Basic 64 — ein vielseitiger Basic-Compiler

Die Stärken dieses neuen Compilers sind ohne Zweifel seine vielen Funktionen. Obwohl es einige Mühe macht, sie alle auszunutzen, kommt aber auch ein Anfänger ohne Probleme mit diesem Compiler zurecht.

Basic 64 konnte in der Ausgabe 2/85 des 64'er nicht berücksichtigt werden, da er zu der Zeit noch nicht in einer endgültigen Version vorlag. Wir testeten damals die Compiler Petspeed, Austro Speed, den BASS- sowie den Exbasic Level II-Compiler. Damit ein Vergleich zu diesen Compilern möglich ist, wurde dieser Test mit den gleichen Testprogrammen und unter den gleichen Voraussetzungen vorgenommen.

Als Meßeinheit diente wieder das Programm EDDI aus der 64'er, Ausgabe 10/84, das eine Länge von 14 Blöcken hat. Wie schon einmal angesprochen, eignet sich dieses Programm deshalb so gut, weil es fast alle Bereiche der Programmierung abdeckt, nicht auf Compiler abgestimmt ist und außerdem einen Fehler aufweist. Dieser Fehler wird beim normalen Programmablauf nicht erreicht und hat auch sonst keinen nachteiligen Einfluß auf die Funktion von EDDI. Es dürfte jedoch interessant sein, wie Basic 64 auf diesen Fehler reagiert.

Mit EDDI wurde die Dauer und Fehlerfreiheit des Compilierens getestet.

Zur Messung der Geschwindigkeit verwenden wir die ebenfalls in Ausgabe 2/85 abgedruckten Benchmarks. Anhand der ermittelten Zeiten können Sie dann feststellen, auf welchen Teilgebieten der Compiler entweder besonders schnell oder langsamer arbeitet (Tabelle 1 und 2).

Der Basic 64-Compiler

Beginnen wir mit dem Lieferumfang des Compilers. Sie erhalten beim Kauf von Basic 64 einen Ringbuchordner, der ein Handbuch mittleren Umfangs und eine Diskette enthält.

Die Dokumentation erwies sich als gut. Sie führt in alle Funktionen des Compilers ein und erläutert ausführlich die vielen Zusätze, die Basic 64 besitzt.

Wie schon der Austro-Speed-Compiler, so ist auch der Basic 64 ein Zwerg in unserer Testreihe. Das Programm besteht aus ungefähr 90 Blöcken (22,5 KByte), in denen alle Funktionen untergebracht sind, auf die wir gleich noch zu sprechen kommen.

Es handelt sich bei Basic 64 um einen 2-Pass-Compiler, der im gesamten Aufbau etwas an Austro-Speed erinnert.

Die Bedienung erwies sich als sehr einfach und dürfte auch dem Neuling keine großen Schwierigkeiten machen. Wenn man Basic 64 mit den vier anderen Konkurrenten vergleicht, dann läßt sich eine Eigenschaft dieses Programms jedoch nicht verleugnen: Basic 64 ist mit Abstand der vielseitigste Compiler. Er bietet eine Fülle von Einstellmöglichkeiten und direkten Befehlen (Direktiven), so daß er sehr anpassungsfähig ist.

Es ist mit ihm zum Beispiel möglich, zwischen Adreßcode- und Assemblercode-Compilierung zu wählen und beide Modi sogar miteinander zu verbinden.

Weitere Möglichkeiten sind zum Beispiel freie Verschiebbarkeit des Compilats im Speicher, Nutzung von über 60 KByte Basic-Speicher, modifizierte Integer-Arithmetik und Simulation von »ON ERROR GOTO«-Befehlen.

Eine Eigenschaft, die Basic 64 ebenfalls stark von seinen Konkurrenten abhebt, ist sicherlich die Behandlung von Programmen, die während eines Programmablaufs nachgeladen werden (Overlay).

Für solche Overlay-Pakete besitzt Basic 64 einen speziellen Modus, der sogar Warm-Overlay, also Nachladen von Programmteilen ohne Löschen der Variablen erlaubt. Es wird so eine hohe Kompatibilität zum Basic V2.0 des Commodore 64 gewahrt.

