Definition von Anwenderworten in Forth

CLAUS KÜHNEL

Für die Klasse spartanisch ausgerüsteter Computer, wie Einplatinencomputer, Einchipcomputer und andere Minimalsysteme, stellt Forth eine ideale Programmierumgebung dar. Die Eigenschaften von Forth bedingen aber einen gleichzeitigen Einsatz in komfortablen Minicomputern. In diesem Beitrag wird die Definition von Anwenderworten in Forth beschrieben, auf die Unterschiede zwischen Worten vom Typ Secondary und Primitive Word wird eingegangen.

In der DDR stehen mehrere Systeme zur Arbeit mit Forth bereit. Genannt werden sollen an dieser Stelle die Systeme ComForth und PopForth der Wilhelm-Pieck-Universität Rostock für Mikrocomputer unter dem Betriebssystem CP/M (SCP), das Forthsystem „Fichte" für die konfigurierbare Datenstation K 1715 und das KC-Forth für die Kleincomputer aus Mühlhausen. Mit der Verfügbarkeit des Forthmoduls M 026 für diese Kleincomputer steht breiten Anwenderkreisen ein dieser Rechentechnik angepaßtes Forth zur Verfügung [1].

Zur Lösung eines Anwenderproblems wird, basierend auf dem Instruktionssatz einer Programmiersprache, eine Folge von Anweisungen notiert, die ihrerseits das eigentliche Programm bilden. In Forth stehen sogenannte Worte zur Verfügung, mit deren Hilfe übergeordnete Worte definiert werden können. Das Programmieren in Forth besteht damit aus einer permanenten Erweiterung des vom betreffenden System angebotenen Wortegrundstocks. Mit der Definition von Anwenderworten befaßt sich der folgende Beitrag anhand einer Erweiterung für doppelt genaue Zahlen.

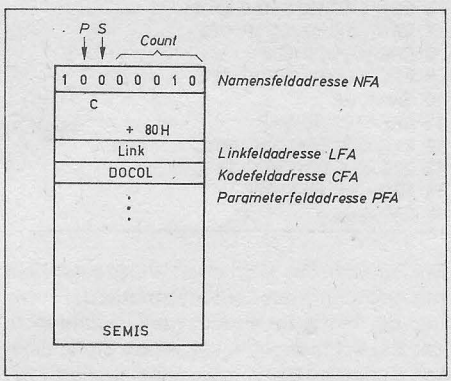
Aufbau von Forth-Worten

Die wichtigsten Worte in Forth sind das sogenannte Secondary und das Primitive Word. Es gibt weitere Worttypen, die für die folgenden Betrachtungen aber nicht benötigt werden. Die Forth-Worte sind im Speicher des Computers, im Dictionary, in der im Bild 1 gezeigten Weise abgelegt.

Jedes Forth-Wort wird durch ein Namensfeld (Header) eingeleitet, dessen erstes Byte Informationen zur Länge des Namens (Count), über die Fertigstellung des Wortes (Smudge) sowie zum hier nicht betrachteten Wortstatus (Precedence) enthält. An dieses Byte schließt sich in Richtung höherer Adressen der Name des Wortes in ASCII-Darstellung an, wobei das letzte Zeichen durch ein gesetztes MSB gekennzeichnet wird.

Das folgende Linkfeld enthält die Adresse des vorher definierten Wortes. Damit ergibt sich eine über das Linkfeld verkettete Liste von Forth-Worten im Speicher.

Dem Linkfeld folgt das Kodefeld. Bei einem Wort vom Typ Secondary steht hier die Kodeadresse einer mit DOCOL (Do the Colon Definition) bezeichneten Routine, die für die Abarbeitung einer Colondefinition verantwortlich ist. Im Falle eines Wortes vom Typ Primitive Word enthält das Kodefeld direkt



Aufbau eines Forth-Wortes vom Typ Secondary im Speicher

die Adresse des ausführbaren Maschinenkodes. Die Kodeadresse gibt damit gleichzeitig Auskunft über den Typ des vorliegenden Forth-Wortes (Secondary, Primitive Word, Konstante, Variable, User-Variable).

An das Kodefeld schließt sich das sogenannte Parameterfeld an. Bei einem Secondary-Wort stehen an dieser Stelle die Kodefeldadressen der in der Colondefinition enthaltenen Forth-Worte, die damit aufrufbar werden. Im Parameterfeld des Worttyps Primitive Word steht im allgemeinen der direkt ausführbare Maschinenkode, der durch einen Sprung zur abschließenden NEXT-Routine beendet werden muß.

