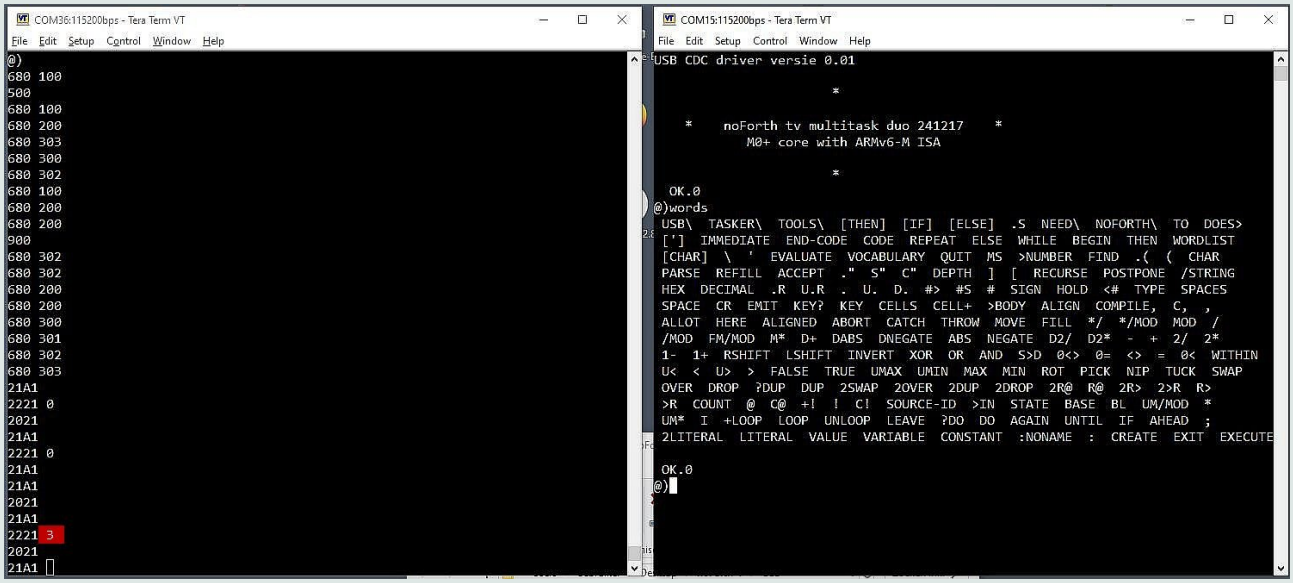
USB op noForth t
(CDC-driver)



De USB startup in beeld

Links start noForth nog in normaal op; met karakter I/O via de RS232 interface. Op de derde regel van onder is de verbinding met de HOST (PC) compleet. Na een korte vertraging gaat de karakter I/O verder via de USB interface.

