Herwig Feichtinger

Pascal und Forth für den AIM-65

Seit einiger Zeit bietet Rockwell für sein kleines Entwicklungssystem AIM-65 zwei neue Programmiersprachen an, die zusätzlich zum 6502-Assembler, zu Basic und PL-65 zur Verfügung stehen: Forth und Pascal. Hier ein kurzer Bericht über deren Eigenschaften.

Wer seinen AIM-65 (oder PC-100 von Siemens) nicht über die Expansions-Steckerleiste erweitert hat, kann zunächst nur Forth nachrüsten: Die zwei Forth-ROMs passen in die beiden PROM-Sockel, die ursprünglich für den 8-KByte-Basic-Interpreter reserviert waren. Im Grundzustand des AIM ist es also nicht möglich, ohne Umstecken von ROMs wechselweise in Basic und Forth zu arbeiten. Bei Pascal sind schließlich sogar fünf ROMs erforderlich; sie belegen die Adressenbereiche B000...BFFF (Forth: B000...CFFF) sowie 4000...7FFF, insgesamt also 20 KByte. Dafür ist eine selbstgebaute Erweiterungskarte mit mindestens fünf EPROM-Steckplätzen oder ein entsprechendes Modul der RM-65-Platinenserie von Rockwell erforderlich. Eine RAM-Erweiterung über die schon vorhandenen 4 KByte hinaus ist aber weder für Forth noch für Pascal nötig.

Forth: leistungsfähig, aber gewöhnungsbedürftig

Zu den beiden 4-KByte-ROMs für Forth gehört auch ein Handbuch, dessen Dicke nahezu die des AIM-Systemhandbuches erreicht. Kein Wunder: Im Vergleich zu Basic oder Pascal ist Forth ziemlich erklärungsbedürftig. Wir wollen hier allerdings keine Einführung in Forth geben, das wäre im Rahmen dieses Aufsatzes nicht möglich; vielmehr sollen die besonderen Eigenheiten des AIM-Forth herausgestellt werden. Trotzdem zuerst einmal die wesentlichen Vorteile von Forth: Die Sprache ist strukturiert; es gibt keinen GOTO-Befehl, statt dessen solche Anweisungen wie DO...UNTIL oder IF...THEN...ELSE. Forth arbeitet stackorientiert in Umgekehrt-Polnischer Notation (UPN, ähnl. wie die meisten HP-Taschenrechner); statt 2 + 3 = schreibt man in Forth: 2 3 +. Das führt zwar zu etwas ungewohnt aussehenden Programmen, erleichtert aber z. B. die Parameter-Übergabe zwischen Unterprogrammen ungemein. Und: Ein Forth-Programm ist nichts weiter als ständiges Neudefinieren von immer leistungsfähigeren Befehlen; das Problem wird stufenweise und ebenenweise in Teilprobleme aufgegliedert, jedes Teilprogramm erhält einen eigenen Namen, der wie ein neuer Befehl verwendet werden kann, und das Gesamtprogramm erhält schließlich selbst einen Namen, so daß es ebenfalls wie ein einziger Befehl verwendet werden kann.
Ebenso, wie es kein Standard-Basic gibt, weil jeder Hersteller seine eigenen Vorstellungen vom „optimalen" Basic hat, so gibt es leider auch kein Standard-Forth. Ein Quasi-Standard ist der von der „Forth Interest Group" vorgeschlagene Wortschatz; das AIM-Forth weicht von diesem jedoch in einigen Punkten ab (*Tabelle 1*).