Tafel 1: Hexadezimalausdruck eines Speicherauszuges

```
998 40 DUMP
 998 85 41 4C 4C 4F D4 89  9  .ALLOT. .
 9A0 75  7 95  8 DC  6 B5  5 U . . I . 5 .
 9A8 81 AC 98  9 75  7 90  9 . , . . U . . .
 9B0  3  8 A0  9 32  7 B5  5 . .  . 2 . 5 .
 9B8 82 43 AC A8  9 75  7 90 .C , ( . U . .
 9C0  9 FB  7 A0  9 41  7 B5 .  .  . A . 5
 9C8  5 81 AD B8  9 CF  9 D1 . . 8 . 0 . 0
 9D0 EI  A7 ED 52 C3 BF  2 81 A'MRC? . .
OK
2BE 20 DUMP
 2BE D5 E5 A  3 6F A  3 67 UE . . O . . G
 2C6 5E 23 56 EB E9 83 4C 49 ^#VKI .LT
 2CE D4  0  0 D3  2  A  3 6F T . . S . . O
 2D6  A  3 67 C3 BF  2 87 45 . . GC? . .E
OK
```

Anhand des in Tafel 1 gezeigten Hexadezimalausdrucks des Speichers (Dictionary) sollen die Aussagen verdeutlicht werden. Die Namensfelder der enthaltenen Forth-Worte wurden eingerahmt. Zu finden sind im Hexadezimalausdruck die Worte ALLOT; , (Komma); C, ; – (Minus) und LIT. Werden die Worte C, und – (Minus) betrachtet, lassen sich folgende Feststellungen treffen. Das an der Adresse 09B8H beginnende Wort C, weist mit seiner Linkadresse auf das Namensfeld des Wortes , (Komma). Im Kodefeld steht die Adresse 0775H , die Kodeadresse der Routine DOCOL. Damit handelt es sich bei dem Wort , (Komma) um ein Wort vom Typ Secondary. Mit dem Wort T

```
: T 2+ NFA ID. ;
```

lassen sich die Bestandteile dieses Secondary-Wortes auflisten zu:

```
990 T HERE OK
803 T 2 OK
9AO T ALLOT OK
732 T ! OK
5B5 ;S OK.
```

Damit sieht also die Colondefinition des Wortes , (Komma) folgendermaßen aus:

```
: , HERE 2 ALLOT ! ;
```

Ab Adresse 05B5H steht die Routine SEMIS (;S), die beim Abschluß der Colondefinition durch ein ; (Semikolon) aufgerufen wird. Das mit der Adresse 09C9H beginnende Wort – (Minus) zeigt mit seiner Linkadresse auf das vorangehende Wort C, . Im Kodefeld befindet sich die Adresse 09CFH, die auf den sich direkt anschließenden Maschinenkode zeigt und aus dem sich das in Tafel 2 gezeigte Maschinenprogramm ergibt. Der Eintritt in die Routine NEXT befindet sich an der Adresse 02C0H, weshalb die letzte Anweisung im Maschinenprogramm stets ein Sprung an diese Stelle sein muß. Die häufig benötigten PUSH-Befehle (PUSH HL, PUSH DE) stehen davor. Das eigentliche Forth-Dictionary beginnt mit dem Wort LIT, an der Stelle 02CBH.

Tafel 2: Maschinenkode des Wortes – (Minus)

D1	POP DE
E1	POP HL
A7	AND A
ED 52	SBC HL DE
C3 BF 02	JMP 02BF
E5	PUSH HL

Tafel 3: Forth-Worte zur Verarbeitung doppelt genauer Zahlen

```
SCR # 100
 0 ( ====== DOUBLE NUMBERS ===1== )
 1
 2 : D– DMINUS D+ ;
 3 : 2DROP DROP DROP ;
 4 : 2SWAP ROT >R ROT R> ;
 5 : PICK 2 * SP@ + @ ;
 6 : 20VER 4 PICK 4 PICK ;
 7 : 2ROT PAD 2! 2SWAP PAD 2a 2SWAP
 8 ;
 9 : D0= OR 0= ;
10 : D= D– D0= ;
11 : D< ROT 2DUP =
12    IF 2DROP U<
13    ELSE ROT ROT D– 0 <SWAP DROP
14    ENDIF ;
15 --
```

```
SCR # 101
 0 ( ====== DOUBLE NUMBERS ===2== )
 1
 2 : DMIN  20VER  20VER D<
 3    IF              2DROP
 4    ELSE  2SWAP 2DROP ENDIF ;
 5 : DMAX 20VER  20VER D<
 6    IF      2SWAP 2DROP
 7    ELSE           2DROP ENDIF ;
 8 : DU< ROT SWAP 2DUP U<
 9    IF 2DROP 2DROP
10    ELSE = IF U< ELSE 2DROP 0
11    ENDIF ENDIF ;
12 : 2VARIABLE  <BUILDS , , DOES> ;
13 : 2CONSTANT <BUILDS , , DOES>
14    2@ ;
15
```

