Einführung in Forth-83

Teil 1

Dr. Hartmut Pfüller (Leiter), Dr. Wolfgang Drewelow, Dr. Bernhard Lampe, Ralf Neuthe, Egmont Woitzel
Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik

Gliederung

1. Einführung und Kernwortschatz

Die dialogorientierte Programmiersprache Forth wurde in den sechziger Jahren von Charles H. Moore in den USA entwickelt und ab 1971 zunächst für die Echtzeitsteuerung von Radioteleskopen eingesetzt. Entwicklungsziele waren maximale Handlichkeit der Sprache zwecks hoher Produktivität des Programmierers und größte Einfachheit des Übersetzerkonzepts. Herausgekommen ist eine organische und kompakte Einheit von Sprache und Übersetzer. Forth ist erweiterbar, das heißt, der Quelltext kann für sich selbst die eigenen Ausdrucksmittel ändern und neue Ausdrucksmittel höheren Niveaus erzeugen. Damit ist problemnahe Programmierung für nahezu jede Anwendung in einheitlicher Softwareumgebung und *in einem Zuge* möglich, also ohne Mehrpaßübersetzung. In Richtung niederen Niveaus sind im Quelltext die Maschinenbefehle des Wirtsprozessors verwendbar. Das ist nützlich für direkte Gerätesteuerungen und Zugriffe auf Treiber.

Forthsysteme sind je nach Ausstattung etwa drei KByte bis über 16 KByte groß und für praktisch alle Rechner verfügbar. Forth wurde inzwischen in mehreren Etappen standardisiert; die vorliegende Beschreibung folgt dem Standard Forth-83. Eventuelle Unklarheiten beim Nachvollziehen der Beispiele sollten Veranlassung sein, die Verträglichkeit des benutzten Systems mit diesem Standard zu überprüfen.

1.1. Die Softwarekonzeption von Forth

1.1.1. Die Architektur von Forth

Das Forthsystem ist ein Rechenprogramm, das gleichzeitig als Betriebssystem, als Compiler und als Kommandoprozessor (*Textinterpreter*) arbeitet. Bild 1.1 soll das an einem Schichtenmodell anschaulich machen. Die niedrigste Schicht enthält alle Programmmoduln, die aus Maschinencode gebildet sind. Dieses Prinzip, die Gegebenheiten einer konkreten Hardware mittels eines zugeordneten Pakets von Maschinenprogrammen *abzufangen*, ist mit dem BIOS-Teil des Betriebssystems CP/M vergleichbar. Auch die nächsthöheren Schichten sind in Moduln gegliedert, allerdings bestehen diese nicht aus Maschinencode, sondern aus sogenanntem Forthcode. Das sind Listen von Adressen, wobei jede Adresse als Pointer zu einem bestimmten Modul aus Maschinencode oder aus Forthcode verweist. Die Betriebssystemschicht und die darüberliegenden sind dadurch maschinenunabhängig, also portabel. Alle Schichten können jederzeit auch im Dialog erweitert werden.

1.1.2. Das Wortkonzept

Das gesamte Forthsystem besteht aus einer größeren Anzahl (je nach Systemgröße z. B. etwa 70 ... 300) von relativ kleinen Moduln (teils aus Maschinencode, teils aus Forthcode). Alle diese Grundfunktionen sind im Hauptspeicher lexikonförmig geordnet aufbewahrt. Wegen der tatsächlichen Ähnlichkeit mit einem Lexikon heißt dieser Programmteil *Wörterbuch*, und jeder der einzelnen Moduln wird als *Wort* bezeichnet. Die Worte im Wörterbuch dienen als Kommunikationsmittel zwischen Mensch und Rechner und müssen demzufolge von beiden *verstanden* werden. Zu diesem Zweck ist jeder Eintrag zweigeteilt und so aufgebaut, daß der Name des Wortes für den Menschen als Text lesbar und der zugehörige Codeteil für den Rechner ausführbar ist. Vier Beispiele für Namen sind:

; */MOD 2! VOCABULARY

Die Namensbildung ist sehr freizügig; Sonderzeichen sind an beliebiger Stelle erlaubt oder können auch einzeln als Name gelten.

