KC-BASIC

Ein wesentliches Unterscheidungsmerkmal der verschiedenen Z9001-Varianten ist die jeweils verwendete KC-BASIC Version. Während beim Z9001 und KC85.1x noch das BASIC von Kassette geladen werden musste oder als separates Steckmodul zugekauft werden konnte, ist beim KC87 das BASIC bereits im Grundgerät enthalten.

Geschichte

Das Basic basic.zip ist eine Entwicklung des FTP Dummerstorf/Rostock (s.a. „Dr.-Ing. Jürgen Lübcke, Dipl.-Ing. Reinhard Vilbrandt: BASIC-Interpreter für K1520, rfe 1/1983, S. 14-16“). Ursprünglich entstand dieses BASIC als Mix aus dem SPECTRUM und dem Microsoft-8K-BASIC, welches für viele Computer der frühen 80er Jahre verfügbar war.

Dr. Keller hat mir Januar 2008 folgende Informationen mitgeteilt:

Woher kommt das BASIC der DDR-Kleincomputer?

Im Sommer 1983 erhielt Messelektronik (MKD) die Aufgabe, für eine sinnvolle Version für den Heimcomputer (damaliges Ziel) zu sorgen. Diese Aufgabe wurde mir übertragen. Wir führten, gemeinsam mit Kollegen von RED, Recherchen durch und nahmen zunächst 5 Versionen in die engerer Auswahl. Nachdem auch die Hardware-Rahmenbedingungen und die Terminlage für den Computer klar wurden, wurde die Auswahl auf 2 Versionen eingeschränkt:

- Version von TUR Dresden, dort komplett selbst entwickelt, sehr gute Version, mit Subroutine-Technik und mit MAT-Befehlen
- Version des Instituts in Dummerstorf, die eine Mischung aus dem Sinclair-Basic und einer einfachen Version des Microsoft-Basic darstellte

Letztere Version wurde im August 1983 vor Ort begutachtet (Keller, Vilbrandt) und anschießend für MKD beschafft. Ausschlaggebend für die Entscheidung war der deutlich geringere Speicherplatzbedarf (< 8 kByte) gegenüber der Version von TUR (ca. 13 kByte). Außerdem wurde wohl auch die etwas größere Verbreitung der Microsoft-Version beachtet. Die TUR-Version orientierte sich mehr am hp-Basic, was später wieder im VisualBasic Beachtung fand.

Nach Übernahme der Dummerstorfer BASIC-Version geschah etwa folgendes:

- Die Software stellt sich als Mischung (Sinclair, MS) heraus, wobei einige Passagen offensichtlich rückübersetze MS-Software waren.
- Etwa 6K-Byte der Software (Kern) wurde so gelassen der Rest wurde überarbeitet/ergänzt/verbessert.
- Insbesondere wurden einige Fehler beseitigt, die schon in der MS-Version steckten und wesentliche Sprachkomponenten wurden angepasst (bei MKD durch Keller/Busch)
- In die Spracherweiterung flossen Gedanken mehrerer Mitarbeiter (auch von RED) ein, vor allem aber die Erfahrungen, die bei MKD mit dem Tischrechner hp9548B (IBM-PC-Vorläufer)(vp: sicherlich HP 9845B) gesammelt wurden (von diesem BASIC wurden einige Sprachkomponenten übernommen.)
- In Ergänzung zu dem 6K-Kern wurde dann eine 8K-Version erarbeitet, die im Wesentlichen nicht

mehr geändert wurde.
- Die Erweiterung (zu 10K-Basic) enthielt dann hardware-bezogene Ergänzungen, die sich mehrfach geringfügig geändert haben, ohne die Kompatibilität zu beeinflussen
- Im Vergleich zu anderen Heimcomputern (Sinclair, C64, Atari) ist festzustellen, dass bei der robotron-Version strikt auf die Schnittstellen des BIOS und BDOS geachtet wurde, wie sie sich bei CP/M durchzusetzen begannen.

Folgende Besonderheiten sind noch zu bemerken:

- der Kassettentreiber wurde bei RED bearbeitet, wobei die zugehörige Hardware bei MKD verbessert, d.h. stabilisiert wurde.
- der fertige Kassettentreiber wurde dann (etwa 12/83 oder 01/84 an Mühlhausen übergeben, so dass zur Frühjahrsmesse 84 bereits die Kompatibilität bezüglich der Datenübernahme gesichert war.
- später gab es durch Hobbyisten (z.B. im ZKI) schnellere und stabilere Kassettentreiber, die auf der jeweils gerätespezifischen Hardware aufsetzten. Häufig wurden diese auf Einzelrechnern auch genutzt, sofern kein Datenaustausch erforderlich war.
- Auch darüber hinaus wurden im BASIC seitens MKD wesentliche Erweitungen, Verbesserungen und Fehlerkorrekturen vorgenommen, die beiden Computern (robotron und Mühlhausen) zu Gute kamen.