Der einzig einschränkende Faktor zum Interpreter besteht in der Tatsache der Dimensionierung. Auch Basic 64 erlaubt, wie die meisten anderen Compiler, keine variable Dimensionierung (DIM A(N)), was durch die spezielle Variablenbehandlung bei Compilern bedingt ist.

Tritt eine solche Dimensionierung jedoch auf, wird der Benutzer nach der Größe des Feldes gefragt und diese eingesetzt. Hierbei ist zu beachten, daß auch der folgende Ausdruck

```
A = 10000 : DIM N(A)
```

eine variable Dimensionierung darstellt, obwohl A eindeutig definiert wurde.

Es ergibt sich ein recht guter Gesamteindruck, was die vielseitige Einsetzbarkeit dieses Compilers angeht. Bei Basic 64 fehlt deshalb auch die Verarbeitung von Erweiterungen nicht, die hier sogar speziell angewählt werden können.

Als Erweiterungen »versteht« Basic 64: Supergrafik 64 (und 64+), Simons Basic, Exbasic Level II und Basic 4.0.

Befehle dieser Erweiterungen werden vom Compiler, wie schon von den anderen Testkandidaten, im Non-Compiled-Code angelegt und dem jeweiligen Interpreter dann zur Ausführung übergeben.

Basic 64 im Test

Nun kommen wir zu unseren Testergebnissen.

Zum Compilieren von EDDI benötigte Basic 64 ungefähr 6,30 Minuten (5,55 im Adreßcode-Modus). Er ist damit nach Austro-Speed der zweitschnellste Compiler.

Damit die Voraussetzungen für einen Vergleich zwischen Basic 64 und den anderen Compilern gegeben sind, wurde das Programm EDDI im Assemblercode-Modus compiliert.

Die Länge des Compilats lag mit 40 Blöcken im Durchschnitt und bietet keinen Grund zur Beanstandung (im Adreßcode-Modus sind es wie beim Austro-Speed nur 32 Blöcke). Positiv fiel auch auf, daß während des Compilierens kein Diskettenwechsel notwendig ist.

Der Fehler in Zeile 1070 wurde von Basic 64 in Pass 2 erkannt, ohne daß jedoch nachteilige Folgen für EDDI daraus entstanden wären. Der Compiler arbeitete weiter und lie-

Hersteller	Austro-Speed	BASS	Exbasic Level II-Compiler	Petspeed	Basic 64
Preis (ca.)	298,–	198,–	298,–	149,–	99,–
Lieferumfang	1 Diskette 1 Handbuch	3 Disketten 2 dicke Handbücher	1 Diskette 1 Handbuch	1 Diskette 3 Seiten Einweisung	1 Diskette 1 Handbuch
Dokumentation	gut	ausgezeichnet	befriedigend	mangelhaft	gut
ungefähre Länge des Programms	63 Blocks	263 Blocks	290 Blocks	300 Blocks	90 Blocks
Anzahl PASSES	2	2 + 2 Assembler (wird nicht mitgeliefert)	2 + 2 Assembler (integriert)	4	2
Programmschutz möglich?	ja	nein	nein	nein	nein
Integer Arithmetik	ja	ja	ja	ja	ja
Erweiterungen möglich?	ja	ja	ja	nein	ja
variables DIM möglich?	ja	nein	nein	nein	nein
Automatisches DIM auf 11 Elemente	ja	nein	nein	ja	ja
Compilierdauer EDDI (14 Blocks)	3 min	7,10 min + 9 min ASSI	12 min	7,30 min	6,24 min/sec
Diskettenwechsel beim Compilieren	nein	ja, 2mal	ja, 4mal	nein	nein
Anzahl der erzeugten Files	2	10	1	2	1 (2 nach Wunsch)
Länge des Compilats (EDDI)	32 Blöcke	39 Blöcke	39 Blöcke	42 Blöcke	40 m-code 32 P-code
Compilertyp	–	Adreßcode + Assemblercode	Assemblercode	Assemblercode	Assemblercode + Adreßcode

Tabelle 1. Fünf Compiler im Vergleich

ferte ein vollkommen funktionsfähiges Compilat.

In Tabelle 1 sind alle Testergebnisse noch einmal zusammengefaßt.

