Presentatie HCC!Forth komende zaterdag

Ernst Kouwe gaat ons vertellen hoe Byteforth te draaien is op een server in het netwerk. Op die manier kan men programma's compileren voor de AVR microcontrollers, zonder op de eigen PC of laptop een byteforth compiler te installeren. Indien het lukt om een lokaal netwerk met router te realiseren kan het systeem ook ter plekke gedemonstreerd worden.

Dick Willemse zal zijn ‘guru’ plug meebrengen om te laten bewonderen.

Zaterdag 12 februari op de bekende locatie:
Gebouw naast de Zuiderkapel aan de Boslaan in Bilthoven

10:30 Zaal open en koffie voor vroege vogels.

11:00 "ByteForth in the cloud" door Ernst Kouwe

12:30 Pauze

13:00 Voortgang werkgroepen

15:00 Sluiting.

Tot ziens.

Doelstellingen van de werkgroepen

De doelstellingen van de ByteForth werkgroep:

- Het bestaande ByteForth-systeem overzetten van CHForth naar een modern 32-bits Forth systeem, zodat het onder Windows kan draaien en liefst ook onder Linux. Hierbij kan gebruik gemaakt worden van het werk van de Windows Console werkgroep!
- Het programmeren van chips via USB implementeren.
- Het bestaande systeem upgraden en verder debuggen, een bootloader genereren, hardware vermenigvuldiging gebruiken, etc.
- Documentatie upgraden en ook in de Engelse taal vertalen.

Trekker: Willem Ouwerkerk

De doelstellingen van de Egel werkgroep:

- Het Egel werkboek omwerken voor de moderne AVR processoren.
- Selecteren van een basis processor voor de nieuwe versie.
- Uitbreiden van de bestaande set voorbeelden. b.v. GPS, Bluetooth, Netwerk I/O, bootloader, Ultrasoon-afstandsmeter, voorbeelden van alle interne mogelijkheden van de AVR (I2C, PBM, SPI, etc.)
- Een Engelstalige versie maken.

Trekker: Willem Ouwerkerk

De doelstellingen van de robot (Ushi) werkgroep:

- Ushi nieuw leven inblazen door het apparaat verder te ontwikkelen.
- Er ligt nog een goede ultrasone afstandsmeter die nog uitontwikkeld kan worden.
- Een van onze leden heeft een draadloze interface gemaakt.
- Meer softwarevoorbeelden ontwikkelen, testen en documenteren.
- Het handboek uitbreiden en in het Engels vertalen.

Trekker: Willem Ouwerkerk

De doelstellingen van de arrayForth-werkgroep:

- Het propageren van de arrayForth programmeertaal.
- Het ontwikkelen van programma's voor chips, die met deze taal werken.
- Het werken met de simulator en ontwikkelomgeving.
- Het ontwikkelen van tools om handig met arrayForth te kunnen werken.
- Onderzoek naar methoden en algoritmes om parallel te programmeren op de arrayForth-chips.

Trekker: Leon Konings

Kijk ook op onze website http://www.forth.hcc.nl/w/Werkgroepen/Werkgroepen voor informatie over de werkgroepen.

Mensen die een bijdrage willen leven in één of meerdere werkgroepen kunnen zich melden bij de genoemde trekkers.

Geheugenbeheer in forth met allocate en free

door Albert van der Horst

Inleiding
De Forth wordset MEMORY kent een paar eenvoudig te gebruiken woorden om geheugen aan te vragen (alloceren) en terug te geven: ALLOCATE en FREE. Dit gaat over een implementatie hiervan.

Motivatie
Deze MEMORY wordset heeft een enkele pointer, die door de heap heen loopt, en alloceert daar stukken uit. De pointer wordt nooit teruggezet. De eerste vrije ruimte die groot genoeg is wordt terug gegeven. Volgens de literatuur is dat een redelijke strategie.

Dit programma is klein genoeg om in twee blokken te passen, maar
daar zit minimaal commentaar bij. In tweede instantie heb ik het herontworpen met gebruik van mijn mini object/class wordset. Daar bleek het nut van OO.
Mijn excursie naar OO-land bracht twee bugs aan het licht.
Ik documenteer de code met de Stallman convention, maar dit soort software heeft ook een beschrijjving van het ontwerp nodig.