Der 8k-Kern wurde (meines Wissens) fast identisch von Mühlhausen genutzt. Die Unterschiede in den Sprachkomponenten wurden in die 2k-Erweiterungen gesteckt.

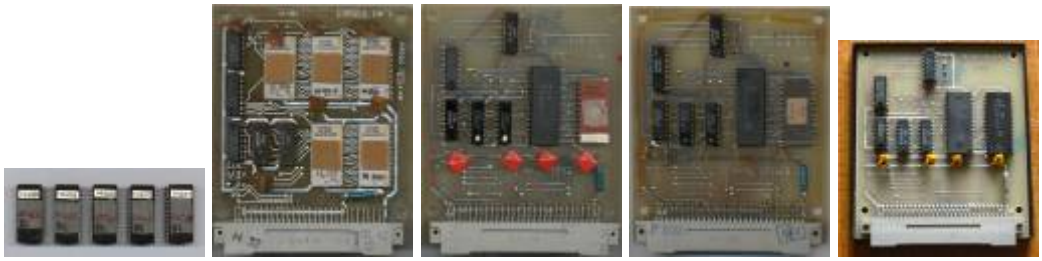
Wann und durch wen wurde beschlossen, das BASIC für alle DDR-KC (Dresden und Mühlhausen) zu nutzen?

Bereits zur 1. oder 2. großen gemeinsamen Beratung der Entwicklungsvorhaben in Mühlhausen (Treffen der beiden Entwicklungskollektive) wurde mit dieser Zielstellung diskutiert. Das heißt, die entsprechende Vorgabe (vom Zentralrat der FDJ oder direkt vom ZK der SED??) wurde schon vorher an beide Betriebe übermittelt. Da die Verantwortung für die Sicherung der Sofwarekompatibilität an robotron übertragen wurde, landete diese bei Messelektronik (MKD) und dann bei mir (Dr. Keller). Die Vorgabe gab es also, bevor das konkrete BASIC ausgewählt war. Die Kompatibilitätsforderungen bezogen sich vor allem auf:

- Kassettentreiber
- Kern-BASIC

Es war klar, dass es wegen der (gewollten) unterschiedlichen Hardware der Rechner auch Unterschiede im Sprachumfang geben würde.

BASIC-Varianten



von links nach rechts: EPROMs M497 bis M501 (BASIC84), aus dem Nachlass der Entwickler (Besitzer: L.E.), ROM-BASIC mit 5x2K EPROM (BASIC85), mit 8K-PROM + 2K-EPROM (BASIC85), Plotter-Modul (BASIC86), Plotter-Module der TU Karl-Marx-Stadt (BASIC86).

Von Anfang an wurde das BASIC modular konzipiert. Es gibt einen 8KByte großen geräteunabhängigen **Kern** und eine 2KByte große Erweiterung, die neben zusätzlichen BASIC-Befehlen die geräteabhängigen Treiber enthält (Bildschirm, Tastatur, Kassette). Diese Erweiterung wurde mehrfach überarbeitet, während der Kern unverändert blieb. Aus Kostengründen(! L.E.) wurde aus diesem Kern nur ein PROM entwickelt, der für **alle KCs** (Mühlhausener und Robtron-KC) genutzt werden konnte, der PROM DS2364D45 BM600. Deshalb enthält dieser PROM z.B. '7F 7F REBASIC' für den KC85/2..4. In diesem Kern gibt es eine definierte Softwareschnittstelle zur Erweiterung.

Für den Z9001 (und Nachfolger) gibt es das RAM-Basic der ersten Z9001 und diverse ROM-Module. Die ROM-Versionen unterscheiden sich nur unwesentlich voneinander. Es gibt 3 wesentliche Varianten (BASIC84, BASIC85, BASIC86), die nacheinander erschienen. Diese Unterschiede stecken i.W. nur in der Erweiterung.

Die folgend aufgelisteten Versionen des BASICs sind im Umlauf. Einfache Unterscheidungsmerkmale sind die Startmeldung („MEMORY SIZE“, „MEMORY END“ und der RENUMBER-Befehl. Im BASIC86 heißt er RENUM. Alternativ kann man das Modul aufschrauben und die enthaltenen ROMs identifizieren. Die Übersicht am Textende erleichtert die Zuordnung. Es gibt Basic-Module mit 5 Stück 2K-EPROMs, Basic-Module mit dem 8K PROM U2364BM600 und einem 2K-EPROM sowie Module mit 2 maskenprogrammierten PROMs.