Tafel 4: Funktion und Stackbewegung der Worte nach Tafel 3

```
D-      d1 d2 -- d3
  d3 ist das Ergebnis der Subtraktion d1 - d2
2DROP   32b --
  32b wird vom Stack entfernt
2SWAP   32b1 32b2--32b2 32 b1
  Die zwei obersten 32 bit Einträge auf dem Stack
  werden miteinander vertauscht.
PICK    n1--n2
  Kopiert den n-ten Parameter auf den Stack und
bringt ihn zum TOS.
20VER   32b1 32b2--32b1 32b2 32b1
  Der zweite 32 bit Eintrag auf dem Stack wird ko-
  piert und an die Spitze des Stacks gebracht.
2ROT    32b1 32b2 32b3--32b2 32b3 32b1
  Die drei obersten 32 bit Einträge werden rotiert,
  der drittoberste Eintrag gelangt an die Spitze.
D0=     d--flag
  Flag ist wahr, wenn d = 0 ist.
D=      d1 d2--flag
  Flag ist wahr, wenn d1 = d2 ist.
D<      d1 d2--flag
  Flag ist wahr, wenn d2 < d2 ist.
DMIN    d1 d2--d3
  d3 ist die kleinere der beiden Zahlen d1 bzw. d2.
DMAX    d1 d2--d3
  d3 ist die größere der beiden Zahlen d1 bzw. d2.
DU      ud1 ud2--flag
  Flag ist wahr, wenn ud1 < ud2 ist. Beide Zahlen
  sind vorzeichenlos.
```

Tafel 5: Definition des Wortes %D0= mit Hilfe des ComForth-Assemblers

```
SCR # 107
 0 ( == DOUBLE NUMBER PRIM ==ASS== )
 1 ASSEMBLER
 2 CODE %D0= DE POP, ( POP DE )
 3              HL POP, ( POP HL )
 4              H A LD, ( LD A,H )
 5                L OR, ( OR L )
 6                D OR, ( OR D )
 7                E OR, ( OR E )
 8           0 HL LDWI, ( LD HL,0 )
 9           1 NZ  JRC, ( JRNZ 1 )
10             HL  DIC, ( INC HL )
11           HPUSH JP, ( JMP NEXT-1 )
12                     ( = PUSH HL )
13 END-CODE
14
15 ;S
```

Tafel 6: Assemblerprogramm und Befehlskodierung des Wortes %D0=

```
              POP DE          D1
              POP HL          E1
              LD A,H          7C
              OR L            B5
              OR D            B2
              OR E            B3
              LD HL,0         21 00 00
              JRNZ 1          20 01
              INC HL          23
              JMP NEXT-1      C3 BF 02
NEXT-1 :      PUSH HL
NEXT :        ...
```

Da es keinen Vorläufer gibt, ist die betreffende Linkadresse 0.

Erzeugen eines Secondary-Wortes

Wie deutlich wurde, stellt Forth zur Schaffung eines neuen Wortes vom Typ Secondary die Colondefinition zur Verfügung. In Tafel 3 sind zwei Screens gezeigt, die das zur Verfügung stehende Forth mit Worten zur Verarbeitung doppelt genauer Zahlen erweitern [2].

Tafel 7: Worte vom Typ Primitive Word zur Verarbeitung doppelt genauer Zahlen

```
SCR # 102
 0 ( ==== DOUBLE NUMBER PRIM ==1=)
 1  HEX
 2 CREATE %D-
 3 21 C, 06 C, 00C, 39 C, 5E C,
 4 71 C, 23 C, 56 C, 70 C, C1 C,
 5 EB C, D1 C, A7 C, ED C, 52 C,
 6 EB C, E1 C, ED C, 42 C, C1 C,
 7 C3 C, BE C, 02 C, SMUDGE
 8 CREATE %2DROP
 9 E1 C, E1 C, C3 C, C0 C, 02 C,
10 SMUDGE
11 CREATE %2SWAP
12 21 C, 06 C, 00 C, 39 C, 5E C,
13 23 C, 56 C, 23 C, 6E C, 23 C,
14 66 C, C3 C, BE C, 02 C,
15 SMUDGE                    →
```

```
SCR # 103
 0 ( ==== DOUBLE NUMBER PRIM ==2=)
 1
 2 CREATE %D0=
 3 D1C, E1 C, 7C C, B5 C, B2C,
 4 B3 C, 21 C, 00 C, 00 C, 20 C,
 5 01 C, 23 C, C3 C, BF C, 02 C,
 6 SMUDGE
 7 CREATE %D =
 8 D1 C, 21 C, 02 C, 00 C, 39 C,
 9 7E C, 71 C, AB C, 4F C, 23 C,
10 7E C, 70 C, AA C, 47
11 C, 02 C, 00 C, 39 C, 7E C,
12  AB C, 5F C, 23 C, 7E C, AAC,
13 B3 C, B0 C, B1 C, 21 C, 00 C,
14 0 C, 20 C, 01 C, 23 C, C1 C,
15 D1 C, C3 C, BF C, 02 C, SMUDGE
```

