Forth WORDLISTs im Flash

Einfache Implementierung und null Verwaltungsaufwand durch wid-Tags.

Zwei Beispiele: STM8 eForth u. Mecrisp-Stellaris.

- Inspiriert durch noForth V von Albert Nijhof & Willem Ouwerkerk -

Manfred Mahlow Forth-Tagung 2018

(Kontakt: Vorname PUNKT Nachname AT forth-ev.de)

WORDLISTs

im RAM (klassische Implementierung)

Wörter werden verknüpft.

z. B. implementiert als

```
VARIABLE vl

: WORDLIST ( -- wid )
  HERE DUP 0 , vl @ , vl ! ;
```

(wid ist die Adresse einer Variablen)

im Flash

Wörter werden markiert.

```
: WORDLIST ( — wid )
  HERE 0 , ;
```

(wid ist die Adresse einer Konstanten)

Das Wörterbuch

Eine verkettete Liste

von Wortlisten.

von Wörtern.

Wörter zum Wörterbuch hinzufügen

Ein neues Wort

wird mit der „compilation wordlist“ verknüpft.	wird mit der Wörterbuch-Liste verküpft und mit dem **wid** der „compilation wordlist“ markiert.

Ein Wort im Wörterbuch suchen

Gesucht wird

in den Wortlisten, deren **wid** in der „Search Order“ enthalten ist.	in der Wörterbuch-Liste nach Wörtern, deren Markierung (**wid-Tag**) in der „Search Order“ enthalten ist.

Wörter aus dem Wörterbuch entfernen

z. B. mit MARKER oder SHIELD

Aktualisierung der Liste der Wortlisten und der Wortlisten.	Keine Verwaltung von Wortlisten erforderlich.

Wörter einer Wortliste anzeigen

WORDS zeigt alle Wörter der ersten Wortliste der „Search Order“ an.

Angezeigt werden alle Wörter

der verketteten Liste der Wortliste.	der Wörterbuch-Liste, deren **wid-Tag** mit dem **wid** der Wortliste übereinstimmt.

Beispiel 1: STM8 eForth

- unterstützt nur eine Wortliste auf die die Variable CONTEXT zeigt
- die Wortliste besteht aus zwei verketteten Listen, eine im Flash/NVM und eine im RAM.
- gesucht wird erst im RAM, dann im Flash
- compiliert wird ins **RAM** oder ins Flash/**NVM**
- Struktur der Forth Worte: **|Link|i/c/x/Length|Name|Body|**

  i = immediate c = compile-only x = tag, nicht verwendet

wid-Tags für das STM8 eForth

- Die Wörter der „forth-wordlist“ (wid = 0) werden nicht mit einem wid-Tag markiert.

- Die Wörter aller anderen Wortlisten werden mit dem wid der Wordliste markiert und das tag-Bit x wird gesetzt:

  |wid|Link|i/c/x/Length|Name|Body|

- Auf das Flash/NVM sind wiederholte Schreibzugriffe erlaubt. So ist es möglich, **das STM8 eForth zur Laufzeit zu patchen**:

  - **CURRENT** hinzufügen
  - **CONTEXT** als Array redefinieren
  - **find** und **WORDS** ändern, das tag-Bit auswerten
  - **NAME?** ändern, die „search order“ auswerten
  - **?UNIQUE** ändern, nur im „compilation context“ suchen
  - **$,n** erweitern, wenn CURRENT @ ≠ 0 ein wid-Tag Feld anlegen und das tag-Bit x setzen

- Der Patch kann mit #require CURRENT geladen werden, wird PERSISTENT gemacht und benötigt 310 Bytes im Flash und 6 Bytes im RAM.

Das STM8 eForth (CORE.ihx) nach dem Patchen *(3961 + 320 Bytes)*:

WORDS
CURRENT CONTEXT find WIPE IRET SAVEC RESET RAM NVM WORDS .S DUMP IMMEDIATE ALLOT VARIABLE CONSTANT CREATE] : ; OVERT ." AFT REPEAT WHILE ELSE THEN IF AGAIN UNTIL BEGIN NEXT FOR LITERAL C, , ' CR [\ (.(? . U. TYPE SPACE KEY DECIMAL HEX FILL CMOVE HERE +! PICK 0= ABS NEGATE NOT 1+ 1- 2+ 2- 2* 2/ */ */MOD M* * UM* / MOD /MOD M/MOD UM/MOD WITHIN MIN MAX < U< = DNEGATE 2DUP ROT ?DUP BL BASE - 0< OR AND XOR + UM+ I OVER SWAP DUP 2DROP DROP NIP >R R@ R> C! C@ ! @ 2@ 2! EXIT EXECUTE EMIT ?KEY 'BOOT COLD ok

- Die Anzahl der Wortlisten in der „search order“ ist auf zwei begrenzt. (Das kann im Quelltext geändert werden. Dann auch NAME? anpassen).

- Nach dem Patchen können Worte wie WORDLIST, VOCABULARY oder VOC mit #require <name> geladen werden, z.B.
  - #require WORDLIST *(+54 Bytes)*

    ```
    WORDS
     WORDLIST ?RAM CURRENT CONTEXT find WIPE IRET SAVEC RESET RAM NVM …
    ```

  - #require VOCABULARY *(+197 Bytes)*

    ```
    WORDS
     ORDER .VOC NVM VOCABULARY DEFINITIONS FORTH CURRENT CONTEXT find …
    ```

  - #require VOC *(+289 Bytes)*

    ```
    WORDS
     NVM VOC VP0 VP FORTH DEFINITIONS CURRENT CONTEXT find WIPE …
    ```

- Die Quelltexte finden sich unter https://github.com/TG9541/stm8ef/releases im aktuellen STM8 eForth Binary Release im Unterverzeichnis lib.