- **RAM-Basic 1984 (BASIC84):** Auf Kassette R0111 enthalten.
- **Basic-Modul 1984 (BASIC84)**: Kern EPROMs M497 bis M500, Erweiterung M501;
  kaum verbreitet, entspricht dem RAM-Basic, hat RENUMBER
- **Basic-Modul ab 1985 (BASIC85)**: Kern EPROMs M507 bis M510, Erweiterung M511;
  meldet sich mit MEMORY SIZE, hat den Befehl RENUMBER
- **Basic-Modul ab 1986 (BASIC85P)**: Kern PROM DS2364D45 BM600 , Erweiterung M511;
  meldet sich mit MEMORY END, sonst genau wie genau wie Basic-Modul ab 1985 (BASIC85)
- **KC87.1x (BASIC85)**: Kern PROM DS2364D45 BM600, Erweiterung PROM DS2364D45 BM602;
  genau wie obiges BASIC-Modul
- **Plotter-Modul (BASIC86)**: Kern PROM DS2364D45 BM600, Erweiterung EPROM M122. Es gibt auch inhaltlich gleiche Module mit Kern PROM DS2364D45 BM600, Erweiterung PROM DS2364D45 BM608;
  meldet sich mit MEMORY END, hat den Befehl RENUM und und enthält zusätzliche Grafikbefehle.
- **KC87.2x/3x (BASIC86):** Kern PROM DS2364D45 BM600, Erweiterung PROM DS2364D45 BM608;
  enthält das Plotter-BASIC im System.
- außerdem ist noch ein leicht anderes Plotter-Modul-Basic aufgetaucht („MEMORY END“, BASIC-86, und einige wenige Unterschiede zum EPROM M122)

Die praktischen Unterschiede zwischen RAM-BASIC und dem BASIC-Modul sind marginal (also zwischen BASIC84 und BASIC85), die Unterschiede zum neuen BASIC-Modul (Plotter, KC87.2x) (BASIC86) betreffen einen erweiterten Sprachumfang (es gibt zusätzliche Grafikbefehle zur Ansteuerung des Plotters bzw. der Vollgrafik; hier wird aber noch ein zusätzlicher Treiber benötigt, s. Plotter), der Befehl RENUM statt RENUMBER und außerdem einen geänderten Kassettentreiber.

Hinweis: Leider wird beim BASIC86 beim Speichern auf Kassette ein unnötiger und mit zufälligen Zeichen belegter Startblock mit Blocknummer 0 geschrieben. Dieser Block bringt manches

Kopierprogramm aus dem Tritt, außerdem verbraucht er zusätzlichen Platz auf der Kassette und das Schreiben und Einlesen dauert auch einen Block länger.

Wichtig ist, dass alle ROM-Varianten dieselben internen BASIC-Speicheradressen nutzen, so dass BASIC-Programme, die direkt auf diese Adressen zugreifen, unter jeder ROM-Version funktionieren. Beachtet man den anderen Anfangsbereich der Speicheradressen beim RAM-BASIC, stimmen die Adressen mit denen des ROM-BASIC + Offset überein.

Bei einigen BASIC-Programmen findet man deshalb eine Zeile

```
10 IF DEEK(-16383)=-15605 THEN AH=1024:VS=0:ELSE AH=11264:VS=10240
```

zur Bestimmung der vorliegenden Version (RAM oder ROM). VS ist der Offset, der zu den Speicheradressen addiert wird, z.B. in Programmen mit Maschinencodeteil wie in CAVE, KNOSSOS oder MAZOGS

```
20 DOKE9,AH:DOKE863+VS,DEEK(863+VS)+8192:RUN
```

Die Erweiterung von Lutz Elßner

Lutz Elßner, einer der Entwickler im Z9001-Team, hat die 2K-Erweiterung um die **vollständige Nutzbarkeit der logischen E/A-Kanäle** des Z9001 angepasst.

Nutzerkatalog 1/88, Kennblatt 46:

Ein Nachteil des BASIC-Interpreters besteht in der eingeschränkten Nutzbarkeit von Ein- und Ausgabekanälen. Eingabe ist nur von Tastatur oder Kassette, Ausgabe nur auf Bildschirm (CONSOLE), Kassette oder Drucker (LIST) möglich. Bei der Aufzeichnung auf Tonbandkassette wird ein Verfahren verwendet, das von dem im Betriebssystem üblichen Verfahren abweicht (Typ/Name am Anfang der Daten, undefinierter oder kein File-Control-Block), was sogar zu Verwechslungen von BASIC- und sonstigen Dateien führt. Die veränderte E/A-Anpassung führt den Datenaustausch grundsätzlich über die im Betriebssystem vorgesehenen logischen Geräte CONST, READER, PUNCH und LIST durch. Diesen logischen Geräten können durch ASGN-Kommando beliebige vorhandene physische Geräte zugewiesen werden. Das gilt auch für das Kassetteninterface. Ein Programm, das die Blockung zu je 128 Byte für das Kassetteninterface realisiert, ist aus Speicherplatzgründen nur bei Verzicht auf das BASIC-Kommando RENUM im Interpreter-Speicherbereich unterzubringen. Es kann aber bei völliger Beibehaltung aller BASIC-Interpreter-Funktionen auch zusätzlich im Speicher bereitgestellt oder weggelassen werden, wenn nicht mit (Analog-) Kassetteninterface gearbeitet werden soll. Dann ist ein anderen externes Speichergerät zuzuweisen. (z. B. Digitalkassettengerät K5261).

