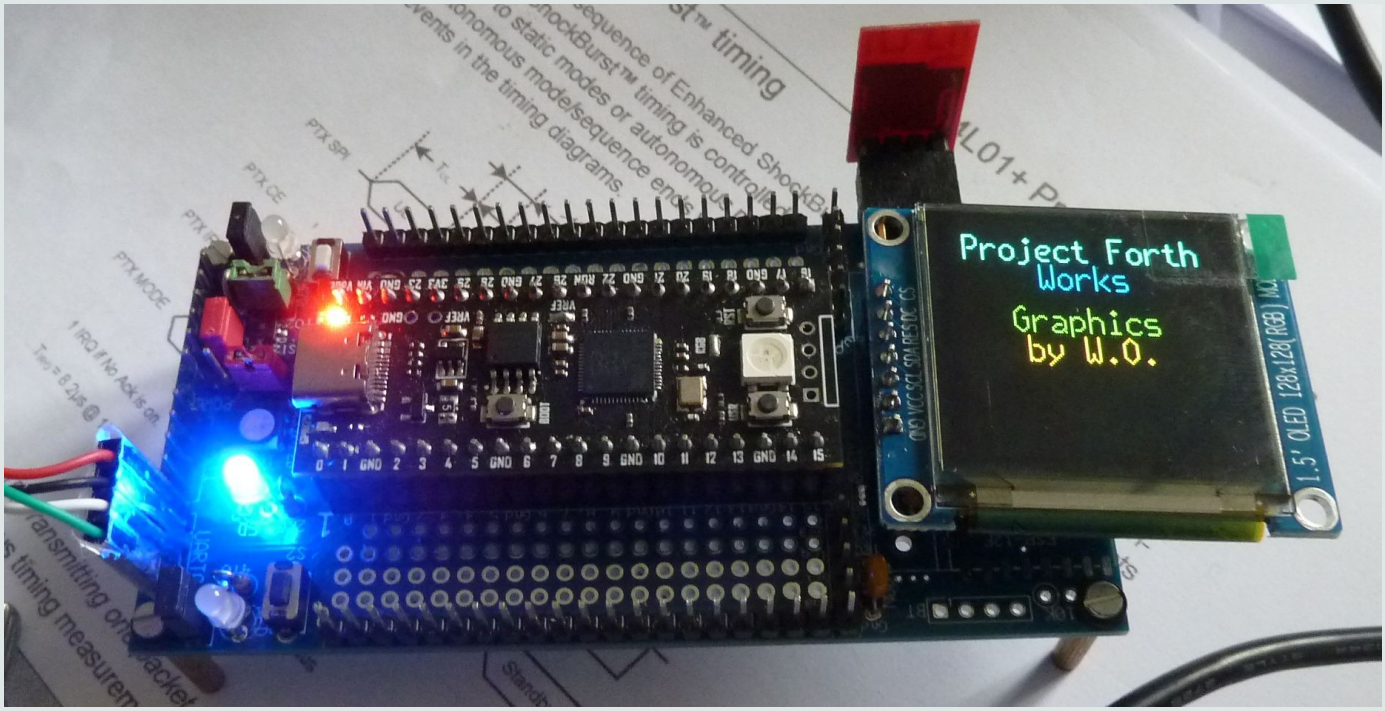
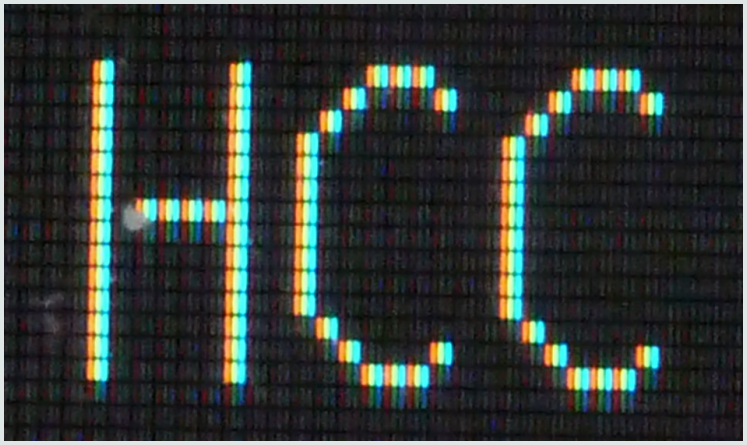
noForth t en de SSD1351