Dit is ciforth code.
Voor andere Forthen moet je **WANT** zien als een specificatie van wat extra nodig is aan faciliteiten.
Het $ prefix specificeert een hex getal (en is misschien in een andere Forth niet aanwezig).
?ERROR doet een THROW bij een foutconditie.

```
( Copyright{2010}: Albert van der Horst, HCC FIG Holland )
( $Id: memory.txt,v 1.3 2010/06/13 11:18:31 albert Exp $)
```

Ontwerp van een eenvoudige allocator.

De geheugenbuffer ('heap') is een blok aaneensluitend geheugen en bestaat uit een gelinkte lijst van blokken. Elke blok start met een pointer naar het volgende blok, dan een pointer naar het einde van de gealloceerde ruimte in dit blok, of nul als het blok volledig vrij is.
Dit noemen we link pointer en space pointer.
De bruikbare ruimte start twee cellen vanaf het begin van het blok.
Een laatste niet-vrij blok linkt terug naar de start, zodat het een circulaire lijst wordt. Het laatste blok mag onder geen beding vrij gemaakt worden!
Er is een allocatie pointer (adp), die door de circulaire buffer heen loopt.
Nieuwe allocaties worden het eerst geprobeerd in het blok waar adp naar wijst.

Basis operaties.
Een vrij blok wordt gealloceerd door zijn space pointer in te vullen. Dit kan vrije ruimte over laten in dit blok. Een blok kan gerealloceerd worden door de space pointer te wijzigen.
Een blok wordt vrijgemaakt door nul te zetten in zijn space pointer.
Blokken worden aaneengesmeed tot één blok door de link pointer van het eerste blok te vullen met de inhoud van de tweede link-pointer, dit noemen we `merge'.
Dat eerste blok hoeft niet vrij te zijn. Het tweede moet dat wel zijn.

Notitie.
Als de circulaire lijst n blokken heeft, zijn er n allocaties en tot n vrije ruimtes. Vrije ruimtes bestaan uit een of meer vrije blokken of wat er over is van een niet-vrij blok.

Opgetuigde operaties.
In het algemeen moeten operaties op het geheugen eerst controles en zoek- of merge-acties doen voordat ze kunnen worden uitgevoerd.
Bij fouten doen ze een THROW.

Het zoeken naar voldoende ruimte start bij de pointer adp, die niet naar een vrij blok mag wijzen. Als we weer terugkomen bij dat blok, dan is er geen ruimte gevonden en er volgt een THROW.
Voordat we kijken hoeveel ruimte er vrij is in een blok wordt het gemerged met volgende vrije blokken.

Forth operaties.
Een FREE operatie doet een simpele free, zoals beschreven onder basis operaties.

Een ALLOCATE smeedt het blok op adp aaneen met eventuele volgende vrije blokken.
Als er genoeg ruimte resulteert, dan wordt het blok eventueel gesplitst als het niet vrij was. De ruimte wordt gealloceerd van het vrije blok.
Anders zoekt ALLOCATE naar de gewenste grootte, vanaf adp die naar het volgende blok wijst. (Dit volgende blok is nooit vrij, zodat we daar ooit terug moeten komen.)
Het gevonden blok wordt gebruikt zoals eerder aangegeven.
De pointer adp wijst na afloop naar het blok dat teruggeven wordt.

Een RESIZE doet weer een merge van het blok waar adp naar wijst.
Als dit resulteert in genoeg ruimte, dan wordt de grootte bijgesteld. Anders wordt genoeg ruimte gealloceerd, het blok gekopieerd en vrijgegeven.
Zodoende wordt adp slechts gewijzigd als het terplaatse wijzigen onmogelijk was.

ISO Forth operaties
ISO Forth operaties draaien de Forth operaties onder controle van CATCH en THROW.
Als er een fout wordt opgevangen, dan wordt dat als I/O resultaat teruggegeven.
Mogelijke fout situaties:

- De SIZE die nodig is, is niet beschikbaar.
- Een poging een blok vrij te geven dat al vrij was.
- Een pointer doorgeven aan FREE of RESIZE die niet bij een blok hoort

(De tweede en derde zijn fouten in de toepassing door de gebruiker. Geen poging wordt gedaan die af te vangen.)