Die Anweisungen PRINT#, LIST#, LOAD#, INPUT# sind mit den Gerätenummern 0 ,2 ,3 und CLOAD, CLOAD*, CSAVE, CSAVE* mit beliebigen zugewiesenen Peripheriegeräten funktionstüchtig, wobei die „BASIC-üblichen“ und die „Betriebssystem- üblichen“ Aufzeichnungsverfahren anwendbar sind. Eine durchgehende Behandlung von OPEN und CLOSE ist dabei gewährleistet. Bei der Anwendung von Treiberprogrammen für serielle Schnittstellen (V24, IFSS) oder parallele Schnittstellen (CENTRONICS) mit verschiedenen Protokollen ist Datenaustausch mit gleichen und anderen Rechnern möglich.

Mir sind bislang 2 verschiedene Revisionen dieser Erweiterung bekannt: Aus dem 192k_modul und

von einer Kassette. Beide verzichten auf den RENUMBER-Befehl, enthalten dafür Blockungsalgorithmen für die Kassettenarbeit.

CP/M-BASIC

Für CP/M gab es von Rossendorf bzw. Robotron angepasste BASIC-Varianten. (ZBASIC, ZBASICT oder auch BAS, BAST):

Das KC-BASIC für den Betrieb unter CP/M modifiziert (aufrufbar als ZBASIC oder BAS). Ausgangspunkt für die Modifikation war das sog. RAM-BASIC. Volle Kompatibilität ist gegeben, sofern diese zum RAM-BASIC gegeben war (d.h. Basic-Programme mit Maschinencode-Anteil laufen i.A. nicht!). Bei der Arbeit mit Disketten ist auf Großschreibung der Dateinamen zu achten, andernfalls kann es insbesondere für den ungeübten Nutzer zu Problemen kommen. Die Dateinamen können max. 8 Zeichen lang sein (Buchstaben,Zahlen). Als Typ wird standardmäßig „ZBS“ verwendet (kann vom Nutzer nicht beeinflusst werden). Z.B. erscheint das Programm PASCH im Diskettenverzeichnis als PASCH.ZBS.

Zum Laden von Kassette und Speicher auf Floppy gibt es die Version ZBASICT bzw. BAST.

Ich habe 2006 die aktuellste BASIC-Version (Das Plotter-BASIC-86) für CP/M als **Version BASG cpm-basic2.zip** umgesetzt, die die **Grafikausgabe** auf Plotter und Grafikzusatz unterstützt. Dieses BASIC bietet damit auch eine bessere Kompatibilität für Basic-Programme mit Maschinencode-Anteil. 2008 ist auch ein DIR-Befehl hinzugekommen.

Der Quellcode

Das BASIC war recht leistungsfähig und auch recht schnell. Was über die offiziellen Befehle hinaus alles machbar war, demonstriert ein kleines Programm, was ich begleitend zum BASIC-Tipps Insider-Vortag der 2. Z1013-Tagung geschrieben hatte.

Offensichtlich gab es für den BASIC_Interpreter eine ausführliche Dokumentation der internen Arbeitsweise. Für die Erweiterung des Z9001-BASICS (also die obere 2K) besitze ich ein paar (extrem schlecht lesbare) Seiten dieser Dokumentation als Thermokopien.

Für den 8K-Kern habe ich nur den Quelltext „HC-BASIC ROM-Version 7. 1. 1987“; die verschiedenen 2K-Erweiterungen sowie die RAMBASIC-Version und BASIC-84 habe ich reassembliert. Eine komplette interne Beschreibung des BASICs ist leider nirgendwo mehr auffindbar. Für die 2K-Erweiterung des originalen Z9001-BASIC habe ich noch einige (extrem schlecht lesbare) Blätter als thermokopierte Unterlagen, in denen Details der Implementierung erläutert werden. Auch wird auf weitere Dokumente verwiesen „...(vergl. ERWEI.HC Seite 26/27, BASIC.HC Seite 44/45)...“

Wer noch Dokumente zum BASIC hat, möge sich bitte mit
mir in Verbindung setzen

Ich habe die diversen BASIC-Varianten reassembliert und auch ein bisschen Zusatzmaterial zur Erweiterung aufbereitet. In diesem Paket basic.zip sind von mir reassemblierte Quellcodes der verschiedenen BASIC-Varianten des Z9001 enthalten.