Kleuren OLED besturing (128 x 128 bits)

De SSD1351 driver chip bevat een “full color” driver. Elk pixel wordt bestuurd door een twee of drie bytes code. Deze code bevat de kleurintensiteit van de rode, groene en blauwe LED, die samen één pixel vormen.
De SSD1351 bevat een opdracht, die de pixels op een opgegeven deelgebied van het scherm afdrukt. Daar maakt de noForth driver volop gebruik van.



Kleuren OLED opgestart

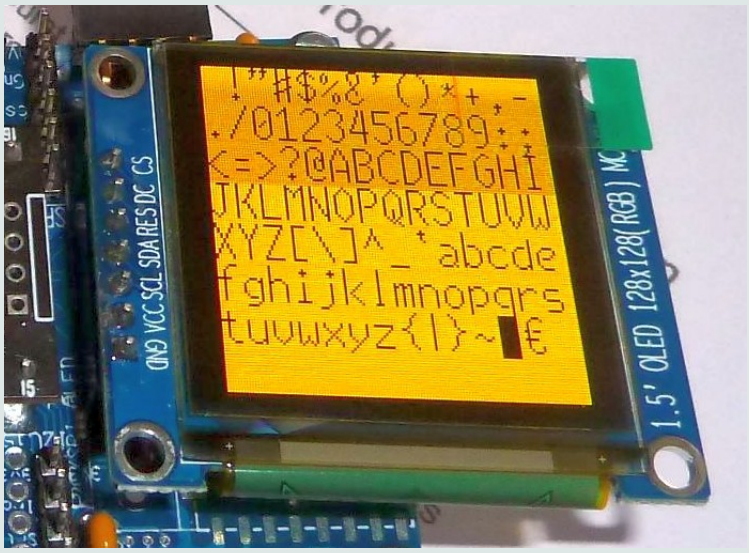


Witte pixels in detail

Acht SSD1351 commando's

Commando	Functie
75	Zet rij adres, begin en einde
15	Zet kolom adres, begin en einde
5C	Start schrijven naar display RAM
AF/AE	Display aan of uit
A6/A7	Kleuren normaal of geïnverteerd
C1	Drie bytes helderheid data voor het hele scherm
--	

Om het scherm op te starten worden nog wat flink wat extra commando's gebruikt.



De grote slanke karakterset afgebeeld met zwarte letters op een oranje achtergrond.

```
|___high____|low_____|
|  00000|000000|00000 |
|__red__|green_|_blue_|
|0...31 |0..63 |0...31|
```

De twee bytes kleurcode 5-bits voor blauw en rood en 6-bits voor groen

Een pixel afdrukken

Om één pixel op het scherm te krijgen moeten we twee bytes versturen. Een nul bij **>PIX** stuurt de ingestelde achtergrond kleur en een één de ingestelde voorgrond kleur.

```
FFFF value LC     \ Letter color, white
0000 value BC     \ Background color, black
: >OL       ( b -- )
    begin  2 4003C00C bit** until  4003C008 !  \ Send byte
    begin  4 4003C00C bit** until  4003C008 @ drop ; \ Receive byte

: {CMD      ( b -- )                  \ Start OLED command string
    0B bitmask gpio-out **bic         \ GPIO11 = DC = 0
    {spi >ol                          \ Start OLED command string
    begin  10 4003C00C bit** 0= until \ SPI no longer busy?
    0B bitmask gpio-out **bis ;       \ GPIO11 = DC = 1
: COLOR>    ( x -- x color )   \ Bit 0 from x to color code
    dup 1 and if  lc exit  then  bc ;
```