Die Funktion der definierten Worte entspricht mit Ausnahme des Zahlenformates der Funktion der Worte für einfach genaue Zahlen. In der Bezeichnung ist leider keine Einheitlichkeit zu finden. So werden selbst im Forth-83-Standard die Worte zur Verarbeitung doppelt genauer Zahlen durch das Voranstellen von 2 bzw. D nicht einheitlich gekennzeichnet. Die Namen der in Tafel 3 angegebenen Worte entsprechen der im Forth-83-Standard verwendeten Namensgebung. In Tafel 4 sind die Funktion und die Bedingungen auf dem Stack für die genannten Worte zusammengestellt. Die beiden letzten Worte sind Definitionsworte für doppelt genaue Konstanten bzw. Variablen. An dieser Stelle soll jedoch nicht weiter auf solche Worte eingegangen werden.

Für viele Anwendungen lassen sich in der gezeigten Weise Dictionary-Erweiterungen vornehmen. Falls jedoch auf Laufzeiteigenschaften zu achten ist, weil in einer zeitkritischen Applikation das betreffende Wort häufig vorkommt, so ist eine Definition als ein Wort vom Typ Primitive Word vorzuziehen. Bei Erweiterungen des Sprachkernes sollte hier stets sorgfältig entschieden werden [3].

Erzeugen eines Wortes vom Typ Primitive Word

Für die Schaffung eines Wortes vom Typ Primitive Word gibt es zwei unterschiedliche Varianten. Ist im Forth-System ein Assembler installiert, können nach dessen Aktivierung durch das Wort Assembler die gewünschten Mnemonics aufgerufen werden, die dann gemeinsam mit entsprechenden Argumenten in das Parameterfeld des neuen Wortes compiliert werden. In den üblichen Forth-Assemblern stehen die Strukturelemente IF...ELSE...ENDIF, DO...LOOP, BEGIN...WHILE...REPEAT u. a. in fast der gleichen Weise wie in der Hochsprache zur Verfügung. Von der Forth-typischen Umgekehrt Polnischen Notation wird üblicherweise Gebrauch gemacht. In der Tafel 5 ist die Verfahrensweise anhand des ComForth-Assemblers für das später noch zu betrachtende Beispiel des Wortes %D0=, des Primitive-Word-Äquivalentes zu D0=, gezeigt.

Mit CODE %D0= wird die Definition eines Wortes vom Typ Primitive Word mit dem Namen %D0= eingeleitet. Dem Namen folgen die in Umgekehrt Polnischer Notation angeführten Mnemonics der Assemblersprache, hier des ComForth-Assemblers. Zur besseren Orientierung ist die übliche Schreibweise an der rechten Seite als Kommentar angegeben. Den Abschluß einer Definition eines Wortes vom Typ Primitive Word bildet das Wort END-CODE, das das so definierte Wort gültig macht.

Ist in einem kleinen System kein Assembler vorhanden, kann man sich bei nicht zu großen Kodelängen damit helfen, daß Operationskodes und Operanden mit den Worten , (Komma) und C, direkt in das Parameterfeld des zu definierenden Wortes compiliert werden. Die Voraussetzung der nicht zu langen Kodelängen dürfte im Falle eines zu definierenden Wortes vom Typ Primitive Word im allgemeinen erfüllt sein. Das Problem besteht dann in der noch vorzunehmenden Kodierung der Assembleranweisungen. Anhand des bereits betrachteten Beispieles des Wortes %D0= soll die Vorgehensweise verdeutlicht werden (Tafel 6). Der mnemonischen Befehlsdarstellung ist die jeweils zutreffende Befehlskodierung gegenübergestellt.