Das BASIC besteht ja aus dem 8K-Kern und der Erweiterung. Zum Kompilieren der gewünschten Version müssen in basic_8k_kern.asm die beiden Symbole ROMTYP und BASTYP wie gewünscht definiert werden:

- Kassette R0111 „BASIC_84“ + „RAM“ ,mit M511
- BASIC84 „BASIC_84“ + „ROM“ ,mit M511
- BASIC85 „BASIC_85“ + „ROM“ ,mit M511
- BASIC86 (BM600.ROM) „BASIC_86“ + „ROM“ ,mit BM608
- CP/M-BASIC orig „BASIC_84“ + „ROM“ (mit Einschränkung, s. CPM-Basic.txt), mit cpm-erw
- Elßner „BASIC_85“oder„BASIC_86“ + „ROM“ ,mit ERW_LE

Dann ist „as m511“ zu starten. Hier wird die Erweiterung M511 mit den zugehörigen Symbolen passend kompiliert und die beiden Teile zum Basic zusammengefügt.

Neben M511 (das „alte“ BASIC mit RENUMBER statt RENUM, aber ohne Grafik-Befehle) gibt es BM608 (das „neue“ BASIC mit RENUM und Grafik) sowie Lutz Elßners BASIC- Erweiterung ERW_LE für vollständige Unterstützung aller logischen Ein/Ausgabekanäle (2 Varianten, Symbol ERWTYP)

basic.zip Paketinhalt:

basic_8k_kern.asm	Quelltext für alle Varianten des Kerns
bm608.asm	Quelltext Plotter-Erweiterung (BASIC86)
erw_le.asm	Quelltext Lutz Elßners Erweiterung (Patch von bm608)
M511.asm	Quelltext der originalen Erweiterung
as.bat	Hilfsprogramm zum Aufruf des Assemblers
basic_8k_kern.lst	Assembler-Listing
bm608.lst	Assembler-Listing
erw_le.lst	Assembler-Listing
M511.lst	Assembler-Listing
basic-decode.pl	Perl-Hilfsprogramm zum Dekodieren von komprimierten BASIC-Programmen (TAP, KCC)
CPM-Basic.txt	Unterschiede der bekannten CPM-KCBASIC-Varianten
erweiterung.txt	Fragment der internen Dokumentation zu M511
basic-2le.rom	BASIC-Erweitwerung von L.Elßner, Version 1.00
basea.rom	BASIC-Erweitwerung von L.Elßner, vermutlich ältere Version (kleine Unterschiede in den Merkzellen für die Kassettenroutinen)

Patch BASIC86

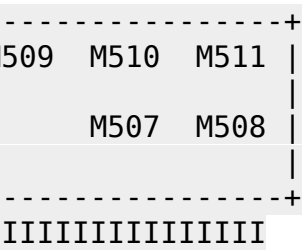
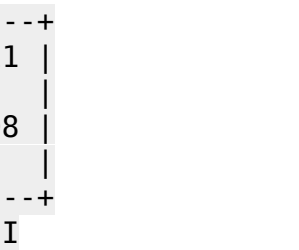
um das o.a. Problem beim BASIC86, einen sinnlosen Kopfblock zu schreiben, zu umgehen, gibt es einen kleinen Patch. Der kleine Nachteil dieser Lösung ist, das jetzt wieder direkt interne Monitoraufrufe getätigt werden.

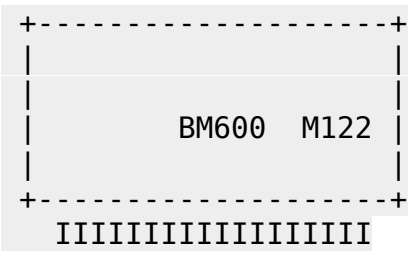
```
; Patch in BASIC-Erweiterung BM608 für write tape
; nicht OPENW nutzen, das (schreibt nur sinnlosen (!!!) Block 0
        org    0E68Bh
                call    0F593h
                ld      a, 1
                ld      (6Bh), a    ; BLNR
```

```
                nop
                nop
;
        org    0E6EEh
                push    de
                call    sub_E7D8
                pop     de


        org   0e7d8h
sub_E7D8:       ld      a, (6Bh)    ; BLNR
                dec     a
                jp      nz, 0F46Fh  ; wenn nicht Block 1
                ld      bc, 1770h   ; sonst: langer Vorton
                jp      0F472h      ; fuer ersten Block
```

Übersicht Basic-Module (Dank an T. Paul) (Bilder s.o.):

BASIC-MODUL BASIC85 robotron 1.40.690001.0 Leiterplatte (Leiterseite): 1.40.535823.1L/A Leiterplatte (Bestückungsseite): 1.40.535823.1B/A RFT 58564 5 EPROMs vom Typ K573RF2	BASIC-MODUL BASIC85 robotron 1.40.690001.0 Leiterplatte (Leiterseite): 1.40.535829.7L/A Leiterplatte (Bestückungsseite): 1.40.535829.7B/A 1.40.535821.5 RFT 59629 1 EPROM vom Typ K573RF2 (M511) 1 PROM U2364D (BM600)
	

