Speed difference between Basic 1.0 and 1.1

compiled from newsgroups <comp.sys.oric> by André C.

Fabrice F.: Hello Anders, I've just read your question in the Ceo-Mag about a 50% speed difference between Basic 1.0 (Oric-1) and 1.1 (Atmos): You were wondering if it was an artefact of Euphoric or something else... Well, don't worry, it's not an artificial behaviour of Euphoric, Basic 1.1 was really optimised in some places, compared to 1.0 (which was really close to the original Microsoft Basic-65... not so close as the direct port of Basic-65 to the Microtan-65, but still very close... :-).

Anders C.: It's always interesting to find out that newsgroup messages find their way into magazines... [Basic 1.1 versus Basic 1.0] Interesting. Back in the 80'ties, I cannot recall reading anywhere in the reviews that Basic seemed to have been speeded up; only bug fixed. BYTE had in 1977 a series of simple benchmarks, which sometimes were reused to give a rough estimate of Basic speed. It might be worthwhile to run those on the two Oric models and compare the results with other Basics if it has not been done before.

Steve M.: I did run some benchmark tests on the Atmos and found that it was not much faster than the Oric-1. The Orics are painfully slow as compared to other machines, but does beat the Speccy on more complicated functions ;o) Some of the basic speed was altered by that poke that effects the number of times the keyboard is scanned IIRC. The Atmos keyboard repeat and cursor flash is quicker. Under Sedoric, this is quicker again.

Anders: I just run through the eight benchmarks, but I cannot claim my results to be very reliable: 1) They were run in Euphoric (Atmos mode), not a real Oric computer. 2) The host machine is a 200 MHz Windows 98; not sure if it is enough. 3) Since I couldn't locate a simple timer, I used my C64 as a timer. However, below are the results compared to premeasured values:

The simple loops seem to take an awful lot of time, probably an emulator artefact? On the other hand, the trigonometric test (#8) is reasonably fast and lifts the machine a number of positions compared to the others. Here are the incremental benchmark listings. The listings marked (new) obviously should be written from scratch, otherwise only add or change the mentioned lines.

```
BENCHMARK 1 (new)
100 REM TEST 1
110 PRINT «START»
120 FOR K=1 TO 1000
130 NEXT K
500 PRINT «STOP»
550 END
```

```
BENCHMARK 2
120 K=0
130 K=K+1
190 IF K<1000 THEN GOTO 130
```

```
BENCHMARK 3
140 A=K/K*K+K-K
```

```
BENCHMARK 4
140 A=K/2*3+4-5
```

```
BENCHMARK 5
150 GOSUB 600
600 RETURN
```

```
BENCHMARK 6
125 DIM M(5)
155 FOR L=1 TO 5
160 NEXT L
```

```
BENCHMARK 7
157 M(L)=A
```

```
BENCHMARK 8 (new)
100 REM TEST 8
110 PRINT «START»
120 K=0
130 K=K+1
140 A=K^2
150 B=LOG(K)
160 C=SIN(K)
170 IF K<1000 THEN GOTO 130
180 PRINT «STOP»
190 END
```

	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8	Avg
Oric est.	1.8	15.7	25.5	27.5	33.0	45.5	68.5	140	44.7
BBC (B ?)	0.6	3.2	8.1	8.8	9.9	14.3	21.9	48	14.3
Acorn Atom	0.5	5.1	9.5	10.8	13.9	19.1	31.1	92	22.8
VIC-20	1.4	8.3	15.5	17.1	18.3	27.2	42.7	99	28.7
Apple II	1.3	8.5	16.0	17.8	19.1	28.6	44.8	107	30.4
Dragon 32	1.6	10.2	19.7	21.6	23.3	34.3	50.0	129	36.2
SVI-328	1.6	5.4	17.9	19.6	20.6	30.7	42.2	236	46.7
ZX81 (fast)	4.5	6.9	16.4	15.8	18.6	49.7	68.5	229	51.2
ZX Spectrum	4.8	8.7	21.1	20.4	24.0	55.3	80.7	253	58.5
Atari 600XL	2.2	7.2	19.1	22.8	25.8	37.6	58.3	412	73.1
TI-99/4A	2.9	8.8	22.8	24.5	26.1	61.6	84.4	382	76.6

