USBLL	3 OUT
1800	37.311657033	host	0.4.3	USBCOM	17 DATA1
1801	37.311669950	4.3	host	USBLL	1 NAK
1802	37.311671866	host	4.2	USBLL	3 IN

noForth USB ACK/NAK in beeld

Het eerste pakket wordt geaccepteerd. Het tweede pakket wordt geweigerd, omdat onze driver het endpoint nog niet heeft vrij gegeven. De driver op de host geeft ons een kleine 600 MS om te reageren op dat volgende pakket.

1807	37.311706450	host	4.3	USBLL	17 DATA1
1808	37.311719366	4.3	host	USBLL	1 NAK
1809	37.311741950	host	4.2	USBLL	3 IN
1810	37.311745283	4.2	host	USBLL	1 NAK
1811	37.311747200	host	4.3	USBLL	3 OUT
1812	37.311750450	host	4.3	USBLL	17 DATA1
1813	37.311763366	4.3	host	USBLL	1 NAK
1814	37.311785866	host	4.2	USBLL	3 IN
1815	37.311789283	4.2	host	USBLL	1 NAK
1816	37.311791200	host	4.3	USBLL	3 OUT
1817	37.311794450	host	4.3	USBLL	17 DATA1
1818	37.311807366	4.3	host	USBLL	1 NAK

Meer geweigerde pakketten (elke 44µs)

1827	37.311882450	host	4.3	USBLL	17 DATA1
1828	37.311895366	4.3	host	USBLL	1 NAK
1829	37.311959533	host	broadcast	USBLL	3 SOF
1830	37.311962783	host	4.1	USBLL	3 IN
1831	37.311966200	4.1	host	USBLL	1 NAK
1832	37.311968116	host	4.2	USBLL	3 IN
1833	37.311971533	4.2	host	USBLL	1 NAK
1834	37.311973450	host	4.3	USBLL	3 OUT
1835	37.311976700	host	4.3	USBLL	17 DATA1
1836	37.311989616	4.3	host	USBLL	1 ACK

Het 2e datapakket wordt na 332 µs geaccepteerd

Hoe wordt de handshake gebruikt

1. Een pakket wordt ontvangen in een endpoint
2. De CDC driver signaleert dat en de start de afhandeling
3. Test of er voldoende ruimte in de beide ring-buffers
4. Zo ja, dan haalt driver haalt het endpoint leeg en naar 5.
   Zo nee, dan doen we niets, we bewaren wel het signaal
5. Het ontvangen van het volgende pakket wordt toegestaan

Het ontvangen van data pakketten

```
USB-HANDLER  ( -- )
   false ep3 >pid !  ep3 prep-rcv  \ Init. USB receive
   0  begin  endpoints  pause  usb-rx  again ;
```

We initialiseren de PID van endpoint 3 op DATA0 en geven daarna endpoint 3 vrij voor data ontvangst. Nieuwe data ontvangst begint altijd met DATA0. Elk volgend pakket wisselt daarna tussen DATA1 en DATA0 in de data ontvangst lus.

```
: ENDPOINTS  ( 0 -- if )  \ Handle used endpoints
   5011,0058 @                \ Read interrupt flags
   dup 04 and if  zlp>  then  \ EP1 active?
   dup 5011,0058 !            \ Clear active flags
   80 and  or ;               \ EP3 active is bit 7
```

Laat op de stack een bitmasker achter. Dit bit geeft aan, dat er een data pakket ontvangen is op endpoint 3.

```
: USB-RX     ( if0 -- if1 )   \ Restart RX after reception
  dup 80 and if                  \ EP3 active (bit 7)?
     5010,009C c@ >r             \ #Data arrived in EP3
     #L #tx - r@ >               \ Enough space in TX buffer?
     #L #rx - r@ >  and if       \ And next RX packet fits too?
         ep3 >buf @  r@          \ Yes, fill RX buffer
         for  c@+ >rx pause  next  drop
         80 xor  ep3 prep-rcv    \ Done, allow next RX packet
     then  rdrop
  then ;
```

Dit bit op de stack activeert de RX afhandeling, vanaf nu kijkt de driver of er ruimte is om het af te handelen. Zo ja, dan wordt het pakket in de ringbuffer gezet, en endpoint 3 vrij gegeven.

1827	37.311882450	host	4.3	USBLL	17	DATA1
1828	37.311895366	4.3	host	USBLL	1	NAK
1829	37.311959533	host	broadcast	USBLL	3	SOF
1830	37.311962783	host	4.1	USBLL	3	IN
1831	37.311966200	4.1	host	USBLL	1	NAK
1832	37.311968116	host	4.2	USBLL	3	IN
1833	37.311971533	4.2	host	USBLL	1	NAK
1834	37.311973450	host	4.3	USBLL	3	OUT
1835	37.311976700	host	4.3	USBLL	17	DATA1
1836	37.311989616	4.3	host	USBLL	1	ACK

Zoals we in bovenstaand plaatje zien gebruikt onze driver de laag niveau ACK/NAK USB handshake om nieuwe data pakketten toe te laten als noForth er ruimte voor heeft. Het ontvangst bitje (bit 7), blijft net zolang op de stack staan tot noForth ruimte heeft om een volgend pakket te verwerken.

Wanneer gebruik wordt gemaakt van de Windows-driver, bedraagt de beschikbare verwerkingstijd per ontvangen USB-pakket circa 0,6 seconde. Effectief zorgt dit voor een betrouwbare handshake, zolang de host maar niet in één keer een grote hoeveelheid data op de USB-bus gooit. Bestanden kleiner dan 16 kB verlopen onder Linux en Windows doorgaans probleemloos. Maar waarom?

Eindigen met een korte demo

1) Start noForth t solo en open terminal
2) Doe WORDS en CHAPTERS
3) Laad disassembler via drag en drop
   met en zonder regel vertraging...

Nog vragen?