PLOTTER-/GRAFIK-MODUL BASIC86 robotron 1.40.690033.2 Leiterplatte (Leiterseite): 1.40.535829.7L/A Leiterplatte (Bestückungsseite): 1.40.535829.7B/A 1.40.535821.5 RFT 59629 1 EPROM vom Typ 2716 (M122) 1 PROM U2364D (600) 	PLOTTER MODUL BASIC86 Verpackung: Technische Universität Karl-Marx-Stadt Sektion Automatisierungstechnik Wissenschaftlicher Gerätebau Plottergrafikmodul 690033.2 zum Kleincomputer Z9001, KC 85/1, KC87.10 + .11 Keine Aufschrift auf Modul Leiterplatte (Leiterseite): TK-142-35-L-0 KCBA Leiterplatte (Bestückungsseite): TK-142-35-B-0 KCBA FNR: - 2 PROM U2364D (600/608) 

Internas

```
#-------------------------------------------------------------------------------
---
# KC-BASIC Arbeitspeicher
#-------------------------------------------------------------------------------
---

Speicherbereich für Variablen
-----------------------------

je 6 Byte pro Variable

numerische Variablen:

        -------------------------
        | 1 | 2 | 3 | 4 | 5 | 6 |
        -------------------------
        |  Name |   Zahlenwert  |
        -------------------------

Name: Byte 1= 2. Buchstabe bzw. 00, Byte 2 = 1. Buchstabe

A = 00 41       A1 = 31 41      AB = 42 41
```

```
Stringvariablen:

        -------------------------
        | 1 | 2 | 3 | 4 | 5 | 6 |
        -------------------------
        |  Name | ^  | 00|Pointer|  <- Adr., wo der String abgelegt ist
        --------- |  ------------
                  |
                  Länge des Strings (max 255 Zeichen)

Name Byte 1: Bit 7 ist gesetzt
A$ = 80 41      A1$ = B1 41     AB$ = C2 41


Funktionsvariablen:

        -------------------------
        | 1 | 2 | 3 | 4 | 5 | 6 |
        -------------------------
        |  Name |
        -------------------------

Name Byte 2: Bit 7 ist gesetzt
wenn Stringfunktion, so ist außerdem Byte 1: Bit 7 gesetzt



Speicherbereich für Felder
--------------------------

alle zusammengehörenden Werte verwenden nur einen Namen. Unmittelbar danach
folgen die Dimensionen; danach dichte gepackt mit jeweils 4 Byte die
einzelnen
Werte.

Beispiel        DIM A1(1,2,3), A(3), C$(4)

        ------------------------------------------------------------------
---------
        | Name       | Länge des | Dimen- | Variablen je      | Anzahl
Folgebyte    |
        | des Feldes | Feldes    | sionen | Dimension         | je 4
Byte/Wert      |
        ------------------------------------------------------------------
---------
        | 31 41 (A1) | 67 00     | 03     | 04 00 03 00 02 00 | 2*3*4 *4 =
96 = 60H |
        | 00 41  (A) | 13 00     | 01     | 04 00             | 4     *4 =
16 = 10H |
        | 80 43 (C$) | 17 00     | 01     | 05 00             | 5     *4 =
20 = 16H |
```

```
        ------------------------------------------------------------------
---------

Länge des Feldes = 1 (Byte f. Dimensionen) + n (Anzahl Diemensionen) +
Anzahl Folgebyte
Bei der Zählung der Dimensionszahlen ist zu beachten, daß mit 0 begonnen
wird: 0..n = (n+1) Werte!

#------------------------------------------------------------------------
---
# Zahlen
#------------------------------------------------------------------------
---

20.2.2011: In Büchern war nichts über das intere Zahlenformat zu finden.
Dehalb habe ich das analysiert. Und so kompliziert ist es gar nicht:

BASIC speichert Zahlen generell als Gleitkommazahlen einfacher Genauigkeit
in 4 Byte ab (7 Bit Exponent mit Vorzeichen, 24 Bit Mantisse)
(floating point number, single precision, a signed seven bit exponent
and a signed 24 bit mantissa):

        -----------------
