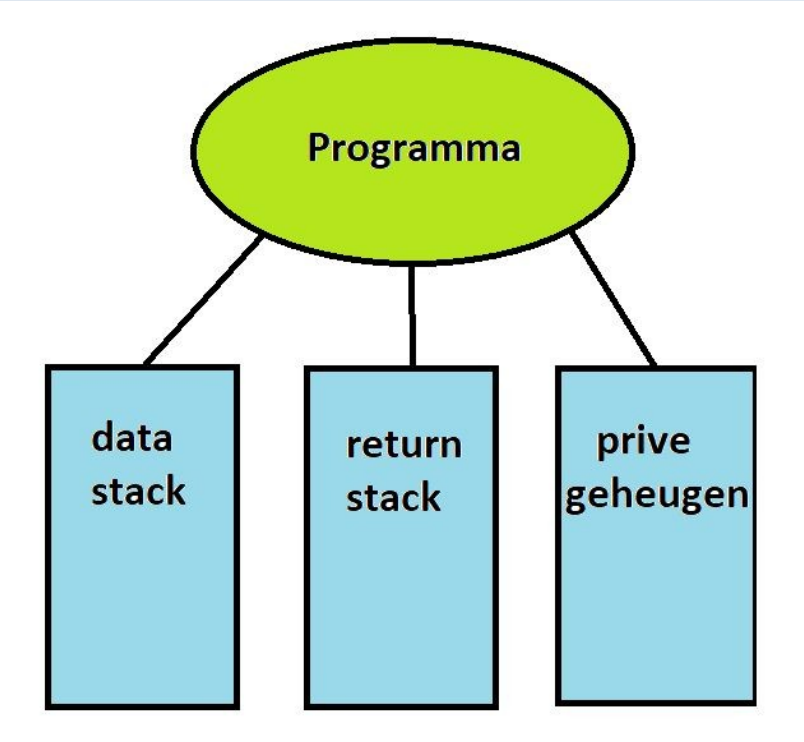
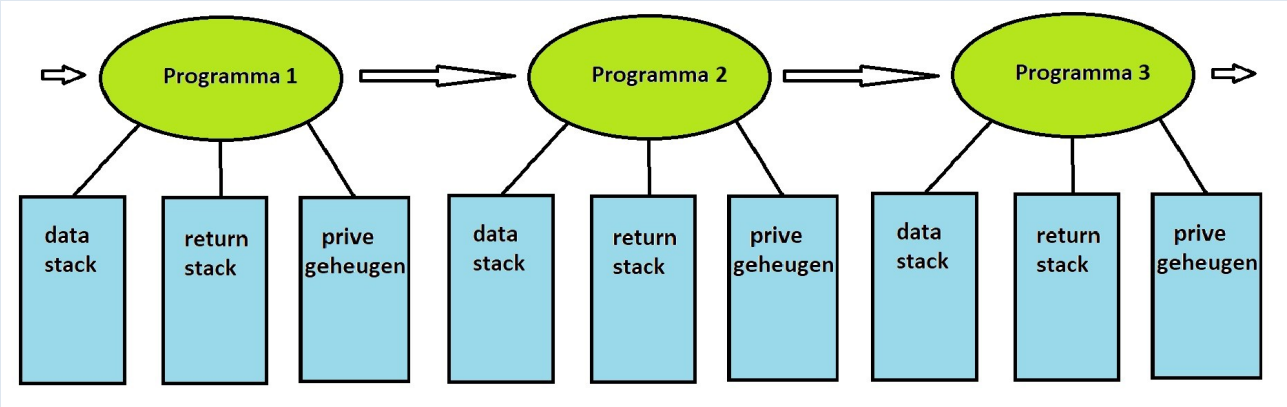
Multitasker voor noForth t

Eigenschappen van een coöperatieve multitasker

- Simpel
- Compact
- Snel
- Taken in een rondlopende lijst
- Je bepaalt zelf de taak wissel (`PAUSE`)
- `Basis woordenset: TASK START-TASK WAKE SLEEP STOP PAUSE`



Een programma in Forth



Drie programmas in Forth

Hoe werkt het mechanisme

TCB	Taak1	Taak2	Taak3
Tstate	-1	-1	0
Link	taak2	taak3	taak1
Mem	adr1	adr2	adr3
RP, etc	ptr1	ptr2	ptr3

noForth t zet meer data in het Taak Controle Blok (TCB), namelijk deze 10 variabelen en value's:

|STATE|LINK|ERR?|TRP|BASE|FP|FLYBUF|FLYBUF/|R0|S0|

In noForth t werkt het anders

TCB	Taak1	Taak2	Taak3
Tstate	taak2	taak1	0
Link	taak2	taak3	taak1
Mem	adr1	adr2	adr3
TRP, etc	ptr1	ptr2	ptr3

PAUSE verzorgt voor de taak wissel

```
code PAUSE  ( -- )   \ 24 machine cycles, is 0.192 microsec. at 125 MHz
    { ip sp tos } push,       \ 4 - Save Forth environment
    sun TP mov,               \ 1 - TCB address to SUN
    day rp mov,               \ 1 - RP to DAY
    day  sun 3 cells #) str, \ 2 - DAY to TRP
    sun  sun ) ldr,           \ 2 - Fetch next link to SUN
    TP sun mov,               \ 1 - Next link address = new TCB
    day  sun 3 cells #) ldr, \ 2 - TRP @ to DAY
    rp day mov,               \ 1 - DAY to RP
    { ip sp tos } pop,        \ 4 - Restore Forth environment
    next,                    \ 6
end-code
```

