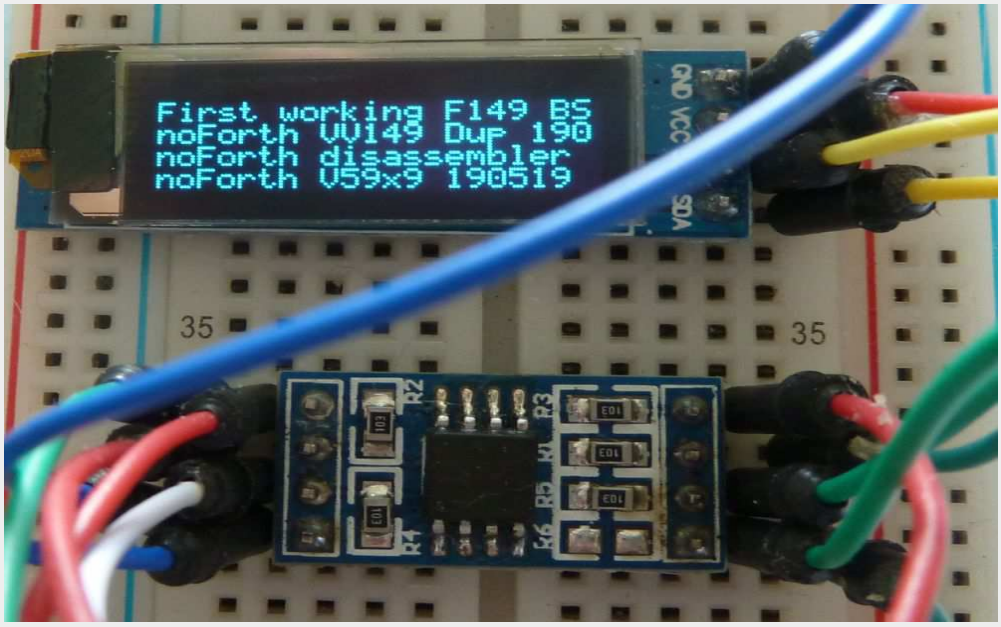
NOF een klein file-OS voor Flash

Voor mijn noForth button met filmpjes en de portable BSL-programmer voor de MSP430, ga ik een 2-MByte SPI Flash geheugen chip gebruiken. Om daar eenvoudig mee te kunnen werken, heb ik in samenspraak met AN een eenvoudig file-systeem bedacht.



De structuur van de 16 MBit Flash chip is bepalend:

- Lezen gaat per byte
- Schrijven gaat in sectoren van 256 bytes
- Wissen gebeurt met sectoren van 4096 bytes, of ... de hele chip in een keer.

Een 2 Mbyte Flash heeft dus 512 pagina's van 4096 bytes. De eerste 10 sectoren noem ik de FET (File Entry Tables). In de FET bevinden zich de FID (File-ID) van 3072 bytes, elk FID-record is 8-bytes. In de resterende 1024 bytes bevindt zich de DID (Directory-ID), elk DID-record is 16 bytes. De resterende 4096 bytes sectoren zijn voor de files. Een file neemt altijd één of meer 4096 byte sectoren in beslag, deze staan altijd achter elkaar!

De FET in plaatjes

FET op 1ste sector (sector 0 t/m 15)	
0000 = FID	0C00 = DID

De indeling van FID

Een FID-record is 4 cellen (8-bytes)				
File	#Sector	Dhash	FHash	Flength
File-1	0010	003F	834F	0010
File-2	0040	0000	753C	0020
Gewist	0000	0000	0000	0000
Vrij	FFFF	FFFF	FFFF	FFFF

De indeling van de DID

DID = 2 bytes voor dir-hash & 14 bytes $			
Dir.	Dir-hash	W-dir-hash	Dir-naam
Dir-1	4E	00	NOFORTH
Dir-2	B1	00	EGEL
Dir-3	3F	00	SOURCE
Vrij	FF	FF	FF FF FF etc.

Het formaat van een File

Eerste 4096 Bytes file-sector	
FIT & XT in 2 cellen	Filenaam = 28 bytes
File data	

Wat is het FIT en XT

Het FIT is het FIle Type byte, NOF kent er nu vier:

```
0 = TI       - TI TXT file
1 = INTELHEX - Intel HEX file
2 = ASCII    - Plain text file
3 = FORTH    - Forth source file
4 = PICTURE  - Bitmap picture(s) file
```

Ruimte genoeg voor een weelde aan file-types. Het XT is een default actie die aan de file toegekend mag worden. Hiermee kan een correcte uitvoerings routine, aan een file gekoppeld worden.

Files staan opgeslagen in blokken van 4096 bytes. Aleen het eerste record bevat een 20 bytes blok met de filenaam, filetype en een default Forth actie in de vorm van een XT.

Wat data over FET

Er is ruimte in het FID voor 384 files. Doordat de records een macht van twee zijn, is de administratie eenvoudig gebleven. Het DID heeft ruimte voor 64 directories. Directory namen mogen meerdere keren voorkomen, zolang ze maar niet in hetzelfde (sub)directory staan! Ook hier is het formaat een macht van twee voor de eenvoud.