        | 1 | 2 | 3 | 4 |
        -----------------
        |   Zahlenwert  |
        -----------------

Schreibt man die Bytes in umgekehrter Reihenfolge (4 3 2 1), so erhält man

        ------------------------------------------------------------------
--
        | 4                             |   3       |    2       |     1
|
        ------------------------------------------------------------------
--
        | Exponent                      | Mantisse
|
        ------------------------------------------------------------------
--
        | 1 Bit expVorzeichen, 7 Bit exp | 1 Bit Vorzeichen, 23 Bit Mantisse
|
        ------------------------------------------------------------------
--
        | VZ, Bit7...Bit0               | VZ,(Bit2^-2)...Bit(2^-24)
|
        ------------------------------------------------------------------
--


Die Mantisse liegt immer im Bereich 2^(-1)<=mantisse<2^0
```

```
Das erste Bit(2^-1) ist daher immer 1, und wird deshalb als Vorzeichenbit
genutzt.

Mantisse Vorzeichen = "+", wenn bit==0, "-" wenn bit==1
stelligkeit=2^exp, wenn expVorzeichen = 1
stelligkeit=2^(exp-128), wenn expVorzeichen = 0 (Zweierkomplement)

Zahl = 0.1Bit(2^-2)...Bit(2^-24) * stelligkeit

Das Exponent-Vorzeichen ist beim KC genau andersherum als üblich!

Beispiel Umrechnung in internes Format:

0.1 dezimal
nächstgrößere 2er-Potenz ist 2^-3=0.125
0.1/0.125=0.8
0.8*2^24=13421772,8 ~ 13421773 = 110011001100110011001101b

Exponent
-3 = -000011b, 2er Komplement 1111101 (=bitweise negiert + 1)
Vorzeichen "-" -> 0
=> 0-1111101

Mantisse
von 110011001100110011001101b erste 1 weglassen, dafür Vorzeichen
=> 0-10011001100110011001101

Gesamtzahl
0-1111101|0-10011001100110011001101=7D4CCCCDh

Ablage im Speicher als CD,CC,4C,7D

Mit einem kleinen BASIC-Programm kann man das verifizieren:

10 A=0.1:!als erste Variable anlegen
100 VI=DEEK(983):!ADR. VON A
1000 PRINT "A=";A;" ";
1010 PRINT PEEK(VI+2);" ";PEEK(VI+3);" ";PEEK(VI+4);" ";PEEK(VI+5)




#-------------------------------------------------------------------------
---
# KC-BASIC CSAVE*
#-------------------------------------------------------------------------
---

Beispiel chessdok_ttt.tap (ein Text im MINTEX-Format; das Feld ist im
MINTEX-
Programm als DIMT$(30) definiert)
```

```
Block 01
D4 D4 D4 43 48 45 53 53  44 4F 4B 49 1A 84 85 7C  ÔÔÔCHESSDOKI.„…|
3F A9 AA 16 3A 9F A0 DD  3E AC AD 31 3E AA AB 87  ?©ª.:Ÿ Ý>¬1>ª«‡
3D 8C 8D FB 3C BE BF 3D  3C 84 85 B9 3B B7 B8 02  =Œû<¾¿=<„…¹;·¸.
3B 43 44 BF 3A 00 01 1D  3A 00 01 1D 3A 00 01 1D  ;CD¿:...:...:...
3A 00 01 1D 3A 00 01 1D  3A 00 01 1D 3A 00 01 1D  :...:...:...:...
3A 00 01 1D 3A 00 01 1D  3A 00 01 1D 3A 00 01 1D  :...:...:...:...
3A 00 01 1D 3A 00 01 1D  3A 00 01 1D 3A 00 01 1D  :...:...:...:...
3A 00 01 1D 3A 00 01 1D  3A 00 01 1D 3A 00 01 1D  :...:...:...:...

Block 02
3A 00 01 1D 3A 00 01 1D  3A EA 05 20 40 20 2A 20  :...:...:ê. @ *
4B 6F 6D 6D 61 6E 64 6F  73 20 6D 69 74 20 6D 69  Kommandos mit mi
6E 64 65 73 65 6E 73 20  7A 77 65 69 20 42 75 63  ndesens zwei Buc
68 73 74 61 62 65 6E 20  65 69 6E 67 65 62 65 6E  hstaben eingeben
20 2D 3E 20 45 4E 54 45  52 40 20 40 40 20 40 20   -> ENTER@ @@ @
2A 20 41 62 62 72 75 63  68 20 64 65 73 20 53 70  * Abbruch des Sp
69 65 6C 73 20 6A 65 64  65 72 7A 65 69 74 20 6D  iels jederzeit m
69 74 20 4E 45 20 6D 6F  65 67 6C 69 63 68 40 20  it NE moeglich@

Block 03
40 20 2A 20 77 65 69 73  73 65 20 6B 75 72 7A 65  @ * weisse kurze
20 52 6F 63 68 61 64 65  3A 20 45 31 47 31 40 20   Rochade: E1G1@
77 65 69 73 73 65 20 6C  61 6E 67 65 20 52 6F 63  weisse lange Roc