Fabrice: Here is the timer:

```
10 T1=DEEK(#276)
...
200 T2=DEEK(#276):IF T1<T2 THEN T1=T1+65536
210 PRINT (T1-T2)/100
```

The emulator counts every cycle and synchronises either with the sound DMA (if you have a SoundBlaster card) or with the host clock. Just try this: WAIT 6000 with your clock in your hand. If it takes 1 minute, the emulator is perfectly synchronized...

Anders: Cool. I was considering something like that (or rather making calls directly to the ROM according to the disassembly I found). After verifying the WAIT statement takes exactly 60 seconds in the emulator, I rerun the second benchmark and got the same results; 15.6 seconds for Oric Atmos or 17.3 seconds for Oric-1 (compared to VICE xvic: 8.0 seconds, x64: 9.6 seconds). It looks like the Oric Basic is one of the slowest ever when it comes to handling an empty increase-and-compare loop. Was there any alternative Basics for Oric, i.e. the BBC Basic which I know floats around even in Z80 versions? I don't know if the hardware on Acorn's BBC allowed faster execution (after all it is a 1 or 2 MHz 6502 too), but it seems silly that two 6502-based systems would differ that much in execution speed — imagine Acorn using a marketing slogan «60% faster than Oric».

Fabrice: Right, it comes partly from the fact all computations are done using floating point values (ol' microsoft floating point format: 5 bytes floating point values, i.e. a 4-bytes-mantissa). Using integer variables (appending a '%' sign to the variable name) is even worse: for each computation, they are first converted to floating point, then the computation is done using floating points, and the result finally converted back to integer. Also, about 20% of the CPU time is consumed by the interrupt routine that polls the keyboard... Oric Basic is not fast, that is a fact. :-(Acorn's Basic is sure a much better Basic, it evolved during several years to reach a great level... (in comparison, Oric Basic only went through two versions : A buggy 1.0 release, and a somewhat debugged and slightly improved 1.1 release...). To be honest, when the company crossed the channel, a new compiled Basic was developed in France, it is mostly compatible at the source level. So, if you try your benchmarks on the Telestrat, you will get much better results... And if you replace your loop with something like:

```
110 COUNT 1000
...
190 UNCOUNT
```

you will find it damn' fast :-) PS Do the VIC and the C64 run at the same clock frequency ?

Anders: [floating point & integer variables] Yep, the same thing happens in Commodore Basic (derived from a one-time licensed copy of Microsoft Basic, which M$ later regretted as they could have received lots more license money if they had been paid per user). However, it takes less space to store the integer variables. *[keyboard]* Glbl. Shouldn't the VIA do all the keyboard decoding? *[Acorn's Basic]* Well, Atom Basic and Electron/BBC Basic look very much different on the UI (but might share code inside). All the benchmarks were done in 1983/84, so it is not like comparing an optimised 2003 Basic with something written twenty years ago. *[VIC and the C64 clock frequency]* Almost the same; the more advanced C64 traditionally has an effective frequency of 0.98 MHz (PAL) when screen is lit while the VIC is about 1.01 MHz. The numbers may differ a little in NTSC land. If the screen is closed, the C64 reaches 8.9 seconds in VICE, which is not quite the VIC speed but a reasonable speed-up.

Answer from Fabrice: [Keyboard] Well, due to the way the Oric was designed, one port of the VIA is shared by the printer and by the sound generator (PSG)... This is one of the most dubious design of the Oric: the PSG is not memory-mapped, you access it through the VIA, with lots of operations in order to handle the control lines of the PSG, and then the I/O port of the PSG is in turn connected to the keyboard columns... Each time you want to access a keyboard column, you spend a lot of clock cycles... *[Acorn's Basic]* I was referring to the latest 6502 versions of Acorn Basic, that has both high-level structures (control structures, procedures, etc.) and in-line assembly... Yet, twenty years ago, BBC Basic was already quite recognised...