De Directory hash in de FID-records bestaan uit 16-bits. Daarin staan de eerste twee hash-bytes uit het DID samen, hiermee is het mogelijk directory namen te hergebruiken. Het NOF systeem zoekt een file op met zowel de twee-bytes directory hash, als de 16-bits file-hash!

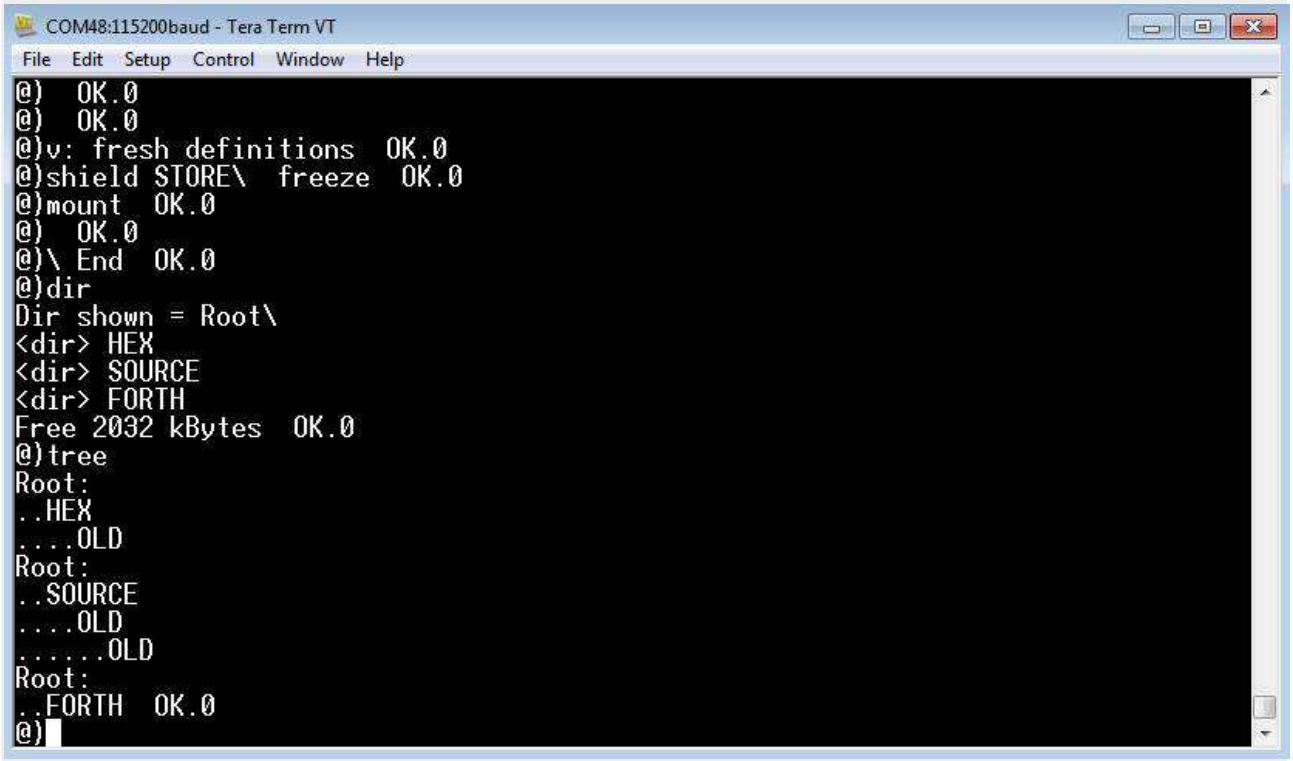
Enkele NOF functies

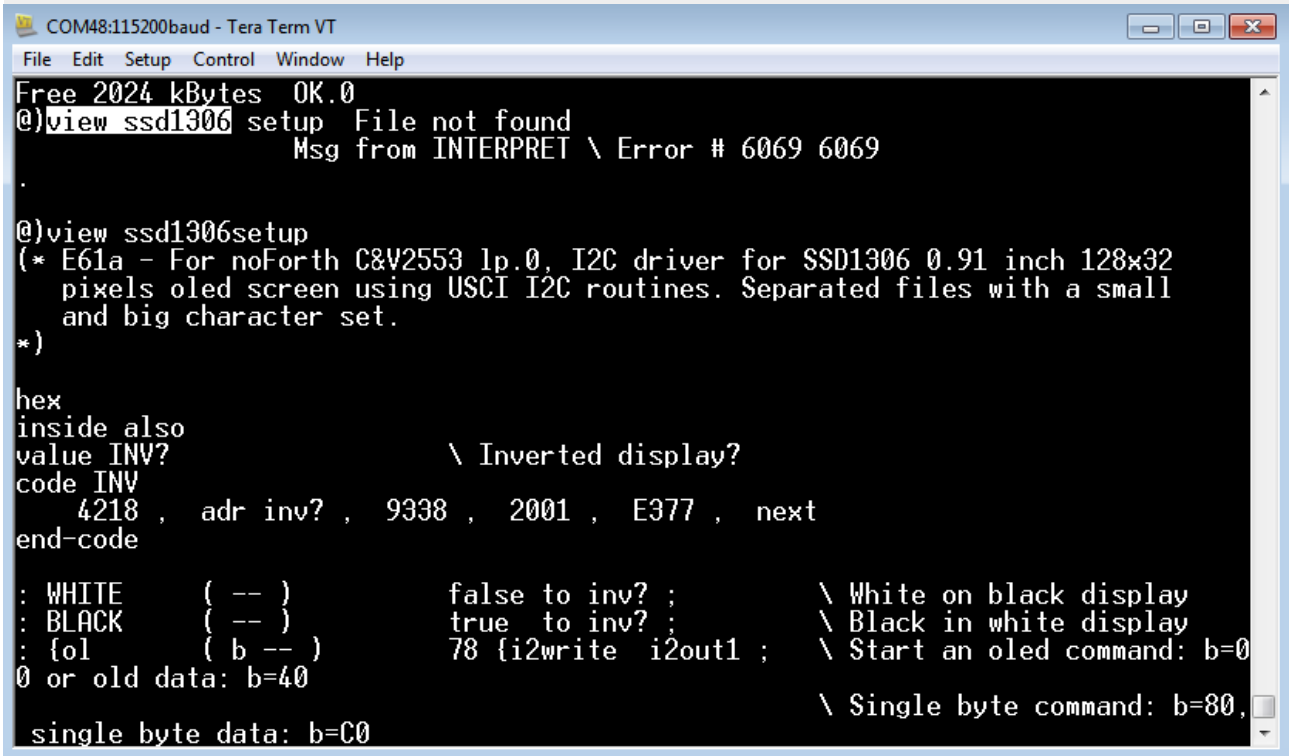
MOUNT	Koppel Flash drive aan NOF
MD	Maak een nieuw directory
CD	Kies een werk- directory
DIR	Toon inhoud van het werk-directory
TREE	Toon de gehele directory boom
VIEW	Display de inhoud van elk filetype
INCLUDE	Laad een noForth source file
PROGRAM	Programmeer een MSP430 met file
DROP-FILES	Wis alle files van disk en vernieuw FET
ERASE-DISK	Wis hele Flash en vernieuw FET
FORTH:	Voeg een Forth source file toe
INTELHEX:	Voeg een IntelHex file toe
PICTURE:	Voeg een bitmap foto/filmpje toe

NOF code voorbeeld

```
0000 constant #FID  \ NOF block with File-ID block (384 files)
0C00 constant #DID  \ and Directory-ID block (64 dirs)

value DID  0A allot \ Hold DIR hash code & nesting
value 'DID          \ Free dir-id pointer
value 'FID          \ Free file-id pointer

: ROOT      ( -- )      0 to DID  0 to DPT ; \ Set root dir

: >DID      ( did -- )  \ Add new top DIR to path
    DPT 0A = ?abort  2 +to DPT
    adr did dup cell+ DPT move  to DID ;

: DID>      ( -- )      \ Remove top DIR from path
    DPT if  adr did dup cell+ swap DPT move -2 +to DPT  then ;

\ Initialise FID and DID to next free location
\ Initialise first free sector location to #SECTOR
: MOUNT     ( -- )          \ Scan FET to intialise disk
