Der totale Patch

Dr.-Ing. Egmont Woitzel
Dipl.-Ing. Udo Schütz

FORTecH Software GmbH
Joachim-Jungius-Straße 9
18059 Rostock
Tel. (03 81) 4 65 94 72
Fax (03 81) 4 65 94 71

In einem Standard-Forth-System führt das mehrfache Definieren von Worten zu einem Verdecken der jeweils zeitlich vorher in demselben Vokabular definierten gleichnamigen Worte. Typischerweise meldet sich das System bei so einer Redefinition auch, um die Verwirrung noch in Grenzen zu halten. Das sieht beispielsweise so aus:

```
1 CONSTANT FIRST¬  okay
FIRST .¬ 1  okay
2 CONSTANT FIRST¬ FIRST gab's schon  okay
FIRST .¬ 2  okay
```

Allerdings wirkt diese Überdeckung nur für den Textinterpreter und keinesfalls an den Stellen im Fadencode von `:`-Worten, wo das früher definierte Wort mal benutzt wurde. Diese bleiben von der Überdeckung unbehelligt, der Code für das ältere Wort existiert ja auch noch ganz unverändert. Also passiert zum Beispiel folgendes:

```
1 CONSTANT FIRST¬  okay
: SECOND FIRST . ;¬  okay
FIRST .¬ 1  okay
SECOND¬ 1  okay
2 CONSTANT FIRST↵ FIRST gab's schon  okay
FIRST .¬ 2  okay
SECOND¬ 1  okay
```

Dieses Verhalten ist in der Regel auch erwünscht, man möchte ja den früher definierten Code nicht ständig wieder durcheinanderwirbeln. Aber manchmal wäre es doch schon schön, wenn man bei einer Redefinition das Verhalten der Benutzer der früheren Version gleich mitändern könnte. Besonders häufig passiert das beim Debuggen. Das sitzt man beim Ausprobieren eines Programms, das - guter Forth-Stil - aus vielen kleinen aufeinander aufbauenden Worten bestehen, richtig schön top-down entworfen und bottom-up dahincodiert. Beim Testen hat man auch sein Bestes gegeben, aber dann passiert es doch. Man entdeckt einen Fehler in einem der Worte, die ganz früh geladen werden, weil sie ganz oft benutzt werden. Und dann wünscht man sich den *totalen Patch*. Es wäre so einfach: irgendwie in die Betriebsart *Patchen* schalten, den korrigierten Sourcecode im Editorfenster markieren und diesen per Knopfdruck neu compilieren. Das wäre so viel schneller als das ganze Programm zu `FORGET`ten und alles noch mal neu zu übersetzen.

Irgendwann kommt dann der Punkt, wo man sich sagt, eigentlich könnte man das ja mal probieren - den *totalen Patch*. Auch wenn es nicht ganz fein ist, Zeit sparen kann man wahrscheinlich ja genug damit. Tja und dann genügt nur noch ein kleiner Anstoß und dann probiert man es wirklich. Es ist ja im Prinzip auch alles ganz einfach. Eigentlich genügt es, wenn man während der Definition eines neuen Wortes beim Feststellen einer Überdeckung dafür sorgt, daß die alte Codefeldadresse so verändert wird, daß sie auch das neu zu definierende Wort aufruft. Das geht zum Beispiel mit einem Stück Assemblerprogramm, das so ähnlich wie `EXECUTE` funktioniert. In der ersten Euphorie schätzt man den Aufwand auf höchstens eine Stunde. Allerdings kommt dann wie üblich eins zum anderen und es wird doch ein halber Programmiertag draus. Dann kann man aber stolz mit `+REDEFINE` einen universellen Patcher aktivieren und siehe da:

```
1 CONSTANT FIRST¬  okay
: SECOND FIRST . ;¬  okay
FIRST .¬ 1  okay
SECOND¬ 1  okay
+REDEFINE¬  okay
2 CONSTANT FIRST¬ Codefeld 39414 von FIRST überschrieben  okay
FIRST .¬ 2  okay
SECOND¬ 2  okay
```

Auch `SECOND` bemerkt jetzt die Änderung von `FIRST`!

Wie funktioniert nun diese Zauberei? Da man ganz tief im System herumstöbern muß, wird man den Patcher kaum portabel implementieren können. Der abgedruckte Quelltext dürfte daher ausschließlich auf comFORTH für Windows laufen (die einzige für comFORTH 3.0 nötige Änderung kann man im Sourcecode finden). Allerdings sollten die Prinzipien leicht auf andere Systeme wie F-PC übertragen werden können.