Waar komt dat pixel terecht:

```
0 value X   0 value Y       \ Relative X & Y (within window)
0 value 'X  0 value 'Y      \ Window size
0 value WX  0 value WY      \ Window start position

: (XY)      ( xe xb ye yb -- )    \ Set X and Y start & end positions
    75 {cmd  7F and  2>ol  spi}
    15 {cmd  7F and  2>ol  spi} ;
: XY        ( x y -- )      \ Set X and Y window position
    to y  to x   wx 'x +   wx x +   wy 'y +   wy y +  (xy) ;

 : &FILL    ( color +n -- ) \ Fill +n pixels with color
    5C {cmd  for  dup b-b 2>ol  next  spi}  drop ;
```

XY Zet een nieuwe XY-positie om pixels naar af te drukken. Dat kan overal in het actieve window zijn.

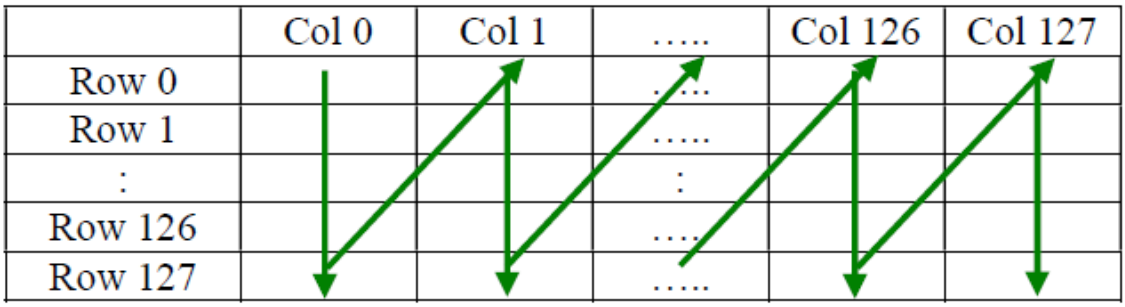
&FILL Druk ‘+n’ pixels met de kleur ‘color’ in het actieve window af.

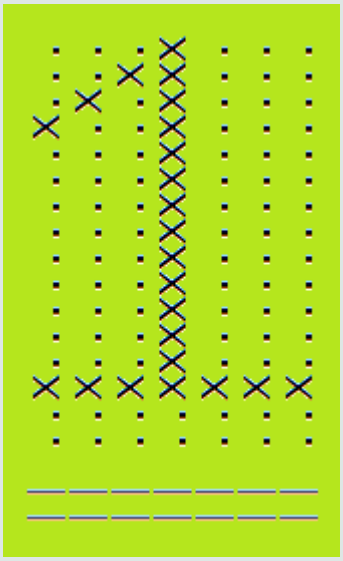
Windows

```
: SLOT   ( 'x 'y -- )        \ Define a window slot at current x y
    y wy + 7F and  75 {cmd  dup >ol  +  >ol  spi}   \ Slot height
    x wx + 7F and  15 {cmd  dup >ol  +  >ol  spi} ; \ Slot width

: BOX    ( 'x 'y x y -- )    \ Open a window at position x y
    to wy  to wx  0 to x  0 to y   1- to 'y  1- to 'x  'x 'y slot ;

: &PAGE  ( -- )            0 0 xy  bc  'x 1+  'y 1+ *  &fill ;
: WINDOW ( 'x 'y x y -- )  box  &page ; \ Remember X Y window start
: WHOLE  ( -- )            80 dup  0 dup window ; \ Use whole screen
```

SLOT Definieert een stukje scherm met de maat 'x 'y
BOX Opent een SLOT op positie XY met de maat 'x 'y
WINDOW Opent een BOX op positie XY en wist de inhoud met &PAGE
&PAGE Maak het scherm schoon met de achtergrond kleur
WHOLE Het volledige scherm is actief en schoongemaakt

Figure 10-3: Address Pointer Movement of Vertical Address Increment Mode





Zo wordt een slot/box/window gevuld met pixel data. Hierdoor kunnen we het opgeslagen karakter in een keer afdrukken. Ook het karakter is in verticale rijen opgeslagen. Linksboven is ook rij-0 en kolom-0.