Again from Anders C.: [PSG] Yes, I've noticed this before when I was considering how much work it would be to port one of my music players to Oric. *[keyboard]* And me who thought the wiring together of joystick port 1 and keyboard was bad enough on the C64.

Answer from Steve: [alternative Basics for Oric] .../... There were one or two Basic toolkits around that added some of those missing commands Programs like Super Extender and Basic Plus, but I'm not aware of complete rewrites except perhaps for Sedoric? The Sedoric Basic has added commands and runs a little faster but I am not sure how much is altered. I have BBC Basic for the Einstein computer, which is both fun and frustrating because there are a number of commands that don't work. Screen commands etc are so different that BBC programs will not run on an Einstein with BBC Basic, which is disappointing.

Question from Fabrice: [Sedoric Basic] Actually, it runs significantly slower... The interpreter routine that fetches consecutive bytes is extended in order to cope with file names used as commands and to differentiate those file names with assignment variables, and this slows down the whole interpretation process... I am considering extending my Evolution Basic to a semi-compiled form (do not know if the 'semi' prefix has any English meaning :-) This semi-compilation would mainly transform sequential accesses in direct accesses (e.g. variable access, line access in a GOTO or GOSUB) and still keep a very compact tokenised form (so, still interpreted but about as fast as Forth)...

From Ventzislav T. : [keyboard] You can speed up the Basic interpreter by changing the time between 2 keyboard handling interrupts: DOKE #306,65535 To set the value back to normal: DOKE #306,10000 For timing right on the Oric you can use the Oric Real time clock, which is somewhere else in this newsgroup

archive. To my knowledge, the VIC's main CPU is clocked at 1.01 MHz, while the C64's CPU is clocked at 985 KHz.

Again from Fabrice: I've re-run your tests on the Microtan-65, Oric-1 and Atmos... Here they are for your collection (see below). You can see that the results are the same you already provided for the Atmos (I have used timers to have a more precise timing, this required additional variables, so I have been cautious to initialise variables in the same order you used them, and to provide the same amount of lines for the lookup occurring in GOTOs and GOSUBs). The only difference with your benchmarking is with BM8, because I've replaced LOG by LN on the Oric-1 and Atmos (LOG is the decimal logarithm, this requires an additional division :-) Except that, I have strictly followed your syntax, even if FOR loops go faster if you don't precise the index variable in the NEXT statement... But, I'm not computing an average, because your average is a «double-strange» average... :-) I prefer to present your individual tests in this way:
BM1 measures the FOR loop
BM2 measures a simple incrementation and loop
BM3-BM2 measures standard arithmetics
BM4-BM3 roughly measures the cost of converting decimal numbers to float binaries (Sinclair machines have a negative value here, because they store the floating point value in the code)
BM5-BM4 measures GOSUBs
BM6-BM5 measures small FOR loops (i.e. the initialisation is very costly)
BM7-BM6 measures array access
BM8-BM2 measures transcendental functions
So, actually, BM7 alone is already a sort of average of BM2, BM3-BM2, BM4-BM3, BM5-BM4, BM6-BM5 and BM7-BM6... Anyway, I was interested in the Microtan-65 timings, because it really represents the original Microsoft Basic 65, from which the Oric Basics are derived. The Microtan-65 runs at 750 kHz only, below I'm providing timings of an hypothetic 1 MHz Microtan to compare with the Oric Basics:

BM1 : 1MHz Microtan 1.4
Oric-1 1.8
Atmos 1.6

Comment: the inner interpreter loop of the Oric-1 introduces some new features, not found on the original Basic-65: 1) New command separators have been introduced (ELSE and the quote). 2) Ctrl-C detection between two executed commands. 3) Ability to CONTinue the program after a Ctrl-C (the TXTPTR is frequently saved). 4) Tests for TRACEd execution (TRON, TROFF). 5) And of course also, the VIA timer interrupt that does a lot of handling, including keyboard polling... I'm not sure about the better result of the Atmos, I think I comes partly from the fact that the mantissa-shifting-routines use the ROR and ROL instructions which were not used before (the first prototypes of 6502 did not have these instructions, so Microsoft didn't use them in its Basic65, and Oric did not notice that until the Atmos), and partly because of slightly faster interrupt routines...

BM2: 1MHz Microtan 9.6
Oric-1 17.1
Atmos 15.2

That is a complete non-sense for me! This is what you noticed immediately: This simple loop is incredibly slow on the Orics! I have to dig into this, I cannot understand!

BM3-BM2: 1 MHz Microtan 8.9
Oric-1 11.9
Atmos 10.2

Ok, not very good, those VIA interrupts are to blame...

BM4-BM3: 1 MHz Microtan 2.3
Oric-1 2.4
Atmos 2

Yeap, I quite remember that the ASCII to float routine has been optimised a little...

BM5-BM4: 1 MHz Microtan 1.4
Oric-1 6.6
Atmos 5.6

Damned! Again an not-understandable result! Why are these GOSUBs so slow?

BM6-BM5 1 MHz Microtan 10.2
Oric-1 13.8
Atmos 12.6

	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8	Avg
BBC (B ?)	0.6	3.2	8.1	8.8	9.9	14.3	21.9	48	14.3
Acorn Atom	0.5	5.1	9.5	10.8	13.9	19.1	31.1	92	22.8
VIC-20	1.4	8.3	15.5	17.1	18.3	27.2	42.7	99	28.7
Apple II	1.3	8.5	16.0	17.8	19.1	28.6	44.8	107	30.4
Dragon 32	1.6	10.2	19.7	21.6	23.3	34.3	50.0	129	36.2
SVI-328	1.6	5.4	17.9	19.6	20.6	30.7	42.2	236	46.7
ZX81 (fast)	4.5	6.9	16.4	15.8	18.6	49.7	68.5	229	51.2
ZX Spectrum	4.8	8.7	21.1	20.4	24.0	55.3	80.7	253	58.5
Atari 600XL	2.2	7.2	19.1	22.8	25.8	37.6	58.3	412	73.1
TI-99/4A	2.9	8.8	22.8	24.5	26.1	61.6	84.4	382	76.6
Microtan65	1.9	12.8	24.7	27.8	29.6	43.2	68.9	243.0	
Oric-1	1.8	17.1	29.0	31.4	38.0	51.8	77.8	230.1	
Atmos	1.6	15.2	25.4	27.4	33.0	45.6	68.5	136.5	

Ok, still the same factor...

BM7-BM6:	1 MHz Microtan	19.3
	Oric-1	26
	Atmos	22.9

Here we go again...

BM8-BM2:	1 MHz Microtan	172.6
	Oric-1	152.3
	Atmos	68.0

Huh?! What is that?
Are they computing the same thing? This is what you spotted too... I am sure they have cheated with the precision (fewer terms in the limited development) ;-)

Anders: [«double-strange» average] Yep, I thought it was killing some of the value from the benchmark, but my source already had performed the averages on the other machines. Maybe because it is easier to compare one value than eight, even if that one value does not represent anything useful.
[BM2 incredibly slow on the Orics] Maybe all sorts of calculation tests are scanned through somehow? At least it is a good example of a programming style to avoid if one can.
[BM5-BM4 (GOSUB added] Ah, you are making incrementary comparisons too. In that case, it might be interesting that ZX Spectrum as the second worst adds 3.6 seconds between these two tests.
[BM8-BM2] A few comparisons: BBC 44.8, VIC-20 90.7, ZX Spectrum 244.3. The article said trigonometric algorithms tend to be dead slow, so it might have been vast improvements in Oric Basic 1.1 which was out a little later than the original benchmarks were done.
Hm. I might try these benchmarks on Plus/4 and C128 which both have other Basics that VIC-20/C64. Since the C128 also runs in 2 MHz, in theory it should beat both the old BBC and the original IBM PC (which I omitted from the benchmark stats together with some other odd computers :-).