Beginnen wir zunächst damit, uns den Aufbau der noch unberührten Worte `FIRST` und `SECOND` unter comFORTH für Windows anzusehen (vergleiche Bild 1). Als erstes fällt die Trennung in verschiedene Segmente auf. Im Headersegment `HSEG` befinden sich die Köpfe der Worte. Diese beginnen mit zwei Zeigern die von `FORGET` benötigt werden, um Speicher im Fadencode- und Assemblersegment freizugeben. Danach kommt das Linkfeld, das auf den Kopf des Vorgängerwortes in diesem Hashzweig zeigt. (Apropos Hashzweig: Um die Suche von Worten zu beschleunigen, besitzt in comFORTH jedes Vokabular nicht nur eine Linkkette, sondern 8 verschiedene. Zu welcher ein Name gehört, wird aus seiner Länge und dem ersten Zeichen ermittelt.) Es folgen der Name des Wortes als abgezählte Zeichenkette sowie der Zeiger auf das Codefeld. Alter Tradition gemäß dienen die oberen Bits des Längenbytes zur Speicherung besonderer Informationen, richtig benutzt wird aber nur Bit 6 durch `IMMEDIATE`. Bit 7 ist immer auf 1 gesetzt, damit man gegebenenfalls das Wörterbuch auch "von hinten" aufräufeln kann, was allerdings nur funktioniert, wenn ausschließlich 7-Bit-ASCII-Zeichen in den Namen verwendet werden. Bei aufmerksamer Betrachtung von `SECOND` fällt außerdem noch auf, daß comFORTH den erzeugten Code sorgfältig auf Wortgrenzen ausrichtet. So wird zwischen dem letzten Zeichen des Wortnamens und dem Zeiger auf das Codefeld gegebenenfalls ein Leerzeichen eingefügt, so daß (FIND) seine Vergleiche ohne Geschwindigkeitsverlust wortweise ausführen kann.

Das Codefeld sowie alle Parameterfelder befinden sich im Datensegment `DSEG`. Da comFORTH indirekt gefädelten Code benutzt, enthalten die Codefelder einen Zeiger auf den zugehörigen Maschinencode, der im Assemblersegment `ASEG` gespeichert wird. `CONSTANT`en verweisen daher auf do`CONSTANT`, Doppelpunktworte auf do:. Hinter dem Codefeld beginnt dann das Parameterfeld, auf dem bei `FIRST` der Zahlenwert selbst liegt. Abweichend von üblichen Implementierungen enthält das Parameterfeld von Doppelpunktworten nicht den Fadencode, sondern nur einen Zeiger darauf. Um die Auslastung des Speichers zu verbessern, wird nämlich dieser in einem eigenen Instruktionssegment `ISEG` gespeichert. In üblichen Programmen wächst dieses Segment am schnellsten. Durch die Trennung von Daten- und Instruktionssegment kann comFORTH größere Programme übersetzen und ausführen als andere segmentierte 16-Bit-Forth-Systeme, die nur die Köpfe und den Maschinencode auslagern. Ein netter Nebeneffekt dieser Implementation ist, daß Doppelpunktworte und Brodie'sche `DOER` miteinander identisch sind. Dadurch können sie unter Verwendung von `MAKE` einfach geändert werden, was im Quelltext z. B. beim Patch von `HIDE`, `REVEAL` und `IMMEDIATE` ausgenutzt wird.

Für den Speicherzugriff auf die neuen Segmente HSEG, ISEG und ASEG implementiert comFORTH eine ganze Reihe spezialisierter Worte, die mit Ausnahme eines einbuchstabigen Präfix wie ihr Vorbild für das Datensegment benannt sind. So gibt es beispielsweise analog zum `,`(Komma) ein `H,` und ein `I,` und ein `A,` mit denen jeweils ein Wort in den