1.1.3. Das Stapelkonzept

Außer Namen können im Eingabetext von Tastatur oder Massenspeicher natürlich auch Zahlen erscheinen. Diese Zahlen werden eine nach der anderen in das *interne Format* konvertiert, so zum *Parameterstapel* (Parameterstack) geschafft und dort abgelegt. Zur internen Wertdarstellung dienen durchgängig 16-Bit-Dualzahlen; für negative Zahlen wird die Zweierkomplementdarstellung verwendet. Auf dem Parameterstapel bleiben die Werte dann so lange liegen, bis sie von irgendeinem Wort wieder abgeholt (*verbraucht*) werden. Damit ist gleichzeitig angedeutet, wie Forthworte mit lokalen Eingangsparametern versorgt werden: Die Parameter müssen irgendwann vor Aktivierung des Wortes auf dem Parameterstapel hinterlegt worden sein. Falls mit mehreren Werten hantiert wird, gilt das LIFO-Prinzip (englisch: last in – first out). Entsprechend wird mit lokalen Ausgangsparametern verfahren: Falls ein Wort lokale Ergebniswerte liefert, bleiben diese nach Ausführung des Wortes auf dem Stapel liegen und sind so zum Beispiel wieder als Eingangsparameter für nachfolgende Worte verfügbar. Da auf diese Weise bei Aufrufen die aus anderen Programmiersprachen bekannten Listen von aktuellen und formalen Parametern entfallen, spricht man bei Forth auch von *impliziter Parameterübergabe*. Für Programmdokumentationen oder Quelltextkommentare kann es erforderlich sein, genauere Angaben über Anzahl und Art der lokalen Ein- und Ausgangsparameter zu machen. Dafür hat sich in Forth eine bestimmte Art der Kommentierung eingebürgert: In runden Kommentarklammern wird notiert, wie der Stapel vor und nach Ausführung des kommentierten Wortes belegt ist. Die Notation

(n1 n2 n3 ===> n4 n5)

bedeutet in diesem Sinne, daß das kommentierte Forthwort drei Eingangsparameter vom Stapel holt – mit n3 an der Stapelspitze – und zwei neue Werte – mit n5 an der Stapelspitze – als Ausgangsparameter hinterläßt. Der Pfeil symbolisiert die Ausführung des Wortes.

1.1.4. Das Erweiterungskonzept

Die Vorgehensweise beim Programmieren in Forth soll an einem Steuerprogramm für einen hypothetischen x-y-Schreiber erläutert werden. Dieser Plotter sei so einfach, daß seine Hardware nur drei Funktionen kennt, und zwar

a) *Stift anheben*
b) *Stift absenken*
c) *geradenwegs die Absolutposition (x, y) anfahren.*

Wie diese Funktionen von der Rechnerperipherie aus zu aktivieren sind, soll bekannt sein. Programmieren in Forth heißt nun, das Forthsystem zu erweitern, indem der Programmierer zu den im Wörterbuch schon existierenden Worten neue Worte *hinzudefiniert*, nämlich solche, die Schritt für Schritt die Programmieraufgabe lösen. Dabei können alle schon im Wörterbuch existierenden Worte ausgenutzt werden. Zum Definieren neuer Worte gibt es (natürlich *auch* im Wörterbuch!) *Definitionsworte*. Ein neues Wort für die Maschinencodeschicht wird zum Beispiel durch Voranstellen des Definitionswortes **CODE** vor einen selbstzuwählenden Namen erzeugt. Mit dem Wort **END-CODE** wird ein solches Maschinenprogramm dann wieder beendet. Der Forthprogrammierer geht nun üblicherweise so vor, daß er entsprechend den Gerätedaten das Assemblerprogramm konzipiert und dann eintippt:

CODE HEBEN . . . . . END-CODE
CODE SENKEN . . . . . END-CODE

Da die konkreten Assemblerbefehle hier nicht im Mittelpunkt stehen, wurden sie nur

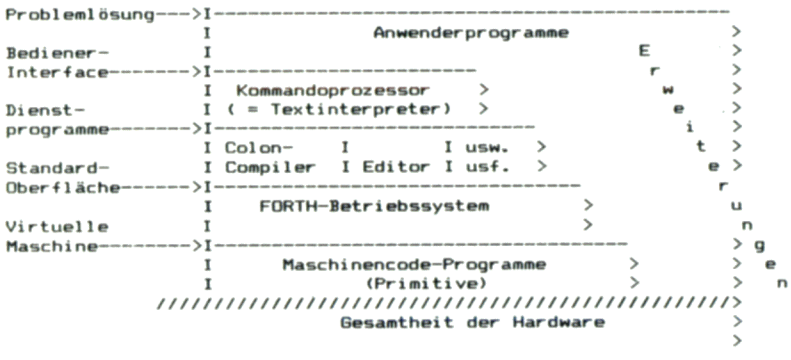

Bild 1.1 Schichtenmodell eines Forthsystems

Kommando	Stapel			
(Anfang)	a	b		
OVER	a	b	a	
OVER	a	b	a	b
*	a	b	a*b	
ROT	b	a*b	a	
ROT	a*b	a	b	
+	a*b	a+b		
/	a*b/(a+b)			