De return stack initialisatie voor PAUSE

Token	XT van programma
Top van de stack	TOS
Data stack pointer	SP
Instruction pointer	Punt waar van taak gewisseld is

De taak variabele TRP in het TCB wijst naar deze data. Het woord START-TASK verzorgt dit

WAKE hangt een taak (weer) in de lijst

- Doe niets als de taak actief is (niet nul).
- Als de taak link nul is, hang de taak weer in de lijst.
- SLEEP haalt een taak uit de lijst en zet hem op nul.

```
code WAKE   ( task -- )
    main ,
code>                       \ TOS = task
    day  tos ) ldr,         \ Read task state link
    day 0 # cmp,  =? if,    \ Task asleep (zero)?
        w  w ) ldr,         \ Yes, W = operator TCB
        hop  w ) ldr,       \ HOP = operator link contents
        hop  tos ) str,     \ Store operator in state link
        tos  w ) str,       \ Store task in operators state link
    then,
    tos  sp )+ ldr,         \ Replace TOS
    next,
end-code
```

TIDY moet wat meer opruimen

De onderstaande voorbeeld code loopt door de taak lijst.

- De oranje code herstelt een taak met vergeten code
- De groene code herstelt een vergeten taak

```
main  begin                     \ Secure the forgetting of tasks!
    cell+ @ dup main <>         \ Next task is not main task?
while
    dup his r0 @ cell-          \ Yes, get tasks XT address
    dup @  here u> if           \ Is tasks XT deleted?
        over sleep              \ Yes, put task asleep
        ['] noop  over !        \ Replace token with NOOP
    then  drop
    dup here u> if              \ Is task forgotten?
        dup his flybuf @  to xhere  \ Yes, release tasks memory
        dup cell+ @  main cell+ !   \ and remove task from link chain
    then
repeat  drop ;
```

Multitasker gereedschappen

```
.TASKS ( -- ) . . . . . Toon alle taken
LOCK   ( semaphore -- ) . . . . Reserveer semafoor
UNLOCK ( semaphore -- ) . . . . Geef semafoor vrij
TDEPTH ( task -- +n ) . . . . . Stack diepte van task
.STK   ( task -- )  . . . . . Doe .S voor task
PASS   ( x1 .. xn +n task -- )  Zet +n data op de tasks stack
                                als +n=0 dan is de stack leeg
```

```
: TDEPTH    ( task -- +n )
  >r r@ main = if  rdrop depth exit  then \ Main Forth task
  r@ his s0 @              \ Read address stack bottom
  r> his trp @ cell+ @     \ Read current SP from R-stack
  -  cell / ;              \ Calculate depth
```

Waar zit PAUSE in de noForth kern?

KEY? KEY EMIT MS

```
: TKEY?     ( -- f )    key?) dup ?exit  pause ;
: TKEY      ( -- c )    BEGIN  tkey? UNTIL  key) ;
: TEMIT     ( c -- )    emit)  pause ;

: MS  ( ms -- )      \ Millisecond delay
  3E8 *              \ 1000 µs for each step
  40054028 @ >r
  begin
      pause  40054028 @ r@ -  \ µs difference
  over u< 0= until r> 2drop ; \ Done when diff U>= µs
```

Tenslotte een voorbeeld en een demo

We definiëren drie taken en twee routines.

```
hex
0 value CNTR  0 value NUM
: T1  0 to cntr  begin  incr cntr  100 ms  again ;
: T2  1 2 3  0 to num   begin  incr num   200 ms  again ;

task ONE
task TWO
task THREE

: TASKER-ON  ( -- )
    ['] t1    one  start-task
    ['] t2    two  start-task ;

' tasker-on     to app
shield TASKS\  freeze
```

Na het opstarten zien de taak status er zo uit.

```
.tasks
 --- task ------------ link ----- error# -- rp -- action ---
       MAIN  wake   21009A5C         0  2101FDD0  Forth
      THREE  sleep  21009A20         0  20040250  NOOP
        TWO  wake   210099E4         0  20040158  T2
        ONE  wake   21000130         0  20040078  T1
```

Demo op noForth tvm duo

cold .tasks one .stk two .stk
cold2 .tasks one sleep .tasks

noForth t duo geheugen gebruik

- Twee 64 kByte blokken voor elke Forth
- Twee 4 kByte blokken vrij
- XIP RAM 16 kByte
- USB RAM 4 kByte

21000000 - 2101FFFF	Geheugen voor noForth-1
21020000 - 2103FFFF	Geheugen voor noForth-2
20040000 – 20040FFF	Taak geheugen voor noForth-1
20041000 - 20041FFF	Taak geheugen voor noForth-2
15000000 - 15003FFF	16 kByte vrij geheugen (XIP)
50100000 - 5010FFFF	4 kByte USB geheugen

Elke SRAM-bank wordt benaderd via een speciale AHB-Lite arbiter. Dit betekent dat verschillende busmasters parallel toegang hebben tot verschillende SRAM-banken.

(Advanced High-performance Bus)

Er kunnen tot vier 32-bits SRAM-acties per klokcyclus plaatsvinden (één per master).

Geheugengebruik per extra taak		
Getal conversie	Return stack	Data stack
20 bytes	20 cells	10 cells
4000.0000-4000.001F	4000.0020-4000.009F	4000.00A0-4000.00DF