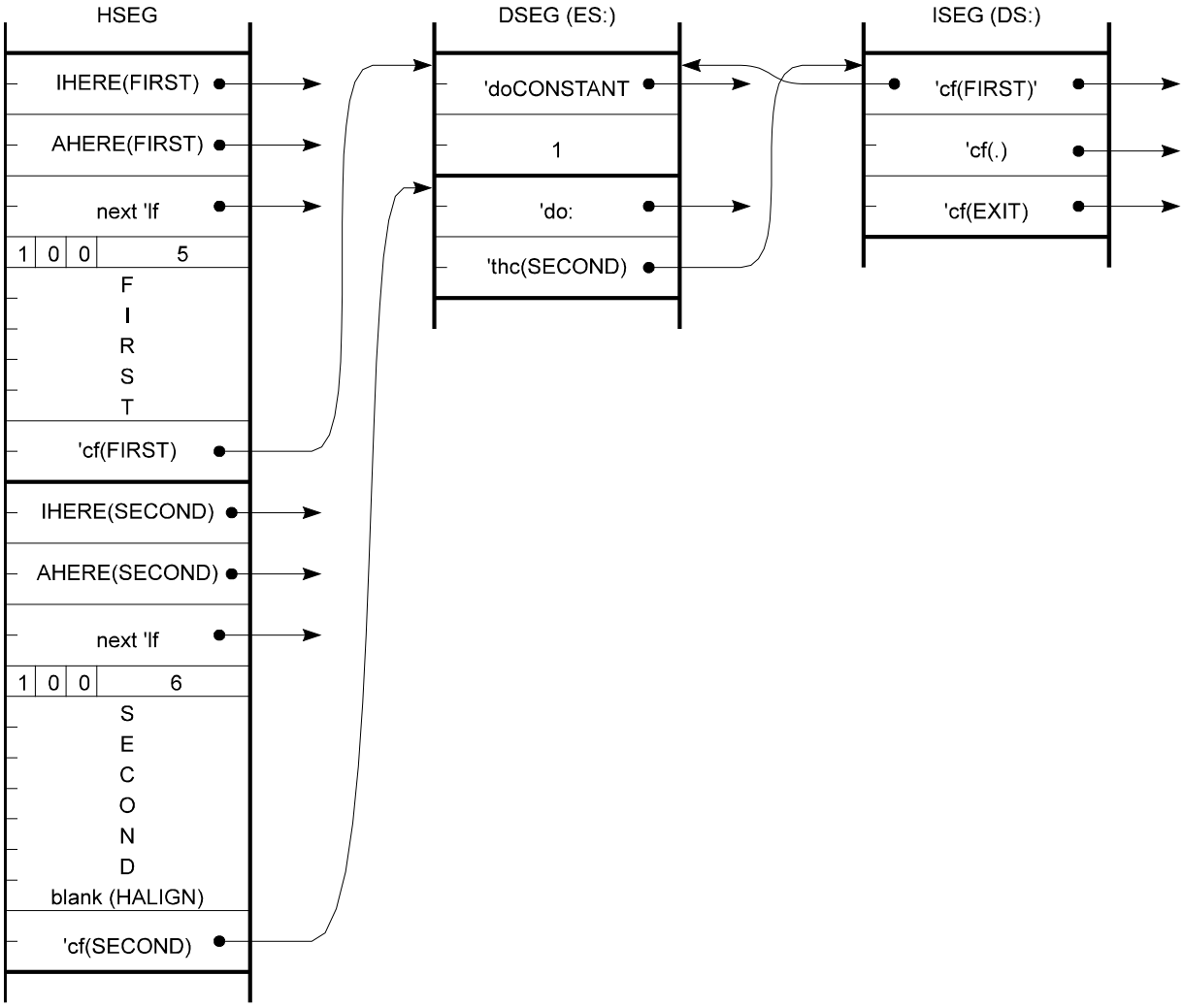

Bild 1: Wörterbuchaufbau vor der Redefinition von `FIRST`

entsprechenden Segmenten gespeichert werden kann, gleiches gilt für @ und ! und eine Reihe weiterer Speicherbefehle.

Mit diesen Informationen versehen sollte das Verständnis des Quelltexts nicht mehr allzu schwer sein. Kernstück des Ganzen ist das Wort PATCHHEADER. Es wird immer dann aufgerufen, wenn beim Bauen eines neuen Wörterbuchkopfs (vgl. die Redefinition von HEADER) eine Überdeckung festgestellt wird, das Redefinieren aktiv ist und das betreffende Wort nicht aus dem Kern stammt. Die letztgenannte Bedingung ist zwar nicht zwingend nötig, schützt den Benutzer aber doch wenigstens ein bißchen vor verstörtem Verhalten des Systems beim Redefinieren von Kernbestandteilen.