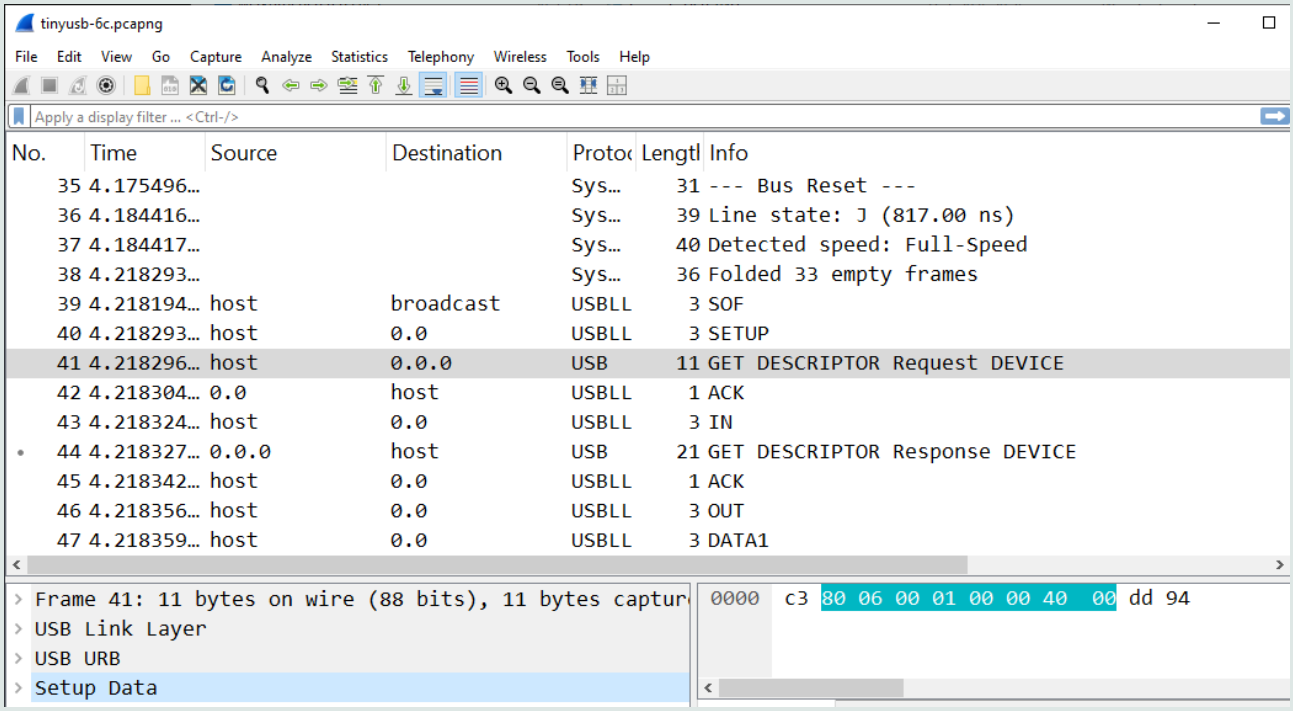
Op de PC verschijnt een nieuw apparaat:

Wat is daar voor nodig:

1. Het initialiseren van de USB hardware op de RP2040.
2. Het ontvangen en beantwoorden van setup pakketten van de host (PC). Deze bepaald hiermee welke driver geladen wordt.
3. Als de host met de setup tevreden is, dan kan het verzenden en ontvangen van data pakketten beginnen.

Dat valt mee zou je zeggen:

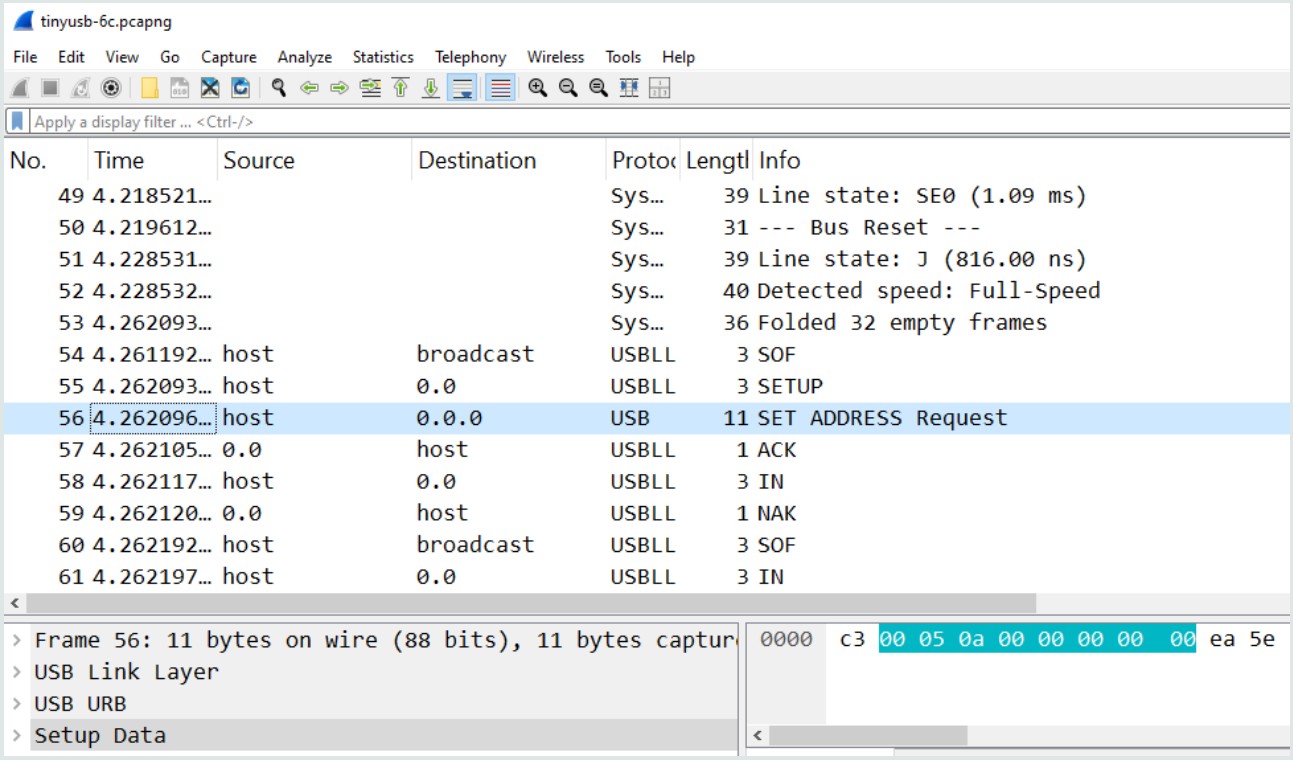
Nou daar zit helaas nog flink wat techniek en protocollen achter. Wat gebeurt er allemaal?