Het printen van een karakter in pixel kolommen:

```
: .LARGE    ( a +n -- )         \ Display one character in it's own window
    x y 2>r                     \ Remember X Y pos.
    for
        r@ to y  h@+  10 for    \ Read 16-bits column
            onscr? if           \ Valid position?
                color> b-b 2>ol \ Yes, print pixel
            then  2/  incr y    \ To next vertical pixel position
        next  drop  incr x      \ To next column
    next  drop  2r> to y to x ; \ Restore X Y position
```

Tenslotte het printen van een karakter in een beschikbaar blokje (SLOT) op het scherm:

```
: THIN-EMIT ( char -- )
    c>n  'x 1+  x -  dup 9 < if      \ Character does not fit?
        dup 0F &eol                  \ Yes, erase to end Of Line
        0  y 10 +  'y over - 0F < if \ To start of new line
            drop  0
        then  xy
    then  drop  6 0F slot            \ Set character box size
    0E * 'thin +  7 .large           \ Print big char
    x 9 +  y  xy ;                   \ To next char
```

THIN-EMIT Druk het karakter ‘char’ af op positie **XY**.
Past het daar niet, verplaats **XY** dan naar het begin van de volgende regel. Druk ‘char’ nu af in het opgegeven **SLOT** en verplaats tenslotte **XY** een karakter positie verder.

noForth T workshop III

0) Pico aansluiten en terminal starten op 460k8

1) noForth uitbreiden en bewaren (systeem afhankelijk)
    File erbij laden: noForth-T-tools.f
    En de assembler: noForth-T-asm.f
    FREEZE
    WORDS

2) Een tweede systeem bewaren
    Laad: noForth-T-das.f
    FREEZE2
    WORDS
    COLD

3) Wisselen tussen beide bewaarde systemen
    COLD2
    WORDS

    COLD
    WORDS

4) De configuratie bekijken en veranderen
    Laad de file: print-cfg.f
    Kloksnelheid omlaag, laad file: config-T-460k8-12MHz.f
    .CFG

    Kloksnelheid omhoog, laad file: config-T-460k8-132MHz.f
    .CFG
    Kloksnelheid standaard: COLD

5) De ADC als temperatuur meter
    Laad voorbeeld: temperature.f
    TEMP-DEMO
    Voeg aan de file uit een ijkroutine toe, om de
    value #CAL met een correcte waarde te vullen.

6) Hardware interrupts, sluit een snoertje aan op GPIO2
   het programma start onmiddellijk.
    Laad voorbeeld: hw-interrupt-1a.f

7) Gebruik van de ingebouwde timer
    Laad de file: Timer-examples.f
    200 alarm00 (Gebruikt armed vlag)
    200 alarm-0 (Gebruikt interrupt vlag)

    one ch A .ch two ch B .ch many (Interval timers)