Die Implementation von PATCHHEADER selbst ist absolut geradlinig. Die Linkfeldadresse des gefundenen und zu patchenden Wortes wird für (;CODE) in der Variablen LAST gespeichert. Dann wird eine Systemnachricht über die erfolgte Redefinition ausgegeben wobei man ein letztes Mal die gepatchte Codefeldadresse zu Gesicht bekommt. Im Anschluß daran wird mit einer Menge A,'s ein Stück Maschinencode generiert, das wie ein EXECUTE für die neue Codefeldadresse (zu diesem Zeitpunkt das HERE des DSEG) wirkt. Dazu muß diese in comFORTH in die CPU-Register AX und DI geladen werden. Da der Inhalt des Codefelds

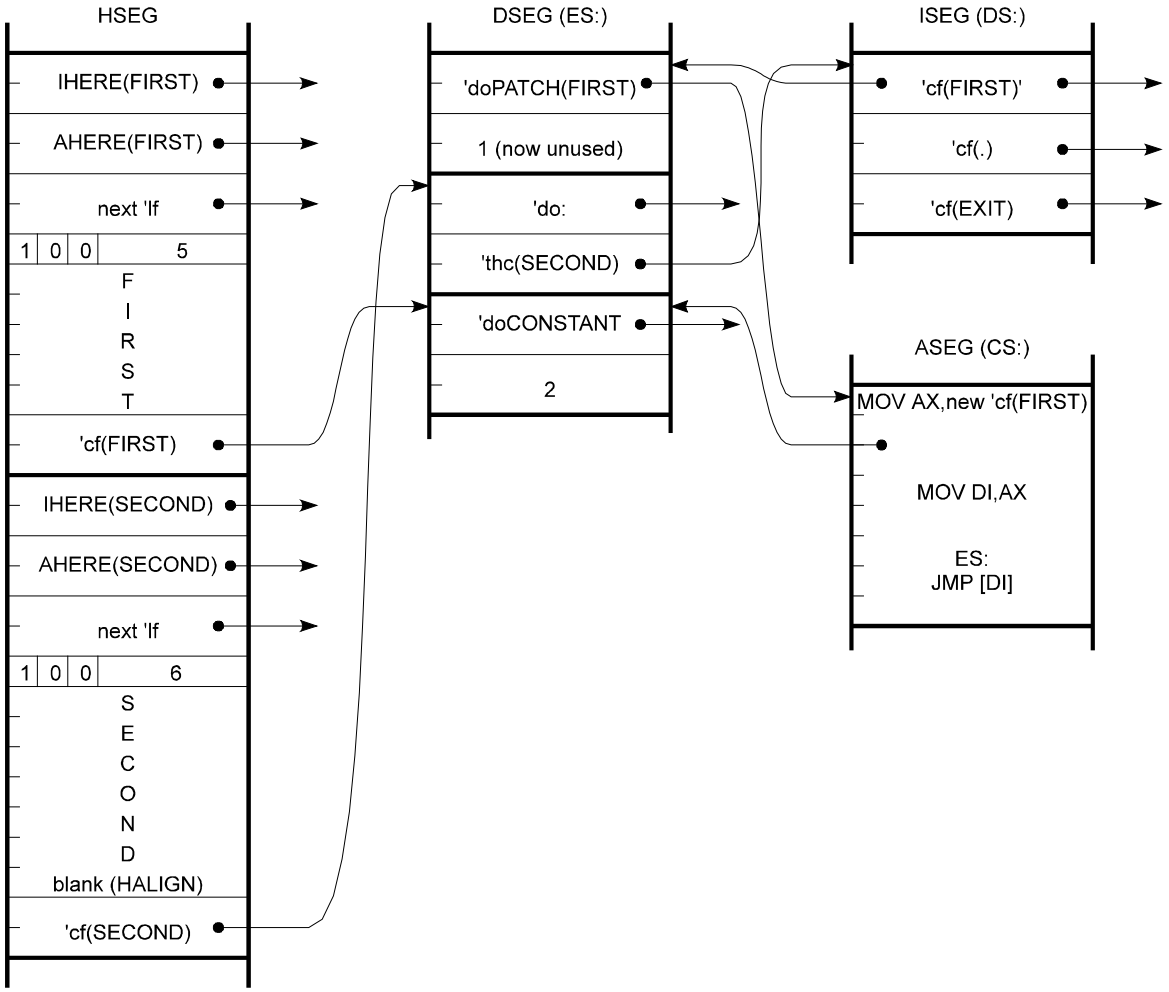

Bild 2: Wörterbuchaufbau nach der Redefinition von FIRST

einen Zeiger auf Maschinencode darstellt, kann anschließend einfach über [ES:DI] gesprungen werden. Die Adresse dieses Codestückchens wird dann nur noch als neue Codeadresse in das alte Codefeld eingetragen. Auf diese Weise brauchen die Definitionsworte von altem und neuem Wort nicht übereinstimmen. Auch das Überladen von CODE-Definitionen, die im DSEG nur das Codefeld und kein Parameterfeld besitzen, ist problemlos möglich. Schließlich und letztendlich muß nur noch die im Kopf gespeicherte Codefeldadresse auf die neue gebogen werden, da sonst (;CODE) bei der Zuweisung des Ausführverhaltens danebenschlägt. Das neue Code- und Parameterfeld von FIRST wird nach dem Abschluß von HEADER durch CREATE bzw. CONSTANT angelegt. Bild 2 zeigt den Zustand des Wörterbuchs nach erfolgter Redefinition. Falls man ein Forth-System ohne Segmentierung verwendet, wird man im Gegensatz zu der implementierten Lösung um ein doppeltes Anlegen des Wörterbuchkopfes wohl nicht herumkommen, der prinzipiellen Realisierbarkeit steht aber nichts im Wege.