- De host detecteert een nieuw apparaat op de USB-bus en ook welk type USB het is (1.10 of 2.00, etc.)
- De host geeft een USB reset opdracht aan de nieuwe device
  - Nu wordt device 0 om informatie gevraagd (wat ben je?)
  - Onder Windows en Linux krijgen we dan nog een reset
  - Nu krijgen we een adres toegewezen, dat adres moeten we vanaf nu gaan gebruiken
  - Dan wordt nogmaals de device informatie gevraagd
  - Vervolgens de configuratie informatie (hoe zit je in elkaar?)
  - Nu worden (deels optioneel) wat strings gevraagd
    - De taal waarin je praat
    - Het apparaat dat je bent
    - De fabrikant
    - Het versie nummer

- De verzoeken worden nu steeds specifieker
  - De RS232 instellingen
  - Configuratie status
  - De lijn status (kan er gecommuniceerd worden?)
- Bij een ja, kan data heen en weer gezonden worden

Let op: De verzoeken en het aantal ervan is verschillend per OS



Een paar details:

- USB heeft een eigen RAM geheugen met speciale functies.
- Dit 4 kByte grote RAM is zogenaamd dual port RAM, d.w.z. dat Forth en de USB-hardware er tegelijkertijd in kunnen werken.
- Het eerste stuk ervan bevat een 8-bytes buffer for setup verzoeken. Daarachter staan de USB besturingsregisters.
- USB praat via buffers die verbonden zijn met de USB-controller in de RP2040. Deze worden endpoints genoemd.
- Onze USB driver gebruikt nu vier 64 bytes grote endpoint buffers
  - Endpoint-0 voor alles rond de setup
  - Endpoint-1 voor controle data
  - Endpoint-2 voor te verzenden data
  - Endpoint-3 voor te ontvangen data

Wat gebeurt er bij een reset via USB?

```
: BUS-RESET         ( -- )
    80000 'usbr 3050  !               \ SIE_STATUS - BUS_RESET (Clear bit)
    0  'usbr 0  !  0 usb-state !      \ USB-address=0 & USB-state=0
    false 3 pid!  prep-receive        \ Init. receive PID for endpoint-3
    false 2 pid!  false u-config ! ;  \ Init. transmit PID for endpoint-2
```

- Zet de interrupt vlag uit
- Antwoord op setup pakketten naar adres nul
- Zet de USB status op niet afgebouwd
- Initialiseer alvast de data ontvangst- en zend registers
- Zet tenslotte een dummy configuratie

Setup data ontvangen & verzenden:

```
: SETUP>            ( a u -- )      \ Send setup data in a loop
    2000 0 pid!  prepare usb-send   \ PID = DATA1, EP = 0, send buffer
    begin
        begin  pause
        1 'usbr 58 bit** until      \ Packet sent?
        1 'usbr 58 **bis
    cnt @ while                     \ More data to be sent?
        usb-send
    repeat  2000 0 pid!             \ Init. PID
    40 0 usb-receive ;              \ Receive dummy packet
```

De USB setup data wordt altijd uitgewisseld via endpoint-0. Elk pakket wordt gecontroleerd. Het Packet ID (PID) en de checksum worden hiervoor gebruikt.

