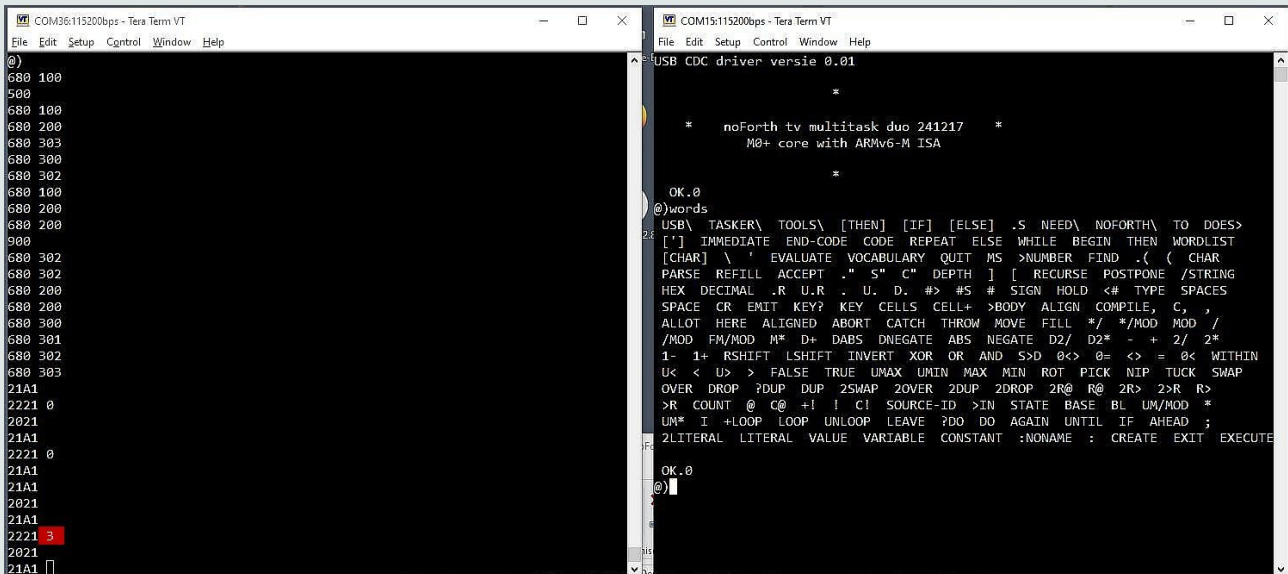


USB op noForth t (CDC-driver)



```
COM36:115200bps - Tera Term VT
File Edit Setup Control Window Help
@)
688 100
500
688 100
688 200
688 303
688 300
688 302
688 100
688 200
688 200
900
688 302
688 302
688 200
688 200
688 300
688 301
688 302
688 303
211A
2221 0
2021
211A
2221 0
211A
211A
2021
211A
2221 3
2021
211A

COM13:115200bps - Tera Term VT
File Edit Setup Control Window Help
USB CDC driver versie 0.01

*
* noForth tv multitask duo 241217 *
* M0+ core with ARMv6-M ISA *
*

OK.0
@)words
USB\ TASKER\ TOOLS\ [THEN] [IF] [ELSE] .S NEED\ NOFORTH\ TO DOES>
['] IMMEDIATE END-CODE CODE REPEAT ELSE WHILE BEGIN THEN WORDLIST
[CHAR] \ ' EVALUATE VOCABULARY QUIT MS >NUMBER FIND .( ( CHAR
PARSE REFILL ACCEPT ." S" C" DEPTH ] [ RECURSE POSTPONE /STRING
HEX DECIMAL .R U.R . U. D. #> #S # SIGN HOLD <# TYPE SPACES
SPACE CR EMIT KEY? KEY CELLS CELL+ >BODY ALIGN COMPILE, C, ,
ALLOT HERE ALIGNED ABORT CATCH THROW MOVE FILL */ */MOD MOD /
/MOD FM/MOD M* D+ DABS DNEGATE ABS NEGATE D2/ D2* - + 2/ 2*
1- 1+ RSHIFT LSHIFT INVERT XOR OR AND S>D 0<> 0= <> = 0< WITHIN
U< < U> > FALSE TRUE UMAX UMIN MAX MIN ROT PICK NIP TUCK SWAP
OVER DROP ?DUP DUP 2SWAP 2OVER 2DUP 2DROP 2R@ R@ 2R> 2R> R>
>R COUNT @ C@ +! ! C! SOURCE-ID >IN STATE BASE BL UM/MOD *
UM* I +LOOP LOOP UNLOOP LEAVE ?DO DO AGAIN UNTIL IF AHEAD ;
2LITERAL LITERAL VALUE VARIABLE CONSTANT ;NONAME ; CREATE EXIT EXECUTE

OK.0
@)