Das AIM-Forth arbeitet (wie Basic, jedoch im Gegensatz zum AIM-Pascal) interpretativ. Das hat den Vorteil, daß man „von Hand" jederzeit den Stack prüfen und manipulieren oder Direktbefehle ausführen kann. Es wird grundsätzlich mit 8- oder 16-Bit-Zahlen gearbeitet; wie man auch Stringbefehle implementiert, wird ausführlich mit Beispielprogrammen am Ende des Handbuches erläutert. Überhaupt ist die umfassende Dokumentation des AIM-Forth eine seiner Stärken; Rockwell setzte hier die lobenswerte Tradition der hervorragenden AIM-65-Handbücher fort. (Allerdings: Ein kommentiertes ROM-Listing bekommt man diesmal nicht mitgeliefert!) *Tabelle 2* zeigt schließlich ein kleines Forth-Beispielprogramm; es bestimmt die hexadezimale Prüfsumme beliebiger Speicherbereiche mit 32 Bit Genauigkeit. Zur Bedienung (man erinnere sich an das UPN-Konzept von Forth): Man gibt die Byteanzahl und den Anfang des interessierenden Speicherbereichs sowie das Codewort CHECKSUM ein, jeweils durch einen Leerraum voneinander getrennt. Vorher kann man Forth mit HEX in den hexadezimalen Rechenmodus schalten.

Pascal, die „Lernsprache"

Wie von der deutschen Rockwell-Vertretung in Martinsried bei München zu hören war, wird Forth vielfältig in industriellen Applikationen eingesetzt, da es sich hervorragend nicht nur für arithmetische, sondern auch für Steuerungsaufgaben eignet (so enthält es z. B. einen

Tabelle 1: Unterschiede zwischen AIM- und FIG-Forth

Nur bei AIM, nicht bei FIG	Nur bei FIG, nicht bei AIM
–CR	+ORIGIN
.S	?LOADING
1–	BACK
2–	BLOCK-READ
2DROP	BLOCK-WRITE
2DUP	DLIST
?IN	DMINUS
?OUT	DR0
?TTY	DR1
ASSEMBLER	FLD
BAUD	INDEX
BOUNDS	LIST
C/L	MINUS
CHAIN	MOVE
CLIT	NEXT
CLOSE	OUT
CODE	POP
CURRENT	PUSH
DNEGATE	PUT
FINIS	R#
FLUSH	TRAVERSE
GET	TRIAD
HANG	X
NEGATE	
NOT	
PICK	
READ	
PR@	
SOURCE	
U<	
UABORT	
UB/BUF	
UB/SCR	
UC/L	
UEMIT	
UFIRST	
UKEY	
ULIMIT	
UR/W	
WRITE	

Tabelle 2: Errechnen einer Prüfsumme in Forth

```
<5>
AIM 65 FORTH V1.3
OK
(32 BIT CHECK-SUM)
(ADDR COUNT ---)
: CHECK-SUM
0 S->D    (ACCUM.)
ROT       (GET COUNT)
0 DO      (COUNT TIMES)
3 PICK    (GET BASE)
I +       (FORM ADDR.)
C@ S->D(32 BITS)
D+        (SUM IT)
LOOP
CR D.     (PRINT SUM)
DROP;     (BASE ADDR.)


Ausführung:

HEX OK
B000 1FFE CHECK-SUM
10D2A3 OK
```

eigenen 6502-UPN-Assembler, der sich in Forth-Sequenzen einbinden läßt). Im Gegensatz dazu wird Pascal bisher eigentlich vorwiegend für Lehrzwecke eingesetzt, um Programmieranfängern strukturiertes Programmieren und natürlich die Sprache Pascal selbst einzubleuen.
Pascal also mehr oder weniger als Selbstzweck? Keine Sorge, wir wollen hier nicht erneut mit Programmiersprachen-Ideologien anfangen, sondern nur auf die größere Bedeutung von Forth in der Applikations-Praxis hinweisen.
Im Vergleich zum Forth-Handbuch ist das Pascal-Handbuch relativ dünn; es enthält aber, wenn auch kein ROM-Listing, immerhin eine vollständige Übersicht der belegten Speicherzellen (z. B. in der Zero Page).
Das AIM-Pascal erhielt von seinem Vater Melvin E. Conway den Namen „Instant Pascal", weil es nach Einsetzen der fünf ROMs sofort einsatzbereit ist. Bei der ROM-Software handelt es sich ähnlich wie bei Basic um eine Art „Compreter", eine Mischung zwischen Interpreter und Compiler: Der mit dem Pascal-eigenen Texteditor zusammengestellte Quellentext wird in einen Zwischencode übersetzt (compiliert) und dieser dann interpretativ abgearbeitet.