PID Type	PID Name	PID<3:0>*
Token	OUT	0001b
	IN	1001b
	SOF	0101b
	SETUP	1101b

Voor setup data pakketten begint de PID altijd hoog (DATA1 genaamd). Is een pakket langer dan de buffer grootte dan wordt elke volgend pakket van een ander PID voorzien. Dat wisselt dan tussen DATA1 en DATA0, etc.

Data	DATA0	0011b
	DATA1	1011b
	DATA2	0111b
	MDATA	1111b

Tenslotte ontvangt de setup routine nog een dummy pakket (ZLP). Dit is het antwoord op een door de host correct ontvangen setup pakket.

Data pakketten hebben ook een PID maar die starten met DATA0 en wisselen dan naar DATA1 etc. Er zijn PIDs voor verschillende pakketten, maar dat regelt de USB hardware gelukkig voor ons.

De setup afhandelaar:

```
: HANDLE-REQUESTS   ( -- )
    20000  'usbr 3050  !            \ INTS - SETUP_REQ (Clear bit)
    'dpram 0 h@
    0102 of  zlp>  exit                 then    \ Set communication feature
    0500 of  get-addr  exit             then    \ Set device address
    0680 of  handle-setup  exit         then    \ Basic usb SETUP handler
    0880 of  u-config 2 setup>  exit    then    \ Get configuration
    0900 of  set-config  exit           then    \ Set configuration
    2021 of  line-coding  exit          then    \ Set line coding
    21A1 of  line-coding>  exit         then    \ Get line coding
    2221 of  get-line-state  exit       then    \ Set control line state
    2321 of  zlp>  exit                 then    \ Set line break
    drop  stall> ;                              \ Unkown command send stall
```

Deze routine reset een interrupt bit om aan te geven dat het setup verzoek afgehandeld wordt. Komt er een onbekend verzoek binnen, dan wordt er met een STALL gereageerd. Dit geeft aan dat het om een onbekend verzoek gaat.

Tracer data van de setup procedure in Linux Mint.

De USB afhandelaar draait in twee achtergrond taken van de multitasker. Alle USB zaken worden daar afgehandeld.

```
: HANDLE-USB        ( -- )  \ Handle USB setup & receiving of chars
    start-usb  false usb-state !
    begin
        'usbr 98 @ >r ( ints )
        r@    10 and if  endpoints        then  \    10 = dm 04 bitmask
        r@  1000 and if  bus-reset        then  \  1000 = dm 12 bitmask
        r> 10000 and if  handle-requests  then  \ 10000 = dm 16 bitmask
        pause
    again ;

: DO-TX     ( -- )           \ Handle the transmitting of chars
    true  ( Unconnected )
    begin
        usb-state @ 3 = if         \ Connected?
            dup if                 \ First time, install vectors
                ['] rx?      to 'key?
                ['] usb-key  to 'key
                ['] usb-emit to 'emit
                dup -              \ Vectors installed!
            then
            tx? if      \ EP2 Any chars to send?
                400 'dpram 90 bit** 0= if      \ Yes, previous packet gone?
                    String                     \ Yes, send char string
                    [ buffers #L + 3 cells + ] \ Addr. of char. count
                    literal @  40 min          \ Limit to EP-size
                    2dup bounds ?do            \ Copy present chars
                        tx> i c!
                    loop
                    2 ep!  prepare usb-send    \ Send to host
                then
            then
        then
        pause
    again ;

task: USB1      task: USB2

: USB-ON            ( -- )
    init-buf  ['] handle-usb  usb1 start-task
              ['] do-tx       usb2 start-task
    begin  pause  usb-state @ 3 = until  20 ms ;
```

Nog vragen?