Bild 1.5 Stapelbelegung während der Formelberechnung

```
2233  -77  444  (cr) ok      (wird auf Stapel gelegt)
 .  (cr) 444 ok               (letzter Wert entnommen)
 .  .  (cr) -77 2233 ok       (nächste Werte entnommen)
```

Bild 1.2 Ausgabe von Zahlen mittels Punktbefehl

```
1  65535  -2  (cr) ok
U.  U.  U.  (cr) 65534 65535 1 ok
```

Bild 1.3 Vorzeichenlose Ausgabe von Zahlen

```
0  1  D.  (cr) 65536 ok
1  0  D.  (cr) 1 ok
```

Bild 1.4 Ausgabe von doppeltgenauen Zahlen

durch Punkte angedeutet. Nachdem nun die neuen Worte **HEBEN** und **SENKEN** definiert sind, kann der Programmierer durch deren Aufruf bei angeschlossenem Plotter kontrollieren, inwieweit die von ihm beabsichtigte Funktion korrekt ausgeführt wird. Die *Ausführung* von Worten erreicht man einfach durch Eintippen ihres Namens und Bestätigung der Eingabe mit der <Enter>-Taste. Diese *normale* Betriebsart von Forth heißt deshalb auch *Ausführungsmodus*. Wenn die neuen Worte augenscheinlich nicht korrekt arbeiten, müssen sie berichtigt neu eingegeben werden, bis der gewünschte Erfolg erreicht ist. Danach werde in ähnlicher Weise die dritte Funktion als Maschinenprogramm codiert:

```
CODE ANFAHREN  . . . . . END-CODE
```

Der Programmierer hat **ANFAHREN** als sinnfälligen Namen für diese Funktion gewählt. Hier ist nun zu beachten, daß dieses Wort die Parameter x und y für die Zielkoordinaten benötigt. In Quellprogrammen gehört es zu gutem Forthstil, mit dem neudefinierten Namen entsprechende Hinweise als Kommentar in runden Klammern zu notieren.

```
CODE ANFAHREN    ( x y ===>)
    . . . . . END-CODE
```

Durch die Ausführung des Wortes **ANFAHREN** werden also zwei Werte vom Stapel entfernt (verbraucht). Auch die Funktion dieses Wortes kann nun im Dialog getestet werden, z. B. durch Eintippen von:

```
50  70  ANFAHREN
```

Wenn das Maschinenprogramm **ANFAHREN** korrekt ist, muß damit die Position (x = 50, y = 70) angefahren werden. Natürlich können Worte auch im Verbund arbeiten, zum Beispiel so:

```
HEBEN  0  0  ANFAHREN
SENKEN  100  50  ANFAHREN  HEBEN
```

Nach Eingabe dieser Zeilen muß der Plotter eine Strecke vom Punkt (x = 0, y = 0) zum Punkt (x = 100, y = 50) zeichnen. Neben dem Definitionswort **CODE** für das Eintragen von Maschinencodeworten ins Wörterbuch gibt es weitere Definitionsworte. Das vielleicht wichtigste von ihnen ist der *Doppelpunkt*. Er dient zur Erweiterung der höheren Schichten um Moduln in Forthcode. Für den Plotter kann zum Beispiel ein Wort definiert werden, dessen Funktion es ist, zwei Punkte durch eine Linie zu verbinden:

```
: VERBINDEN  ( x2 y2 x1 y1 ===>)
   ANFAHREN SENKEN ANFAHREN HEBEN ;
```

Hier werden die Worte **ANFAHREN**, **SENKEN** und **HEBEN** nach Eingabe *nicht* ausgeführt, sondern als Programm in Forthcode unter dem neuen *Stichwort* **VERBINDEN** ins Wörterbuch kompiliert. Das wird dadurch erreicht, daß der Doppelpunkt generell nach seiner Aktivierung die Betriebsart vom Ausführungsmodus in den sogenannten *Kompilationsmodus* umschaltet. Das abschließende Semikolon (*auch* ein Forthwort!) schaltet dann wieder vom Kompilationsmodus in den Ausführungsmodus zurück. Die Gesamtkonstruktion einschließlich Doppelpunkt und Semikolon wird Doppelpunktdefinition genannt. Nach der Definition kann natürlich auch das Wort **VERBINDEN** im Dialog getestet werden, zum Beispiel so:

```
80  100  0  20  VERBINDEN
```