Rockwells Pascal unterscheidet sich in einigen Punkten von dem von Prof. Niklaus Wirth an der ETH Zürich definierten Standard-Pascal. Kleinbuchstaben werden nicht verarbeitet, geschweifte Klammern werden durch runde Klammern und folgendem Sternchen ersetzt, NIL und FILE gibt es nicht, dagegen wurde OTHERWISE neu eingeführt. Strings der Länge N sind als Arrays aufzufassen (ARRAY [0...N] OF CHAR), wobei das nullte Element N enthält. Der Typ POINTER ist nicht implementiert, ebenso keine Varianten von Record-Typen. Auf SETs lassen sich die Operatoren +, −, * nicht anwenden. Und, zum Glück, wie manche Strukturierungs-Ideologen sagen werden, kann die GOTO-Anweisung nicht aus dem Block springen, in dem sie selbst steht.

Die von Rockwell implementierten Worte, Symbole und Funktionen gehen aus den *Tabellen* 4 und 5 hervor. Gerechnet wird beim REAL-Typ nicht nur (wie bei manch anderem Mikro-Pascal) mit 16 Bit Genauigkeit, sondern wie in Basic im Exponentialformat.

Tabelle 3: Reservierte Worte und Symbole im „Instant Pascal"

AND	END	NIL	SET
ARRAY	FILE	NOT	THEN
BEGIN	FOR	OF	TO
CASE	FUNCTION	OR	TYPE
CONST	GOTO	OTHERWISE	UNTIL
DIV	IF	PROCEDURE	VAR
DO	IN	PROGRAM	WHILE
DOWNTO	LABEL	RECORD	WITH
ELSE	MOD	REPEAT	

.	,	:	;		
+	−	*	/	:=	:
=	<=	>=	<>	<	>
(	)	(*	*)	[	]

Tabelle 4: Standard-Identifiers im AIM-65-Pascal

Konstanten:	FALSE	TRUE		
Typen:	BOOLEAN	CHAR	INTEGER	
	REAL	STRING	TEXT	
Funktionen:	ABS	FUNC	ROUND	SUCC
	ARCTAN	LN	SIN	TRUNC
	CHR	ODD	SQR	
	COS	ORD	SQRT	
	EXP	PRED	SUBR	
Proceduren:	READ	READLN	WRITE	WRITELN
	BREAK			

Komfort auf preiswertem System

Insgesamt gesehen bietet der AIM-65 somit zu relativ geringen Kosten die Möglichkeit, mit so leistungsfähigen Sprachen wie 6502-Assembler, PL-65, Microsoft-Basic, Forth und Pascal Programme zu entwickeln und in unterschiedlichsten Applikationen einzusetzen.
Man muß allerdings sehen, daß ein Gerät wie der AIM-65 mit einzeiligem LED-Display und zwanzigstelligem Thermodrucker schon von seiner Ausgabe her für höhere Sprachen nur begrenzt geeignet ist. Dies gilt insbesondere für die Ausgabe an sich übersichtlicher Pascal-Programme. Ein Video-Interface ist hier also von großem Vorteil. Für Assembler, PL-65 und die recht kurze Forth-Schreibweise dagegen stellen die AIM-eigenen Ausgabemedien dagegen keinerlei Beschränkung dar. Gerade in Forth lassen sich viele Probleme einfacher, übersichtlicher und kürzer lösen als in Assembler. Wenn der Forth-Interpreter auch etwa zehnmal langsamer als ein entsprechendes Assemblerprogramm ist, so ist er doch gegenüber Basic noch um vieles schneller, und auch seine Befehlsstruktur ist an typische Steuerungsaufgaben und industrielle Applikationen weit besser angepaßt.