    spi-setup  init-rwdata
    #FID  begin
    dup 0 f@ -1 <> while
    dup to TMP 8 + repeat  to 'FID
    #DID  begin
    dup 0 f@ -1 <> while  10 +  repeat  to 'DID
    #FID 'FID <> if     \ FID used?
        TMP 0 f@  TMP 6 + 0 f@ + \ Yes, calc. next free sector
    else  10  then      \ No, set to first free sector
    to #sector  ROOT ;  \ Init. it, start in ROOT

\ Check hash in work DIR, leave true if found, otherwise false
: CHECK-DHASH   ( h -- fl ) \ h = Byte wide hash
    'DID to TMP             \ Save current dir-entry address
    #DID 'DID over -  bounds ?do \ Check directory space
        dup i 0 f@ = if     \ for hash match, file found if so
            i to TMP        \ Save dir address
            drop  true  unloop exit \ Drop hash & leave true
        then
    10 +loop  drop  false ; \ To next DIR entry, or no match
```

NOF aan het werk





NOF include file

```
Forth: Egel-demo
\ Include file for lecture timer

include Asm
include FR5-usci0-i2c
include Ssd1306setup32
include Small-chars
include Thin-chars
include Fat-chars
include Graphic-chars
include Egel

\ End ;;;
```

NOF Intel-Hex file

```
' Program-FR  intelhex: C2x55

:20180000C09D1A9E2E9E449A249EB69D5C9E549BB29428846E8448843E806C9E3E20E02E27
:08182000302F802F070010009B

:0E183200CF070006F400060801020400515 51D

:20FF7E0000137EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFFFD
:20FF9E007EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF73
:20FFBE007EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF53
:20FFDE007EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF7EFF33
:02FFFE00529916

:00000001FF
```