Die Ausführung des Wortes **VERBINDEN** bedeutet nun, daß genau diejenigen Befehle ausgeführt werden, die in der Doppelpunktdefinition notiert sind, also: **ANFAHREN SENKEN ANFAHREN HEBEN**. Man mache sich klar, daß die Richtung vom Punkt (x = 0, y = 20) zum Punkt (x = 80, y = 100) führt: Die Koordinaten des Punktes (x = 0, y = 20) liegen obenauf und werden deshalb als erste vorgefunden und angefahren. In ähnlicher Weise können im Dialogbetrieb nach und nach weitere Worte definiert, schrittweise zu komplexeren Funktionen zusammengefaßt und getestet werden. Zum Beispiel könnte man die Worte definieren, die lediglich die Parameter von markanten Punkten des Achsenkreuzes liefern:

```
: NULLPUNKT  ( ===> x0 y0)  0  0 ;
: XMAX  ( ===> xmax y0)  100  0 ;
: YMAX  ( ===> x0 ymax)  0  100 ;
```

Das erlaubt dann sinnfällige Wortsequenzen wie

```
. . . NULLPUNKT ANFAHREN . . .
```

Weiter könnte damit nun beispielsweise ein Wort zum Zeichnen der positiven Achsen definiert werden:

```
: ACHSEN  ( ===>)
  NULLPUNKT  YMAX  VERBINDEN
  XMAX  NULLPUNKT  VERBINDEN ;
```

Wenn Programme ausprobiert werden sollen, ohne daß das anzusprechende Gerät verfügbar ist, kann man sich dadurch helfen, daß die Grundfunktionen durch Doppelpunktworte definiert werden. Dabei muß das Stapelverhalten korrekt nachgebildet werden. Anstatt die eigentliche Funktion auszuführen, können zum Beispiel Zeichenkettentexte auf dem Bildschirm angezeigt werden:

```
: HEBEN    ." heben " ;
: SENKEN   ." senken " ;
: ANFAHREN  ( x y ===>)
   . .     ." anfahren " ;
```

Nach dem Eintippen dieser Worte könnten die obigen Beispiele zur Plottersteuerung nachvollzogen werden.

1.2. Das Bedienerinterface

1.2.1. Integerzahlen

Der Textinterpreter von Forth versucht für jeden (in Leerzeichen eingegrenzten) Eintrag zunächst, die Zeichenfolge als Name eines Forthwortes im Wörterbuch aufzufinden. Falls das mißlingt, wird als nächstes geprüft, ob die eingebene Zeichenfolge als Zahl verstanden werden kann, das heißt, ob sie ausschließlich aus gültigen Ziffernzeichen (allenfalls mit führendem Minuszeichen) besteht. Wenn diese Prüfung erfolgreich ist, wird die Ziffernzeichenfolge in die rechnerinterne Zahlenwertdarstellung umgewandelt und so auf dem Parameterstapel abgelegt.

1.2.2. Worte

Jedes eingegebene Wort, ob es nun eine (ein- oder mehrstellige) Zahl oder ein im Wörterbuch enthaltenes Forthwort repräsentiert, muß vom nachfolgenden Wort im Programmtext durch mindestens ein Leerzeichen getrennt sein. Nachfolgend wird die Benutzung von Zahlen und Forthworten am Beispiel einiger Ausgabebefehle demonstriert.

Wortname: .

Stapeleffekt: (n ===>)

Funktion: Das ASCII-Zeichen *Punkt* ist in Forth der einfache Ausgabebefehl; er holt (d. h. entfernt) den obersten 16-Bit-Wert vom