Der verbleibende Quelltext ist nicht mehr besonders spannend. Der erste Abschnitt dient nur dem Ein- und Ausschalten des Patch-Modus. Daran folgen die Redefinitionen von `HIDE` und `REVEAL`, welche in comFORTH das Verbergen fehlgeschlagener Definitionen im Wörterbuch steuern und im Fall einer Redefinition unwirksam gemacht werden müssen (das `PREDEFINED` sorgt jeweils für den Aufruf des vorher zugeordneten Codes). Eine nette Überraschung bereitete `NAME>"` (wird von `NAME.` benutzt), das den Wörterbuchbereich ab `HERE` zum Umkopieren benutzt und dabei den von `NAME` gelieferten String zerstörte. Die Redefinition ist etwas seriöser und kopiert den Namen direkt auf den Kettenstapel. Die Definitionen von `MAKEHEADER` und `HEADER` entsprechen mit Ausnahme des Factoring weitestgehend dem Original, wovon man sich leicht mit `SEE HEADER` überzeugen kann. Und so stellt man am Ende fest: Eigentlich ist wirklich alles ganz einfach!

Bei Anwendung des Patchers gibt es verblüffend wenig Probleme. Die im Wörterbuch verbleibenden Restbestände unbenutzten Codes lassen sich leider nicht beseitigen, dazu würde man einen Heap als Wörterbuch benötigen. Der bei jeder Redefinition generierte Maschinencode läßt sich leicht verschmerzen, das Assemblersegment wird ohnehin am wenigsten belastet. Der einzige echte Seiteneffekt wurde bei `WORDS` beobachtet. Da sich WORDS bei der Auflistreihenfolge der Worte zwischen den verschiedenen Hash-Zweigen nach der Codefeldadresse richtet, "springen" redefinierte Worte ab und zu ein Stück "nach vorne". Aber deswegen auch noch `WORDS` zu patchen, war uns dann doch zuviel.

Viel Spaß beim Patchen!

PS: Praktiziert werden diese Techniken vermutlich schon lange, zumindestens im System HOLON von Wolf Weijgaard muß ein ähnliches Verfahren implementiert sein.