68 61 64 65 3A 20 45 31  43 31 40 20 40 20 2A 20  hade: E1C1@ @ *
53 74 61 74 75 73 20 75  6E 64 20 57 65 72 74 61  Status und Werta
65 6E 64 65 72 75 6E 67  65 6E 20 6E 75 72 20 64  enderungen nur d
75 72 63 68 66 75 65 68  72 65 6E 20 77 65 6E 6E  urchfuehren wenn
20 59 4F 55 52 20 61 6E  67 65 7A 65 69 67 74 20   YOUR angezeigt

..

Block 0D
74 61 74 69 6F 6E 20 66  75 65 72 20 5A 20 31 30  tation fuer Z 10
31 33 20 43 48 45 53 53  2D 4D 41 53 54 45 52 40  13 CHESS-MASTER@
20 2A 2A 2A 2A 2A 2A 2A  2A 2A 2A 2A 2A 2A 2A 2A   **************
2A 2A 2A 2A 2A 2A 2A 2A  2A 2A 2A 2A 2A 2A 2A 2A  ****************
2A 2A 2A 2A 2A 2A 2A 2A  2A 2A 2A 2A 2A 2A 2A 40  ***************@
20 40 20 4C 61 64 65 6E  3A 20 4C 20 31 30 30 20   @ Laden: L 100
33 34 36 30 40 20 53 74  61 72 74 3A 20 4A 20 33  3460@ Start: J 3
34 35 30 40 20 03 61 6D  6D 64 6F 6B 75 6D 65 6E  450@ .ammdokumen
               ^
               | Ende

Block FF ist nochmal Block 0D und ohne Bedeutung



Analyse Block 01
D4 D4 D4                        Kennung als Feld
43 48 45 53 53 44 4F 4B         Dateiname "CHESSDOK"
```

```
49 1A                   ??? vermutlich eine Länge ?

Es gibt keine direkte Möglichkeit zu ermitteln, ob es sich um
ein Zahlen- oder Stringfeld handelt. Auch die Größe ist nicht direkt
bestimmbar
(mit ein paar Programmtricks geht es aber -> total commander plugin listtap2
zeigt auch Felder an!)

84 85 7C 3F A$(0)       je 4 Byte pro Feldwert
A9 AA 16 3A A$(1)       das 1. Byte ist die Länge des Strings
9F A0 DD 3E A$(2)       das 2. Byte ohne Bedeutung (aber =Länge+1)
AC AD 31 3E A$(3)       das 3.+4. Byte ist die originale Adresse im
Stringspeicher
AA AB 87 3D A$(4)       die Strings werden von den oberen Adresse abwärts
8C 8D FB 3C A$(5)       abgelegt; deshalb ist die erste Adresse 3F7C
BE BF 3D 3C A$(6)       3F7C + 84 (Länge) = 4000H, das ist das obere RAM-
Ende
84 85 B9 3B A$(7)       (bei einem RAM-Modul)
B7 B8 02 3B A$(8)
43 44 BF 3A A$(9)
00 01 1D 3A A$(10)      die ab hier folgenden Strings haben die Länge 0,
sind also leer
00 01 1D 3A A$(11)
00 01 1D 3A A$(12)      die Adresse dürfte m.E. ohne Bedeutung sein; beim
00 01 1D 3A A$(13)      Einlesen wird der String ohnehin auf eine andere
Adresse geschoben
00 01 1D 3A A$(14)
00 01 1D 3A A$(15)
00 01 1D 3A A$(16)
00 01 1D 3A A$(16)
00 01 1D 3A A$(17)
00 01 1D 3A A$(18)
00 01 1D 3A A$(19)
00 01 1D 3A A$(20)
00 01 1D 3A A$(21)
00 01 1D 3A A$(22)
00 01 1D 3A A$(23)
00 01 1D 3A A$(24)
00 01 1D 3A A$(25)
00 01 1D 3A A$(26)
00 01 1D 3A A$(27)
00 01 1D ...
Block 02
...      3A A$(28)
00 01 1D 3A A$(29)
00 01 1D 3A A$(30)
EA 05                   Länge der nachfolgende Stringhalde
                        ( = Summe der Längen aller Strings
                        84+A9+9F+AC+AA+8C+BE+84+B7+43+00+...+00 = 5EA)
                                es folgt der Stringheap
20 40 20 2A 20 4B 6F 6D 6D 61 ....      die einzelnen Strings dicht
```

```
                                        gepackt hintereinander. Es gibt
keine
                                        Trennzeichen o.a. !

die Strings sind wie auf dem Heap in umgekehrter Reihenfolge abgelegt:
        ....
        A$(8)
        A$(7)
        A$(6)
        A$(5)
        A$(4)
        A$(3)
        A$(2)
        A$(1)
        A$(0)
```

12.5.03 Dank an Lutz Elßner für das „Ur-Basic“ 1984 (M497-M501)

From:
https://hc-ddr.hucki.net/wiki/ - **Homecomputer DDR**

Permanent link:
https://hc-ddr.hucki.net/wiki/doku.php/z9001/basic?rev=1320934747

Last update: **2011/11/10 14:19**