16-Bit-Wert vom Parameterstapel, faßt ihn als vorzeichenbehaftete Zahl in Zweierkomplementdarstellung (−32768 ... 32767) auf, wandelt ihn in die externe Ziffernzeichendarstellung und sendet diese ASCII-Zeichenkette als externe Darstellung der Dualzahl zum Ausgabegerät (Bild 1.2).
Zur Unterscheidung von den Bedienereingaben ist die Rechnerreaktion in diesen Dialogmitschnitten jeweils durch Unterstreichung gekennzeichnet. Das zusätzlich eingefügte (cr) soll bedeuten, daß vor dieser Stelle die Entertaste betätigt wurde.
Wortname: **U.**
Stapeleffekt: (u ===>)
Funktion: Ausgabebefehl wie Punkt; faßt den Wert als vorzeichenlose Dualzahl (0 ... 65536) auf (Bild 1.3).
Wortname: **D.**
Stapeleffekt: (n n ===>) oder identisch: (d ===>)
Funktion: Ausgabebefehl; holt die oberen beiden (16-Bit-)Werte vom Stapel und faßt dieselben gemeinsam als 32-Bit-Ganzzahl in vorzeichenbehafteter Zweierkomplementdarstellung (−2147483648 bis 2147483647) auf (Bild 1.4).
Eine Kurzbeschreibung aller aufgeführten Befehle ist in Tafel 1.2 zu finden. Da diese in vielen Fällen ausreichend ist, wird es im weiteren Verlauf seltener erforderlich sein, Worte so ausführlich wie eben vorgeführt zu beschreiben. In allen drei obigen Beschreibungen wurde absichtlich dazugesagt, wie der Ausgabebefehl den Stapelwert *auffaßt*. Diese *Auffassungsfrage* gilt nicht nur für Ausgabebefehle, sondern generell: Auf dem Stapel werden physisch nur 16-Bit-Werte verwaltet. Der Programmierer allein entscheidet bei neuen Definitionen durch die von ihm organisierte Weiterverwendung darüber, ob sein Forthwort einen Wert als vorzeichenbehaftet, als ASCII-Zeichen mit nur sieben gültigen Bits oder mit weiteren Parametern gemeinsam als mehrfachgenauen Wert benutzt oder auf völlig andere Weise. Bei der Verwendung von bereits existierenden Worten muß er sich natürlich darüber informieren, welche Bedeutung diese jeweils den Parametern beilegen.

1.3. Der Parameterstapel

Da die Werte auf dem Parameterstapel für sehr verschiedene Anwendungsfälle als Eingangsparameter verwendet werden können, ist es manchmal zweckmäßig, vorhandene Werte zu kopieren oder umzusortieren. Dafür sind die in den nachfolgenden Unterabschnitten aufgeführten Worte nützlich.

1.3.1. Vervielfältigung

Die sechs Worte
DUP OVER PICK ?DUP 2DUP 2OVER
stellen alle in irgendeiner Weise Kopien von bestimmten Werten irgendwo auf dem Stapel her, die dann als Ergebnisparameter auf das obere Stapelende (englisch: Top of stack, Abk.: TOS) abgelegt werden. Die Stapeldiagramme und Kurzbeschreibungen sind in Tafel 1.2 enthalten. Mit **PICK** kann ein beliebiger Wert zum TOS kopiert werden, dessen Position man allerdings explizit angeben muß. **?DUP** dupliziert den Wert im TOS genau dann, wenn er nicht gleich Null ist. Das ist besonders in Verbindung mit Entscheidungsstrukturen nützlich.

1.3.2. Ordnungsbefehle

Jedes der fünf Worte
ROT SWAP ROLL 2ROT 2SWAP
leistet in irgendeiner Weise eine Umsortierung von Werten auf dem Parameterstapel; der Füllstand des Stapels wird dabei nicht geändert.

1.3.3. Entfernen von Werten

Die Worte **DROP** und **2DROP** entfernen vom TOS einen 16-Bit-Wert bzw. einen 32-Bit-Wert. Das wird zum Beispiel dann benötigt, wenn erst nach Entscheidungsfolgen klar wird, welcher von mehreren Parametern weiter benötigt wird und welcher nicht. Überflüssige Werte können dann durch Umsortierung nach oben gebracht und mit diesen Befehlen *vernichtet* werden.

1.3.4. Stapeltiefe

Das Wort **DEPTH** (deutsch: Tiefe) liefert als Ergebnisparameter eine Zahl, die angibt, mit wieviel 16-Bit-Werten der Parameterstapel gefüllt ist.