En dan nu de code, met Engels commentaar:

```
( Copyright{2010}: Albert van der Horst, HCC FIG Holland by GNU Public
License)
( $Id: memory.frt,v 2.9.1.1 2010/06/13 10:09:49 albert Exp $)
WANT NIP
WANT UNUSED
\ ------------- data structure ----------------------------
\ Allocation buffer with circular list of blocks.
CREATE _alloc-buf HERE _ , 0 ,   UNUSED 4 / DUP $1000 MOD -
    ALLOT HERE OVER ! DUP , ,
\ Allocation pointer, address ( like DP), and content (like HERE).
VARIABLE _ADP   _alloc-buf _ADP !    : _AH _ADP @ ;
2 CELLS CONSTANT overhead  \ Administration at start of each block.
\ ------------- basic operations --------------------------
\ Have _ADP point to the next block.
: _bump   _AH @ _ADP ! ;
\ Force SIZE upon the block at ADDRESS.
: _alloc_f   CELL+ DUP >R CELL+ + R> ! ;
\ Free the block at ADDRESS.
: _free   0 SWAP CELL+ ! ;
\ Split the non-free block at ADDRESS.
: _split DUP $@ SWAP @ ALIGNED >R 0 R@ CELL+ ! R@ ! R> SWAP ! ;
\ Merge all free blocks following ADDRESS into address,
\ unless it is the last block.
: _merge DUP @ OVER < IF DROP ELSE DUP >R   BEGIN @ DUP CELL+
    @ UNTIL R> ! THEN ;
\ ------------- information fetches ------------------------
\ For block at ADDRESS return "it IS free"
: _free? CELL+ @ 0= ;
```

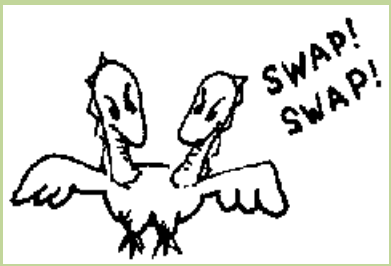
```
\ For block at ADDRESS return what is AVAILABLE to use in total.
: _avail   DUP @ SWAP -  overhead - ;
\ For block at ADDRESS return what AVAILABLE to use remaining.
: _remain   DUP $@ SWAP @ DUP IF - NIP ELSE DROP SWAP - THEN overhead - ;
\ For block at ADDRESS return ALLOCATED size.
: _size_alloc  CELL+ $@ SWAP - ;
\ ------------- adorned operations (they throw) -------------
\ From here all ADDRESSES are user addresses (2 cells ahead.)
\ Search for SIZE from `` _AH '' that must be non-free, leave _AH
\ pointing at space, or throw.
: _search  _AH BEGIN DUP _merge 2DUP _remain > WHILE @
  DUP _AH = 18 ?ERROR .S REPEAT _ADP ! DROP ;
\ Allocate SIZE, return ADDRES.
: _allocate  _AH _merge   DUP _AH _remain > IF _bump DUP _search THEN
  _AH _free? 0= IF _AH _split _bump THEN   _AH _alloc_f
  _AH overhead + ;
\ Move from ADDRESS SIZE bytes to newly allocated address. Return IT.
 : _alloc&move DUP _allocate DUP >R SWAP CMOVE  R> ;
\ Resize at ADDRESS to allocated SIZE. Return ADDRES with size.
: _resize   OVER overhead - >R   R@ _merge   R@ DUP _avail > IF
    _alloc&move R> _free ELSE R> _alloc_f THEN  ;

\ ---------------- ISO Forth operations -------------------------
: ALLOCATE   ['] _allocate CATCH ;
: FREE   overhead - _free 0 ;
: RESIZE  ['] _resize CATCH ;
```

1] Read catch, not CATCH.

Albert van der Horst, UTRECHT,THE NETHERLANDS
Economic growth -- being exponential -- ultimately falters.
albert@spe&ar&c.xs4all.nl &=n http://home.hccnet.nl/a.w.m.van.der.horst

Noot van de redactie:
Dit verhaal en programma van Albert is al eens gepresenteerd op de bijeenkomst en ook in de nieuwsgroep comp.lang.forth.



Ook iets te melden?

Stuur uw ideeen, programma's of projecten naar de redactie, zodat anderen daar ook kennis van kunnen nemen.

Bijdragen liefst per E-mail naar:

f.l.van.der.markt@kader.hcc.nl

website van de HCC!Forth: **http://www.forth.hccnet.nl**

Indien u deze mailing dubbel ontvangt, dan zit uw mailadres in beide verzendlijsten:
* de centrale verzendlijst naar het hccnet.nl mailadres van alle leden van HCC!Forth
* de lijst die samengesteld is op basis van het door u opgegeven mailadres