- Bei Verwendung des e4thcom Terminals müssen die Verzeichnisse lib, mcu und out/<Version>/target (z.B. als symbolischer Link) im aktuellen Arbeitsverzeichnis vorhanden sein.

- Getestet wurde der Patch bisher mit den Versionen CORE, MINDEV und STM8S105K4.

Beispiel 2: Mecrisp-Stellaris

- unterstützt nur eine Wortliste, die aus zwei verketteten Listen besteht, eine im RAM und eine im Flash
- compiliert ins RAM (compiletoram) oder ins Flash (compiletoflash)
- sucht
  - im compiletoram Modus erst im RAM, dann im Flash
  - im compiletoflash Modus nur im Flash
- Struktur der Forth Worte:

  |Link|Flags|Length|Name|Body|

wid-Tags für Mecrisp-Stellaris

- Die Wörter einer Wortliste werden mit dem **wid** der Wordliste markiert:

  |wid|Flags|Length|Name|Body|

- Die Wörter des Mecrisp-Kerns werden nicht markiert aber der „forth-wordlist“ zugeordnet.

- Das Flash wird in Mecrisp-Stellaris strikt als nur einmal beschreibbar betrachtet. Ein Patchen ist nicht möglich.

  - alle Wörter, die Header compilieren, müssen redefiniert werden (wid-Tags anlegen)

  - die Suche im Wörterbuch (FIND) und die Anzweige des Wörter-buchs müssen redefiniert werden (wid-Tags auswerten)

- Das Einbinden der WORDLISTs Erweiterung erfolgt über einen Einhängepunkt im Mecrisp-Kern, die Variable HOOK-FIND.

  Der Author des Mecrisp-Stellaris, Mathias Koch, war so freundlich, HOOK-FIND in die -ra Version (ab 2.3.8) aufzunehmen.

  HOOK-FIND, das ursprünglich auf **CORE-FIND** zeigt, wird beim Laden der WORDLISTs Erweiterung auf **find-in-dictionary** gesetzt.

- Der Quelltext für WORDLISTs und für weitere darauf aufbauende Erweiterungen (search-order.txt, vocs.txt und classes.txt) findet sich im Mecrisp-Stellaris Archive im Ordner

  /mecrisp-stellaris-2.x.y/common/experimental/vocs-0.7.0/

  Vor dem Ausprobieren bitte unbedingt die README Datei lesen.

Mecrisp-Stellaris Wörterbuch mit WORDLISTs Erweiterung :

(+3596 Bytes)

```
???
RAM: forth
FLASH: forth
               lfa: 000001EC xt: 00000212 name: --- Mecrisp-Stellaris Core ---
               lfa: 00000624 xt: 00000630 name: 2dup
                                                . . .
               lfa: 00004C74 xt: 00004C86 name: irq-timer2b
               lfa: 00004C9E xt: 00004CBE name: --- Flash Dictionary ---
               lfa: 00005000 xt: 00005010 name: wordlist
wtag: 00005024 lfa: 00005024 xt: 0000503A name: forth-wordlist
wtag: 00005024 lfa: 00005048 xt: 0000505E name: inside-wordlist
wtag: 00005024 lfa: 00005744 xt: 00005758 name: show-wordlist
wtag: 00005024 lfa: 00005A08 xt: 00005A1A name: set-context
wtag: 00005024 lfa: 00005A2C xt: 00005A3E name: get-context
wtag: 00005024 lfa: 00005A4C xt: 00005A5E name: set-current
wtag: 00005024 lfa: 00005A70 xt: 00005A82 name: get-current
wtag: 00005024 lfa: 00005A90 xt: 00005A9C name: wlist
wtag: 00005024 lfa: 00005AB0 xt: 00005ABE name: \wlist
wtag: 00005024 lfa: 00005AD4 xt: 00005ADC name: :
wtag: 00005024 lfa: 00005AEC xt: 00005AFC name: constant
wtag: 00005024 lfa: 00005B0C xt: 00005B1C name: 2constant
```

```
wtag: 00005024 lfa: 00005B2C xt: 00005B3C name: variable
wtag: 00005024 lfa: 00005B4C xt: 00005B5C name: 2variable
wtag: 00005024 lfa: 00005B6C xt: 00005B7C name: nvariable
wtag: 00005024 lfa: 00005B8C xt: 00005B9A name: buffer:
wtag: 00005024 lfa: 00005BAC xt: 00005BBC name: (create)
wtag: 00005024 lfa: 00005BCC xt: 00005BDA name: create
wtag: 00005024 lfa: 00005BEC xt: 00005BFA name: <builds
wtag: 00005024 lfa: 00005C0C xt: 00005C18 name: forth
wtag: 00005024 lfa: 00005C2C xt: 00005C3E name: definitions
wtag: 00005024 lfa: 00005C50 xt: 00005C5C name: order
wtag: 00005024 lfa: 00005C88 xt: 00005C92 name: .?
wtag: 00005024 lfa: 00005D00 xt: 00005D0A name: ??
wtag: 00005024 lfa: 00005D24 xt: 00005D2E name: ???
wtag: 00005024 lfa: 00005D48 xt: 00005D56 name: inside
wtag: 00005024 lfa: 00005D6C xt: 00005D78 name: init
wtag: 00005024 lfa: 00005DF0 xt: 00005E04 name: wordlists.txt
```