In Tabelle 2 sehen Sie die Ergebnisse des Benchmark-Tests. Hier zeigt Basic 64 durchschnittliche Werte, die in etwa denen des Austro-Speed entsprechen.

Es muß an dieser Stelle der Vollständigkeit halber jedoch noch erwähnt werden, und das gilt auch für die Compiler-Tests in der Ausgabe 2/85, daß die Geschwindigkeiten des Benchmark-Tests unter Umständen ein unklares Bild liefern können. Die Testprogramme sind nämlich generell nicht an einen Compiler angepaßt und können demzufolge nicht dessen spezielle Eigenschaften ausnutzen. Durch optimierende Eingriffe in Programme und durch Anweisungen an die Compiler können deshalb in der Regel noch höhere Leistungen erzielt werden.

Basic 64 im Überblick

Basic 64 zeigte in diesem Test Eigenschaften, die durchweg als positiv zu sehen sind. Die Dokumentation ist sehr ausführlich und gibt dem Leser viele Informationen über die Arbeitsweise dieses Compilers.

Der Compiler selbst ist sowohl für Anfänger (laden, starten, fertig) als auch für diejenigen mit viel Programmiererfahrung geeignet. Der Erfahrene wird viele Möglichkeiten finden, seine Programme auf die Stärken des Compilers hinzutrimmen. Dabei wird er unterstützt durch die vielen wählbaren und einstellbaren Funktionen des Basic 64.

Die Geschwindigkeitstests ergaben durchschnittliche Werte, die in etwa denen des Austro-Speed entsprechen. Durch kleine Änderungen im Basic-Programm und durch Wahl anderer Basic 64 Funktionen kann die Geschwindigkeit jedoch noch gewaltig erhöht werden, so daß in günstigen Fällen sogar die Zeiten des Petspeed überboten werden können.

Insgesamt gesehen also viel Leistung für 99 Mark.

(Karsten Schramm/gk)

		Basic	Exbasic + BASS	Pet-speed	Austro-Speed	Basic 64 P-Code	Basic 64 M-Code
Test	1a Einlesen (100 Werte)	1,53	0,67	0,50	0,46	0,58	0,5
Test	1b Sortieren	64,16	24,85	8,23	16,25	18,88	15,83
Test	1c Anzeigen	0,55	0,67	0,10	0,33	0,37	0,33
Test	1 Gesamt (a + b + c)	66,25	26,18	8,83	17,05	19,83	16,67
Benchmark	1 FOR NEXT	1,83	1,01	0,28	1,10	1,05	1,05
	2 430 K = K + 1 440 IF K <1000 THEN 430	13,22	2,16	0,72	1,38	1,86	1,22
	3 A = K/K x K + K – K	11,25	4,70	5,54	4,38	5,47	4,66
	4 A = K/2 x 3 + 4 – 5	12,00	6,51	6,42	5,32	5,18	4,48
	5 GOSUB RETURN	16,65	0,45	0,18	0,15	0,31	0,3
	6 FOR L = 1 TO 5:NEXT L	14,47	6,34	1,93	6,50	6,65	6,27
	7 FOR L = 1 TO 5:M(L) = A:NEXT	24,85	8,80	2,50	3,95	5,74	4,3
	8 A = K!2 B = LOG(K) C = SIN(K)	112,30	102,19	93,25	101,98	103,73	102,36
Gesamtzeit (Test 1 + Bench 1 bis 8)		390,86 ≙100%	198,53 ≙50,79%	145,52 ≙37,23%	172,96 ≙44,25%	182,02 ≙46,57%	167,72 ≙42,91%

Tabelle 2. Die fünf Compiler im Zeitvergleich. Im Test 1 wurden mit der RND-Funktion 100 Zeichen ermittelt, sortiert und nebeneinander ausgegeben. Die Benchmark-Tests 1 bis 8 waren so aufgebaut, daß die Zeit für die angegebenen Befehle selbst ermittelt werden konnten. Deshalb ist die Gesamtzeit nicht identisch mit der Summe der einzelnen Testzeiten. Jeder Befehl (Benchmark 1 bis 8) wurde 1000mal durchgeführt.