1.4. Arithmetische Operationen

1.4.1. Umgekehrte polnische Notation

Im Plotterbeispiel wurde eine Befehlsfolge **NULLPUNKT ANFAHREN** benutzt. Dabei folgt auf den (zusammengesetzten) *Parameter* **NULLPUNKT** die *Operation* **ANFAHREN**. Nach diesem Prinzip kann auch mit Zahlen operiert werden. Zum Beispiel gibt es in Forth eine Operation für das Addieren; das entsprechende Forthwort hat den Namen + erhalten. Angenommen, auf dem Stapel liege ein Wert **5**, dann führt die Befehlsfolge **3** + dazu, daß die **5** um den Wert **3** erhöht, also zu einer **8** wird. Wesentlich an dieser Betrachtung ist, daß in Forth generell die Operation *hinter* dem (oder den) Operanden steht (sogenannte Postfixnotation; andere übliche Bezeichnungen dafür sind UPN = umgekehrte polnische Notation oder englisch: RPN = reverse Polish Notation). In Verbindung mit der LIFO-Verwaltung des Parameterstapels wird sich das als sehr praktisch erweisen. Trotzdem ist es etwas unüblich, denn die in der Schule gelehrte Notation setzt den Operator *zwischen* die Operanden (sogenannte Infixnotation). Es läßt sich zeigen, daß daneben auch noch eine Präfixnotation möglich ist (Operator *vor* den Operanden; z. B. von der Programmiersprache LISP bevorzugt) und daß alle diese Notationen ineinander überführbar sind. Ungewohnt ist die Postfixnotation am Anfang vielleicht am ehesten bei größeren Formeln.

1.4.2. Die Grundrechenarten

Tafel 1.2 zeigt die Funktion der sechs Worte
+ − * / D+ D−
Die ersten vier erwarten jeweils zwei Parameter auf dem Stapel, entfernen diese und hinterlassen im TOS das Ergebnis der Operation. Der Programmierer muß wissen, daß Bereichsüberschreitungen zum Beispiel bei Addition oder Multiplikation nicht reklamiert werden. Weiter ist wichtig zu beachten, daß die Division nur den ganzzahligen Teil des Quotienten liefert; ein eventueller Rest wird ignoriert. Alternativen dazu enthält der nächste Abschnitt. Die beiden Worte **D+** und **D−** hinterlassen entsprechend die doppeltgenaue Summe bzw. Differenz von doppeltgenauen Eingangswerten. Die praktische Benutzbarkeit der Postfixnotation in Verbindung mit Kopier- und Ordnungsbefehlen soll an einem Beispiel gezeigt werden. Angenommen, auf dem Stapel liegen zwei Parameter a und b, und es soll der Ausdruck
(a * b)/(a + b) berechnet werden. Das ist mit der folgenden Sequenz möglich:
OVER OVER * ROT ROT + /
Da das nicht besonders anschaulich ist, kann es für Entwicklungszwecke zum Selbstverständnis nützlich sein, in einer Art *vertikalen Notation* die aktuelle Belegung des Stapels nach jedem Kommando als Kommentar festzuhalten (Bild 1.5).

1.4.3. Division mit Rest

Die beiden Worte
/MOD MOD
können benutzt werden, wenn die Vernachlässigung des Restes bei der ganzzahligen Division nicht zulässig ist. **/MOD** liefert dann im TOS den Quotienten und darunter den Rest. **MOD** liefert nur den Rest selbst.

1.4.4. Skalierung

Die beiden Worte
***/ */MOD**
erwarten drei Werte x y z als Parameter und berechnen den Wert von x*y/z, wobei die Zwischenergebnisse intern mit doppelter Genauigkeit geführt werden. ***/** liefert einen Ergebnisparameter. Mit diesem Operator läßt sich unsere Formel (a*b)/(a+b) aus Abschnitt 1.4.2. auch so berechnen:
OVER OVER + */
Ganz ähnlich wie der eben besprochene Befehl ***/ liefert */MOD** ebenfalls das Ergebnis im TOS, dazu aber als zweiten Wert (unter dem TOS) zusätzlich den Rest der abschließenden Division.

1.4.5. Gemischtgenaue Operationen

Gemischtgenaue Operationen sind solche, bei denen Parameter verschiedener Genauigkeit eine Rolle spielen. Zum Beispiel erwartet **UM*** zwei einfachgenaueParameter (vorzeichenlos) und hinterläßt deren doppeltgenaues Produkt (ebenfalls vorzeichenlos). **UM/MOD** erwartet im TOS einen einfachgenauen Divisor und darunter einen doppeltgenauen Dividenden. Das Ergebnis ist wieder einfachgenau; alle Parameter werden als vorzeichenlos aufgefaßt.

1.4.6. Begrenzerbefehle

Die vier Befehle
MAX MIN DMAX DMIN
erwarten je zwei Eingangsparameter und liefern als Ausgangsparameter das Extremum der beiden.