```

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:

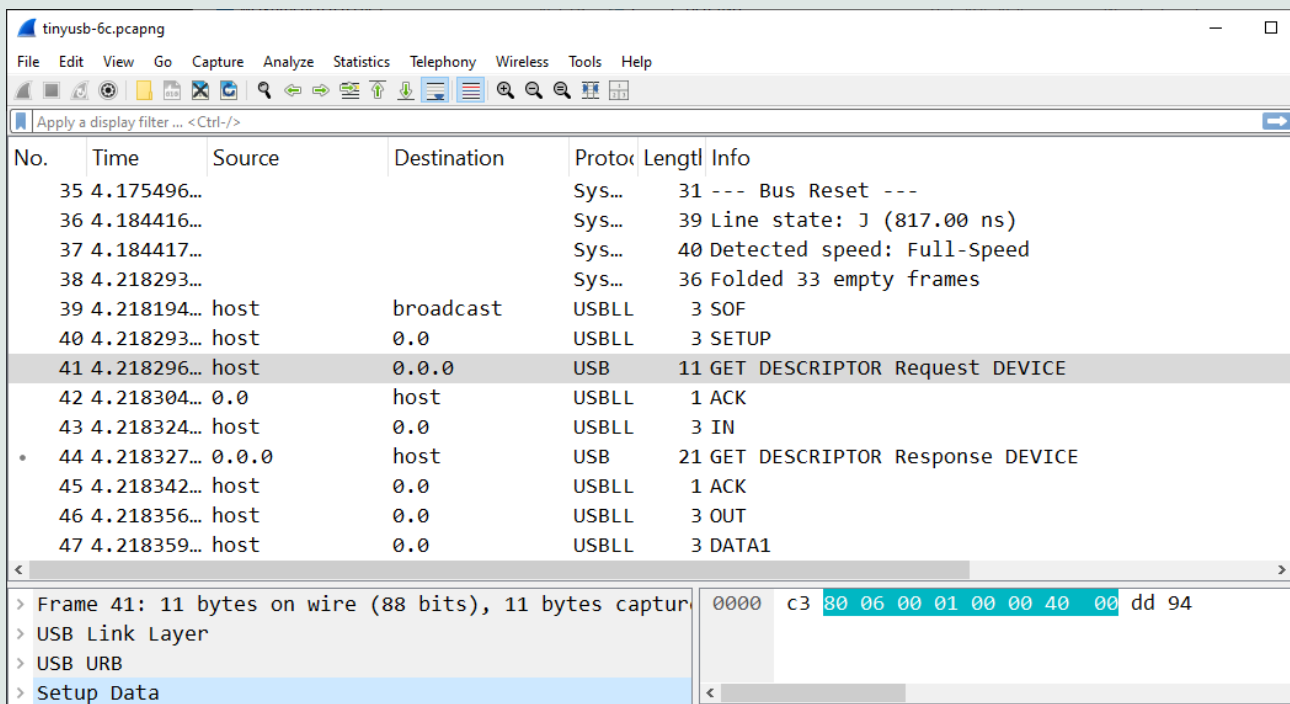


Wat is daar voor nodig:

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?



No.	Time	Source	Destination	Proto	Length	Info
35	4.175496...			Sys...	31	--- Bus Reset ---
36	4.184416...			Sys...	39	Line state: J (817.00 ns)
37	4.184417...			Sys...	40	Detected speed: Full-Speed
38	4.218293...			Sys...	36	Folded 33 empty frames
39	4.218194...	host	broadcast	USBLL	3	SOF
40	4.218293...	host	0.0	USBLL	3	SETUP
41	4.218296...	host	0.0.0	USB	11	GET_DESCRIPTOR Request DEVICE
42	4.218304...	0.0	host	USBLL	1	ACK
43	4.218324...	host	0.0	USBLL	3	IN
44	4.218327...	0.0.0	host	USB	21	GET_DESCRIPTOR Response DEVICE
45	4.218342...	host	0.0	USBLL	1	ACK
46	4.218356...	host	0.0	USBLL	3	OUT
47	4.218359...	host	0.0	USBLL	3	DATA1

> Frame 41: 11 bytes on wire (88 bits), 11 bytes captured on interface (88 bits) on 0
> USB Link Layer
> USB URB
> Setup Data

0000 c3 80 06 00 01 00 00 40 00 dd 94

- 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

The screenshot shows the Wireshark interface with a USB capture. The packet list pane displays the following data:

No.	Time	Source	Destination	Protocol	Length	Info
49	4.218521...			Sys...	39	Line state: SE0 (1.09 ms)
50	4.219612...			Sys...	31	--- Bus Reset ---
51	4.228531...			Sys...	39	Line state: J (816.00 ns)
52	4.228532...			Sys...	40	Detected speed: Full-Speed
53	4.262093...			Sys...	36	Folded 32 empty frames
54	4.261192...	host	broadcast	USBL	3	SOF
55	4.262093...	host	0.0	USBL	3	SETUP
56	4.262096...	host	0.0.0	USB	11	SET ADDRESS Request
57	4.262105...	0.0	host	USBL	1	ACK
58	4.262117...	host	0.0	USBL	3	IN
59	4.262120...	0.0	host	USBL	1	NAK
60	4.262192...	host	broadcast	USBL	3	SOF
61	4.262197...	host	0.0	USBL	3	IN

The packet details pane for frame 56 shows the following structure:

- > Frame 56: 11 bytes on wire (88 bits), 11 bytes captured on interface (88 bits) on 0
- > USB Link Layer
- > USB URB
- > Setup Data

The hex dump for the Setup Data field is: 0000 c3 00 05 0a 00 00 00 00 00 ea 5e

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.

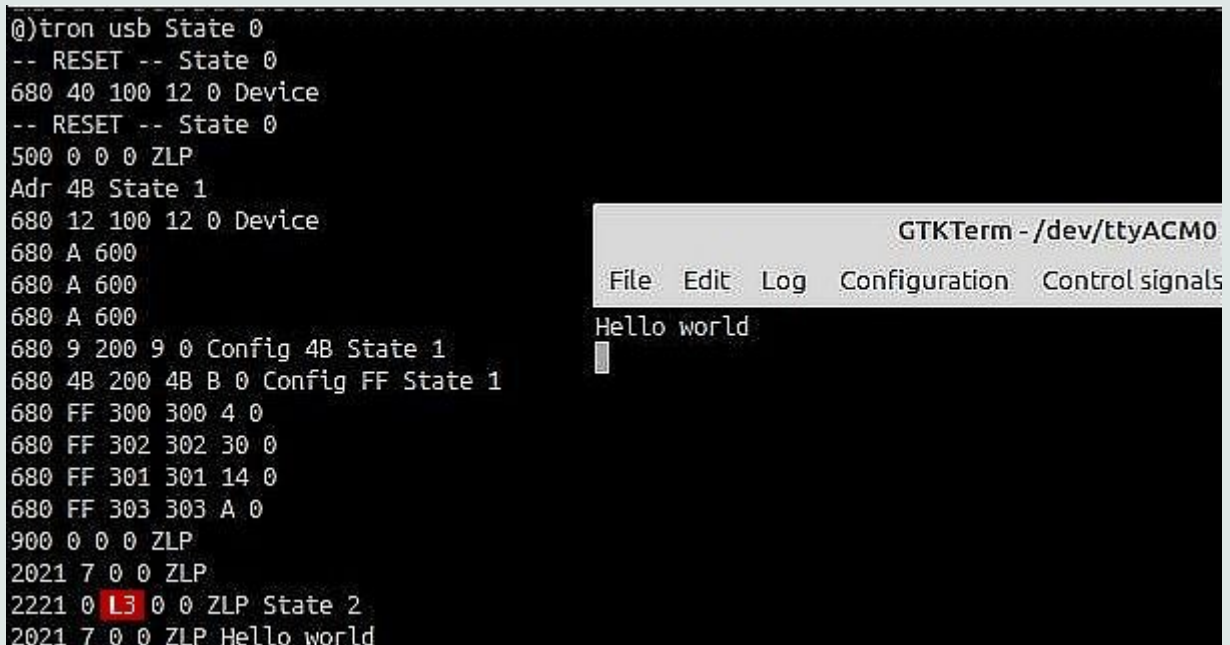
Data pakketten hebben ook een PID maar die starten met DATA0 en wisselen dan naar DATA1 etc. Er zijn PIDs voor verschillende pakketten, maar dat regelt de USB hardware gelukkig voor ons.

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.



```
@)tron usb State 0
-- RESET -- State 0
680 40 100 12 0 Device
-- RESET -- State 0
500 0 0 0 ZLP
Adr 4B State 1
680 12 100 12 0 Device
680 A 600
680 A 600
680 A 600
680 9 200 9 0 Config 4B State 1
680 4B 200 4B B 0 Config FF State 1
680 FF 300 300 4 0
680 FF 302 302 30 0
680 FF 301 301 14 0
680 FF 303 303 A 0
900 0 0 0 ZLP
2021 7 0 0 ZLP
2221 0 L3 0 0 ZLP State 2
2021 7 0 0 ZLP Hello world
```

GTKTerm - /dev/ttyACM0
File Edit Log Configuration Control signals
Hello world

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?