Again from Fabrice: [easier to compare one value than eight] Yeap, that is always the problem with benchmarks... Does a small benchmark mean anything? People tend to prefer to compare machines on full-size applications nowadays...
[BM2] Ah, ok, I see... The original Microsoft Basic only accepted a line number after a GOTO... The Oric Basic accepts any expression after a GOTO, e.g. GOTO 100+N*10 So, the line number in the benchmark is first converted to float, then back to an integer... Sure, nobody would write this on the Oric. We have a nice REPEAT...UNTIL structure, so if BM2 was measured on

```
120 REPEAT
130 K=K+1
190 UNTIL K=1000
```

Then the BM2 result would fall to 11.5 for the Oric-1, and 10.4 for the Atmos.
[BM5-BM4 GOSUB added] Yes, I remember my ZX81 also allowed these computed GOTO/GOSUBs, because they lacked a ON...GOTO/GOSUB structure. But the Oric does have this ON...GOTO/GOSUB, so maybe Oric only added the computed GOTO/GOSUB to gain some room in the ROM, by calling the standard expression evaluator...
[BM8-BM2] Oups, sorry, I substracted BM8-BM7 instead of BM8-BM2... The above should read:

BM8-BM2:	1 MHz Microtan	221.2
	Oric-1	213.0
	Atmos	121.3

I have checked, the code of ^, LN (LOG on the Microtan) and SIN are exactly the same on these 3 machines. What has changed on the Atmos is the use of ROR and ROL instructions when shifting a mantissa (the original Microsoft Basic did not use them), and these shiftings are mostly responsible for the duration of floating-point additions (which in turn are largely used in the polynomials of these transcendental functions).

Again from Anders: [The Oric Basic accepts any expression after a GOTO] So does Sinclair Basic, but not Commodore Basic. Not sure about the other (faster) Basics. Sometimes this could be resolved by ON GOTO, but it requires a well lined (eh) program.

Again from Fabrice: Ah, I forgot to run these benchmarks on the Telestrat, here they are below:
Telestrat's Hyper-Basic is a Basic compiler, this is why the first loop is this fast (there's even a COUNT 1000: ... : UNCOUNT structure that is compiled in a few machine code instructions, but I have not used it here). Of course, GOTOs and GOSUBs cost nearly nothing, but curiously, it seems that literals are not stored in binary form (BM4)... Anyway, one problem with Telestrat's Hyper-Basic is that the code is much less compact than the interpreted one... So, I'm experimenting a hybrid interpreter, with compiled address variables (not need to look them up), and literals in binary form... For now, I have BM2 on the Atmos running in 4.9s (instead of 15.2). I am still amazed at the results of the Acorn computers... surely they only use integers in FOR loops, don't they?

Anders: Telestrat is what, a 2 MHz 6502 or something?
[Acorn & integers in FOR loops] I do not know what they are doing right, but BBC emus should be easy to get and try if you don't have the real thing. Worth noticing is that even the original IBM PC (at whatever clock frequency it used) with Rom Basic was slower than BBC.

Fabrice: Nope, 1 MHz only :-) But the bundled Basic is a compiler...

	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8
Microtan65	1.9	12.8	24.7	27.8	29.6	43.2	68.9	243.0
Oric-1	1.8	17.1	29.0	31.4	38.0	51.8	77.8	230.1
Atmos	1.6	15.2	25.4	27.4	33.0	45.6	68.5	136.5
Telestrat	0.5	3.6	11.4	13.2	13.4	18.7	26.3	109.6