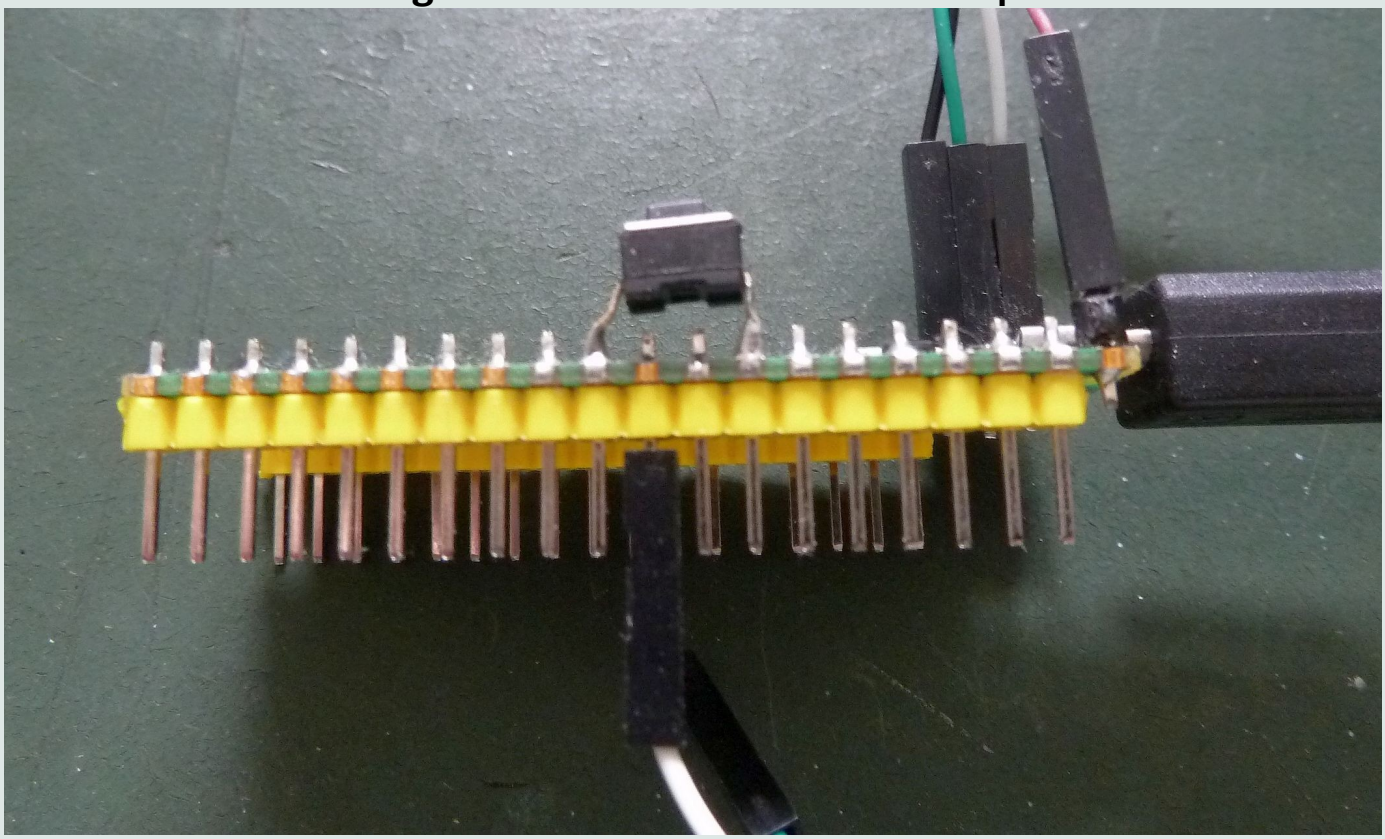
    start-alarm0 (Timer interrupt)

8) Extra seriële poort maken met de PIO

Vorige code er uit: ASM
Laad de file: PIO-assembler.f
Laad de file: PIO-disassembler.f
FREEZE
Selecteer de folder: PIO examples
Laad de file: uart-0.f

Sluit GPIO26 en GND een tweede RS232 aan
en open een tweede terminal op 115K2
ABC stuurt de string ‘ABC’ naar 2e terminal
PICO stuurt de string ‘RP2040’ naar 2e terminal