1.4.7. Vorzeichenbefehle

ABS und **DABS** bilden den Betrag von einfach- bzw. doppeltgenauen Zahlen; **NEGATE** und **DENEGATE** bilden jeweils das Zweierkomplement des einfach- bzw. doppeltgenauen Eingangsparameters. Damit wird eine Vorzeichenumkehr erreicht.

1.4.8. Maschinenahe Operationen

Die Gruppe

1+ 1– 2+ 2– 2/ D2/

faßt solche Operationen zusammen, die in Programmen erfahrungsgemäß häufig vorkommen und für die gleichzeitig besonders schnelle Realisierungen in Maschinencode möglich sind. Das ist das Erhöhen und das Vermindern um eins und um zwei sowie eine bitweise Rechtsverschiebung des Eingangsparameters (einfach- bzw. doppeltgenau), die in der internen Darstellung eine Division durch zwei realisiert.

wird fortgesetzt

Tafel 1.1 Symbole für die Bedeutung der Stapelparameter

TOS	oberster Wert auf dem Parameterstapel (englisch: Top Of Stack)
+n	Stapeleintrag im Bereich 0 < x < = 32767
16b	Stapeleintrag mit 16 gültigen Bits
32b	doppeltgenauer Stapeleintrag mit 32 gültigen Bits
8b	Stapeleintrag mit 8 gültigen Bits (b0/ bis b7)
?	Stapeleintrag mit den Werten "true" (? ≠ 0) bzw. "false" (? = 0)
addr	Stapeleintrag, der als Adresse angesehen wird
c	Stapeleintrag, der ein (ASCII-)Zeichen spezifiziert
d	doppelter Stapeleintrag im Bereich −2147483648 < x < = 2147483647
+d	positive doppelt genaue Zahl 0 < x < = 4294967295
n	Stapeleintrag im Bereich −32768 < x < = 32767
u	Stapeleintrag im Bereich 0 < x < = 65535
ud	doppelter Stapeleintrag im Bereich 0 < x < = 4294967295
w	Stapeleintrag im Bereich −32768 < x < = 65535
wb	Stapeleintrag im Bereich −128 < x < = 255
wd	Stapeleintrag im Bereich −2147483648 < x < = 4294967295