In den in Tafel 7 gezeigten Screens sind einige Worte zur Arbeit mit doppelt genauen Zahlen als Primitive Word definiert. Das Prozentzeichen wurde zur Unterscheidung der ansonsten funktionsgleichen Worte verwendet. Mit dem Wort CREATE wird der Header des betreffenden Wortes aufgebaut und als noch unfertig (Smudge-Bit = 1) markiert. Es folgt anschließend die byteweise Eingabe der Operationskodes und Operanden durch das Wort C, . Abgeschlossen wird jedes Primitive Word durch einen Sprung nach NEXT bzw. an die vorgelagerten PUSH-Befehle (hier NEXT-1). Die Gültigkeit des nun fertig definierten Wortes wird durch SMUDGE (Smudge-Bit = 0) bewirkt. Auf die vorgestellte Weise lassen sich alle vorher in Forth definierten Worte (Secondaries) durch ein entsprechendes Primitive Word ersetzen. In

Tafel 8: Test des Laufzeitverhaltens

```
SCR # 104
 0 (====== LAUFZEITTEST ====== )
 1 DECIMAL 20000 CONSTANT Z
 2 : T0 Z 0 DO 1 DROP LOOP ;
 3 : T1 Z 0 DO LOOP ;
 4 : T2 Z 0 DO 1.  2DROP LOOP ;
 5 : T3 Z 0 DO 1. %2DROP LOOP ;
 6 : T4 Z 0 DO
 7       2. 1. D- 2DROP LOOP ;
 8 : T5 Z 0 DO
 9 :                2.1.%D-%2DROP LOOP ;
10 : T6 Z 0 DO 1.D0= DROP LOOP ;
11 : T7 Z 0 DO 1.%D0= DROP LOOP ;
12 : T8 0 DO 1.1.D= DROP LOOP ;
13 : T9 Z 0 DO 1.%D= DROP LOOP ;
14 ;
```

Tafel 9: Laufzeiten der einzelnen Tests

Test	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
tins	6	4	11	7	22	13	13	9	27	13

Tafel 7 ist das auszugsweise für unterschiedlich komplexe Worte vorgenommen worden. Auf Grund des geringeren Komforts, der zur Definition eines Wortes vom Typ Primitive Word genutzt werden kann, muß sich die Frage nach dem Nutzen, der an dieser Stelle bei der Laufzeit zu erwarten ist, ergeben. Zur Abschätzung des Laufzeitverhaltens wurden einige orientierende Tests zusammengestellt.

Test des Laufzeitverhaltens

In der Tafel 8 sind die zum Test des Laufzeitverhaltens herangezogenen Tests angegeben. Um ein hinreichend problemloses Stoppen mit der Armbanduhr zu ermöglichen, wurden Schleifen mit einer entsprechend hohen Durchlaufzahl verwendet. Tafel 9 gibt die Laufzeiten der einzelnen Tests bei 20 000 Schleifendurchläufen wieder.
Die Laufzeitunterschiede zwischen den vergleichbaren Tests T2 und T3, T4 und T5, T6 und T7 sowie T8 und T9 sind deutlich. Da in der Schleife weitere Worte enthalten sind, die den Test erst möglich machen, werden die Unterschiede noch größer. Das Auszählen der erforderlichen Taktperioden für das Wort %D= ergab beispielsweise 174 Taktperioden, wodurch bei einer Taktfrequenz von 2,5 MHz mit einer Laufzeit von ungefähr 100 μs gerechnet werden muß.

Zusammenfassung

Die Definition von Anwenderworten läßt sich sowohl in den Worttypen Secondary als auch Primitive Word vornehmen. Das Primitive Word hat dabei naturgemäß die besseren Laufzeiteigenschaften, weshalb bei Erweiterungen des Sprachkerns sorgfältig abgewogen werden sollte, welchem Worttyp der Vorzug gegeben wird. Zeitkritische Applikationen erfordern häufig Worte vom Typ Primitive Word.
Zur Definition eines Wortes vom Typ Secondary wird die Colondefinition herangezogen, während für die Definition eines Primitive Word ein im Forth-System installierter Assembler oder das direkte Compilieren von Operationskode und Operanden Verwendung findet. Anhand einiger Worte zur Verarbeitung doppelt genauer Zahlen werden beide Wege beschrieben und das unterschiedliche Laufzeitverhalten verdeutlicht.

Literatur

[1] Domschke, W.; Katzmann, K.: Der Modul M 026 Forth für die Kleincomputer KC 85/2 und KC 85/3. Mikroprozessortechnik, Berlin 1 (1987) 8, S. 244–246
[2] Briner, R. G.: Double Number Words in Forth. Elektroniker, Aaarau 22 (1983) 18, S. 49–55
[3] Zech, R.: Die Programmiersprache Forth. München: Franzis Verlag 1985