Verbinding voor UART-0 voorbeeld op GPIO26

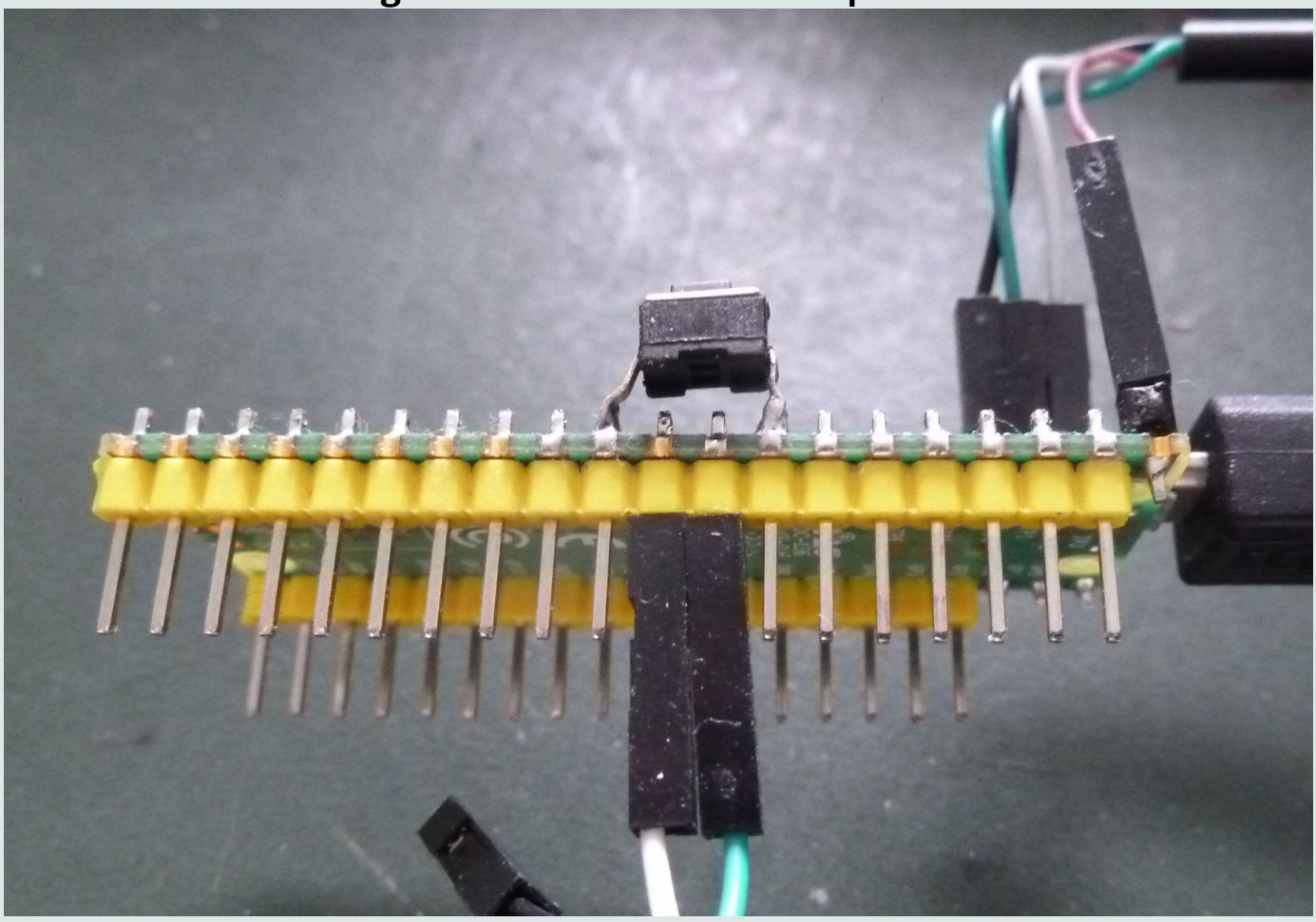


9) Als alles lukt kun je deze nog proberen!
De noForth seriële poort omzetten naar GPIO26 & 27.
Laad de file: uart-4.f
Zet de tweede terminal op 460k8 en verbind GPIO26 en GPIO27 beiden met de tweede RS232 kabel

Probeer het eerst uit door PICO uit te voeren
Krijg je RP2040 in de 2e terminal, dan werkt het.