Der Quelltext des universellen Patchers:

```
\ Universeller Patcher
\ E. Woitzel, U. Schütz
\ FORTecH Software GmbH 1994

ONLY STRING ALSO DICTIONARY ALSO COMFORTH ALSO FORTH DEFINITIONS

\ --- Zustandsverwaltung

2VARIABLE 'REDEFINE? ( ps: ===> addr )( die erste Zelle enthält ?t, wenn )
        ( Definitionen gleichen Namens hart redefiniert werden dürfen )
        ( die zweite Zelle enthält ?t, wenn das tatsächlich geschah )

: +REDEFINE ( ps: ===> )( schaltet in harten Redefinemodus )
        'REDEFINE? ON ;

: -REDEFINE ( ps: ===> )( schaltet in harten Redefinemodus )
        'REDEFINE? OFF ;

: REDEFINED ( ps: ===> )( markiert letzte Definition als Redefinition )
        'REDEFINE? 2+ OFF ;

: DEFINED ( ps: ===> )( markiert letzte Definition als Neudefinition )
        'REDEFINE? 2+ ON ;

: DEFINED? ( ps: ===> ? )( ?t, wenn letztes Wort eine Neudefinition war )
        'REDEFINE? 2+ @ ;

-REDEFINE       \ erst mal ganz normal bleiben
DEFINED         \ und auf Neudefinitionen stellen


\ --- HIDE und REVEAL-Verwaltung robust machen

MAKE HIDE ( ps: ===> )( nur bei Neudefinition Hiden )
        DEFINED? IF  PREDEFINED  THEN ;

MAKE REVEAL ( ps: ===> )( nur bei Neudefinition Revealen )
        DEFINED? IF  PREDEFINED  THEN ;


\ --- Seiteneffekte zu NAME vermeiden

: LSTR@ ( ps: sel off len ===> )( "s: ===> s )( auf Kettenstapel holen )
        255 UMIN DUP 1+ NEGATE "P+!    \ freihalten
        "P@ C!                         \ Länge
        DSEG "P@ COUNT LCMOVE ;

MAKE NAME>" ( ps: 'nf ===> )( "s: ===> name )( keine Funktionsänderung )
        ( es werden nur Seiteneffekte via HERE vermieden )
        HSEG SWAP HCOUNT 31 AND LSTR@ ;
```

```
\ --- Behandlung neuer Köpfe

HEX
: PATCHHEADER ( ps: 'nf ===> )( altes Wort umpatchen )
        DUP N>LINK LAST !       \ 'lf für (;CODE) setzen
        DUP DUP HC@ 2DUP        \ Längenbyte lesen
        BF AND HSEG ROT LC!     \ IMMEDIATE-Bit zurücknehmen
        1F AND                  \ Namenslänge bestimmen
        1+ + HALIGN             \ Adresse von 'cf
        DUP H@                  \ alte 'cf bestimmen
        MOOD>R >ERROR           \ ausführlich anmaulen
        ." Codefeld " U.        \ alles in der Fehlerfarbe
        ." von " SWAP .NAME
        ." überschrieben " R>MOOD
        AHERE                   \ special EXECUTE generieren
        B8 AC,  HERE A,         \ MOV AX,'cf
\       comFORTH für Windows Beta II: Forth-DSEG steht in ES:
        F88B A, 26 AC, 25FF A, \ MOV DI,AX  JMP [ES:DI]
\       comFORTH 3.0 unter MS-DOS: Forth-DSEG steht in SS:
\       F88B A, 36 AC, 25FF A, \ MOV DI,AX  JMP [SS:DI]
        OVER H@ !               \ 'ca in altem cf patchen
        HERE SWAP H!            \ Zeiger im Header auf neues cf biegen
        REDEFINED ;             \ HIDE und REVEAL beide Augen zudrücken

: MAKEHEADER ( ps: 'hash 'str ===> )( neuen Kopf bauen )
        HALIGN IHERE H, AHERE H,        \ für 'FORGET'
        HHERE DUP LAST !                \ 'lf für HIDE und REVEAL
        2 HALLOT                        \ lf reservieren
        SWAP DUP C@                     \ Namenslänge bestimmen
        1+ HHERE SWAP >HCMOVE           \ Name kopieren
        HHERE CACHE>                    \ Cacheplatz auspusten
        SWAP HLINK                      \ Linkfeld einbinden
        80 HHERE DUP HC@ 1+ HALLOT      \ Namensplatz reservieren
        HCSET                           \ und Traverse-Bit setzen
        HALIGN                          \ ausrichten
        ALIGN HERE H,                   \ Zeiger auf Codefeld
        DEFINED ;                       \ HIDE und REVEAL aktivieren
DECIMAL

\ --- im Kernel herumbiegen, Achtung: hier nicht Modulkomprimieren!!!

MAKE HEADER  ( ps: ===> )( ib: name )( baut oder patcht Wörterbucheintrag )
        CURRENT @ CURRENT 2+ !                  \ Vokabular für HIDE/REVEAL
        BL NAME                                 \ neuen Namen lesen
        CURRENT @ +HASH 2DUP @ (FIND)           \ in CURRENT suchen
        IF      \ im aktuellen Vokabular gefunden
                DUP HFENCE @ U>                 \ Finger weg vom Kern
                'REDEFINE? @ 0<> AND
                IF      \ es soll redefiniert werden
                        PATCHHEADER             \ alten Kopf verbiegen
                        2DROP EXIT              \ aufräumen und verschwinden
                THEN    \ ps: 'str 'hash 'nf
                MOOD>R >ERROR DUP .NAME R>MOOD\ 'gab's schon' Meldung
                1 MESSAGE
        THEN    \ ps: 'str 'hash 'str|'nf
        DROP SWAP                               \ ps: 'hash 'str
        MAKEHEADER ;                            \ neuen Kopf anlegen
```