Tafel 1.2 Kurzbeschreibung der Forthworte

Name	Stapeleffekt	Beschreibung
Zahlenausgaben		
	(n = = = >)	16-Bit-Wert mit MSB als Vorzeichen ausgeben
U.	(u = = = >)	16-Bit-Wert als Positivwert ausgeben
D.	(d = = = >)	32-Bit-Wert mit MSB als Vorzeichen ausgeben
Parameterstapel; Vervielfältigungen		
DUP	(16b = = = > 16b 16b)	obersten Stapelwert identisch duplizieren
OVER	(16b0 16b1 = = = > 16b0 16b1 16b0)	zweiten Wert zum TOS kopieren
PICK	(16bj 16bi 16bh . . . 16b0 i = = = > 16bj 16bi 16bh . . . 16b0 16bi)	i-ten Wert zum TOS kopieren
?DUP	(0 = = = > 0) (16b = = = > 16b 16b)	TOS nur dann duplizieren, wenn er nicht null ist
2DUP	(32b = = = > 32b 32b)	32-Bit Wert duplizieren
2OVER	(32b0 32b1 = = = > 32b0 32b1 32b0)	zweiten doppeltgenauen Wert zum TOS kopieren
Ordnungsbefehle		
ROT	(16b0 16b1 16b2 = = = > 16b1 16b2 16b0)	dritten Wert nach oben "rollen"
SWAP	(16b0 16b1 = = = > 16b1 16b0)	obere beide Werte tauschen
ROLL	(16bj 16bi 16bh . . . 16b0 i = = = > 16bj 16bh . . . 16b0 16bi)	i-ten Wert nach oben "rollen"
2ROT	(32b0 32b1 32b2) = = = > 32b1 32b2 32b0)	dritten doppeltgenauen Wert nach oben "rollen"
2SWAP	(32b0 32b1 = = = > 32b1 32b0)	obere beide doppeltgenauen Werte tauschen
Entfernen von Werten		
DROP	(16b = = = >)	TOS entfernen
2DROP	(32b = = = >)	doppeltgenauen Wert vom TOS entfernen
Stapeltiefe		
DEPTH	(= = = > +n)	Gesamtzahl der 16-Bit-Werte auf dem Stapel
Grundrechenarten		
+	(w1 w2 = = = > w3)	liefert w3 als Summe aus w1 und w2
–	(w1 w2 = = = > w3)	liefert w3 als Diffrenz aus w1 – w2
*	(w1 w2 = = = > w3)	bildet w3 als Produkt aus w1 und w2
/	(n1 n2 = = = > n3)	bildet n3 als Quotient n1/n2
D+	(wd1 wd2 = = = > wd3)	addiert doppeltgenaue Werte wd1 und wd2 zu wd3
D–	(wd1 wd2 = = = > wd3)	subtrahiert zwei doppeltgenaue Zahlen zu wd3
Division mit Rest		
/MOD	(n1 n2 = = = > n3 n4)	bildet Quotient n4 und Rest n3 von n1/n2
MOD	(n1 n2 = = = > n3)	bildet den Rest n3 der Division n1/n2
Skalierung		
*/	(n1 n2 n3 = = = > n4)	n4 = n1*n2/n3; Zwischenergebnis n1*n2 ist doppeltgenau
*/MOD	(n1 n2 n3 = = = > n4 n5)	n5 = n1*n2/n3; n4 ist der Rest bei der Division
Gemischtgenaue Operationen		
UM*	(u1 u2 = = = > ud)	doppeltgenaues Produkt einfachgenauer Positivwerte
UM/MOD	(ud u1 = = = > u2 u3)	Quotient u3 und Rest u2 von ud/u1
Vergleichende Befehle		
MAX	(n1 n2 = = = > n3)	n3 ist die größere der beiden Zahlen n1 und n2
MIN	(n1 n2 = = = > n3)	n3 ist die kleinere der beiden Zahlen n1 und n2
DMAX	(d1 d2 = = = > d3)	d3 ist die größere der doppeltgenauen Zahlen d1 und d2
DMIN	(d1 d2 = = = > d3)	d3 ist die kleinere der doppeltgenauen Zahlen d1 und d2
Vorzeichenbefehle		
ABS	(n = = = > u)	u ist der Absolutbetrag von n
DABS	(d = = = > ud)	ud ist der Absolutbetrag von d
NEGATE	(n1 = = = > n2)	n2 ist das Zweierkomplement von n1
DNEGATE	(d1 = = = > d2)	d2 ist das Zweierkomplement von d1
Maschinennahe Operationen		
1+	(w1 = = = > w2)	w1 wird inkrementiert zu w2
1–	(w1 = = = > w2)	w1 wird dekrementiert zu w2
2+	(w1 = = = > w2)	w1 wird um 2 inkrementiert zu w2
2–	(w1 = = = > w2)	w1 wird um 2 dekrementiert zu w2
2/	(w1 = = = > w2)	w1 wird um ein Bit arithmetisch rechtsverschoben
D2/	(wd1 = = = > wd2)	analog 2/ für doppeltgenaue Zahlen

Zu diesem Kurs

Hier wird dem interessierten Leser die Möglichkeit gegeben, sich über einige der wesentlichen Eigenarten und Potenzen von Forth grob zu orientieren. Wer dadurch angeregt wird, sich ernsthafter mit diesem zukunftsträchtigen Softwarekonzept zu befassen, dem sei das sorgfältige Studium der Bücher von Brodie /1/, /2/ und Zech /3/ empfohlen. Daneben kann man sich in der Speziallliteratur über Themen informieren, die in diesem Kurs nur kurz oder gar nicht erwähnt werden, zum Beispiel: Editieren in Forth / Assemblerprogrammierung / Interrupts, Echtzeit und Multitasking / Metakompilation und Crosscompilation / Fließkommarechnung / Forthprozessor in Hardware.

Hilfreich ist sicherlich auch der Kontakt zu anderen Forthprogrammierern. Für beruflich Interessierte bietet sich hier die Kammer der Technik an. Dort arbeitet in der wissenschaftlichen Sektion Computer- und Mikroprozessortechnik des Fachverbandes Elektrotechnik ein Fachausschuß Forth. Auch außerberuflich an Forth Interessierte haben sich beim Kulturbund in Leipzig zu einer Interessengemeinschaft Forth zusammengefunden, die landesweit aktiv ist.

Literatur

/1/ Brodie, L.: Programmieren in Forth. München: Carl Hanser Verlag 1984
/2/ Brodie, L.: In Forth Denken. München: Carl Hanser Verlag 1986
/3/ Zech, R.: Forth-83. München: Franzis Verlag 1987

✉ KONTAKT ☎

WPU Rostock, Sektion Technische Elektronik, WB Automatische Steuerungen, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 452 14