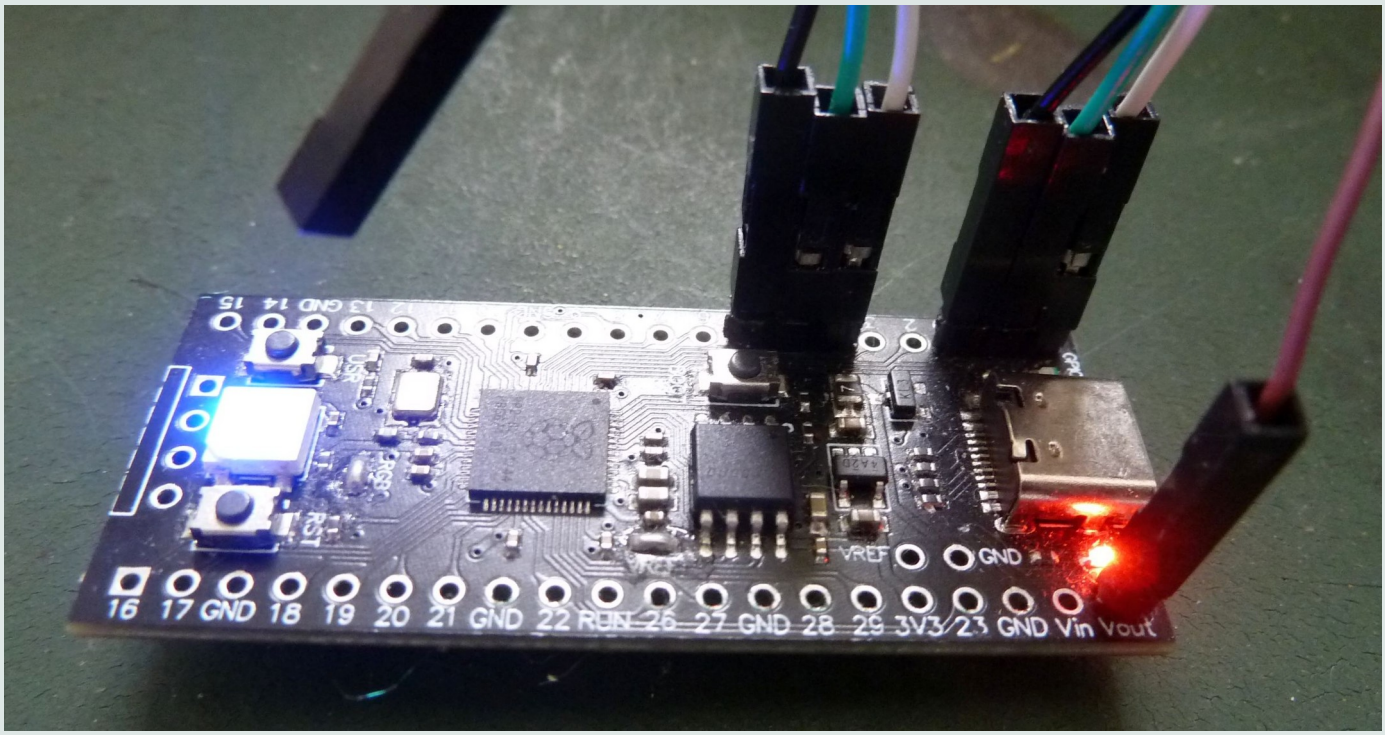
Type nu ALT in en noForth werkt in de 2e terminal!

Verbinding alternatieve UART op GPIO26 & 27



10) Gebruik van eerder gegenereerde PIO drivers.
    PIO-assembler er uit: ASM\
    Laad de file: mini-PIO.f
    Laad een file uit de folder: PIO-ready
1) Flash-1.f   (Simpele LED flitser)
2) Flash-2.f   (Bestuurbare LED flitser)
3) Multiflash.f (WS2812, alleen op Chinese kloon)

WS2812 PIO driver op Chinese Pico kloon



11) WS2812 op de Pico-kit gebruiken:
    De code mini-PIO.f er uit: ASM\
    Laad de file: PIO-assembler.f
    Laad de file: PIO-disassembler.f
    FREEZE

    Laad file uit:..\Pico-kit-examples\PIO-examples
    1) WS2812-P2.f   (Duo WS2812 LED der)

12) Nog vragen