Erfahrungen mit KC-PASCAL

Altes Wissen neu entdeckt

Dietmar Uhlig, Garitz, 22.4.2023

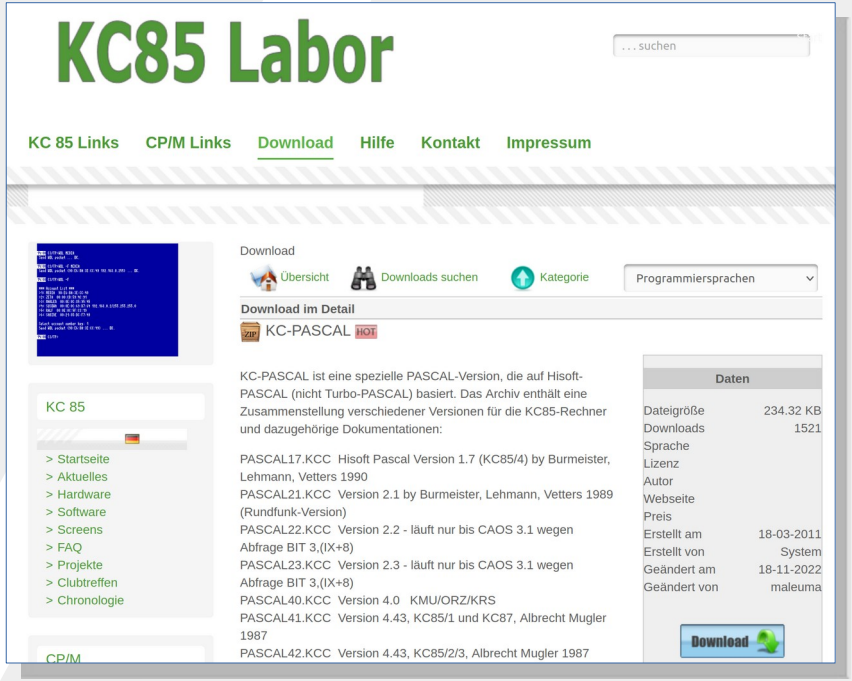
EINLEITUNG

Motivation

- programmieren auf dem KC für den KC (CAOS) in einer Hochsprache
- gute Geschwindigkeit des erzeugten Programms
- und natürlich: *Nostalgie*

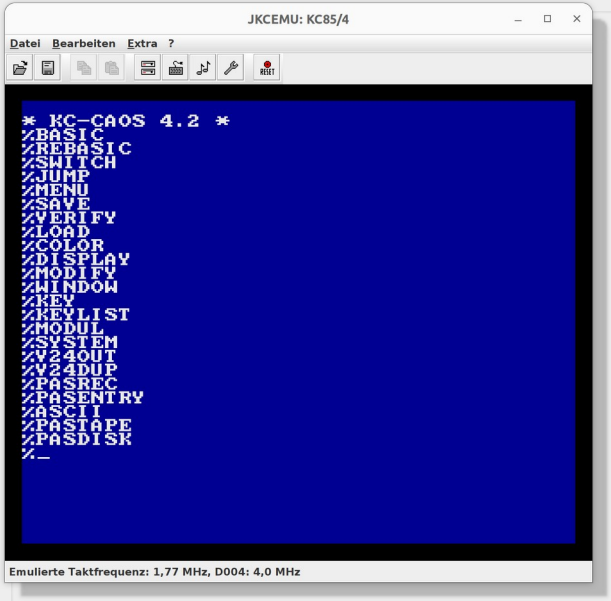
Bezugsquellen

- Programm
  - KC85-Labor
    http://kc85.info/index.php/download-topmenu/-viewdownload/5-programmiersprachen/43-kc-pascal.html
- Doku
  - KC85-Labor
  - CPC-Wiki, Hisoft Pascal 4T Manual
    https://www.cpcwiki.eu/imgs/6/64/Hisoft_Pascal_4T_Manual_%28English%29.pdf
    → Syntaxdiagramme, Infos zum IX-Register



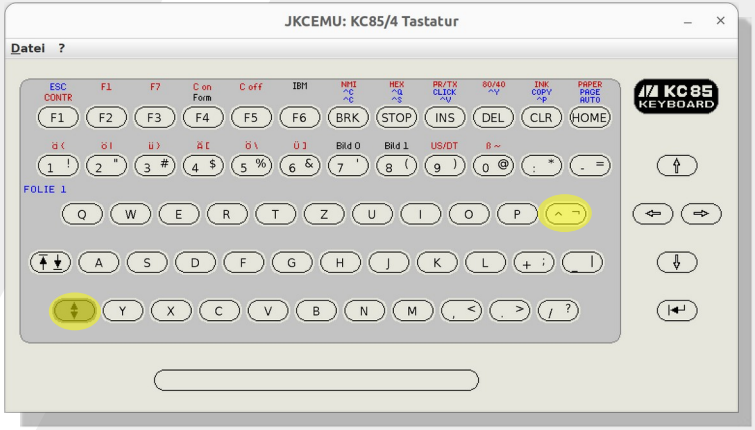
„Arbeitsumgebung“

- KC 85/4 (CAOS 4.2) mit D004
- KC-Pascal 5.1
  - integriertes PASEX
- JKC-Emu
  - praktisch für unterwegs



EDITOR

Editieren im JKCEmu

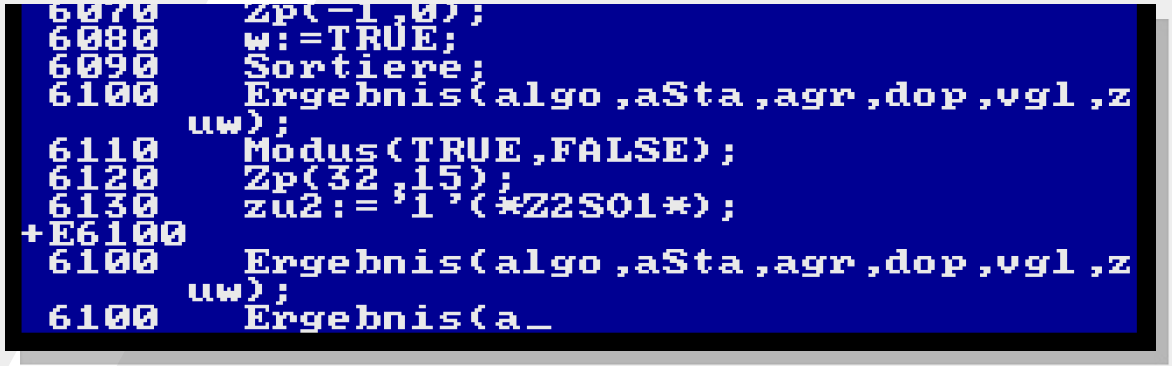


- Paste funktionert nicht
  - „Schreiben in den Tastaturpuffer“ → Zeichen werden verschluckt
  - andernfalls → eckige Klammern kommen nicht an
- „]“ im JKCEmu per emulierter Tastatur (Shift+Neg.) (Shift+Leer → „[“)
- Quelltext-Export am besten per Listing (Compile)
  - „Drucker“

```
+L
    10 (*$P+*)
    20 (*2023-02-12,
```

Integrierter Editor (1)

- Zeileneditor
  - Eingabepuffer 80 Zeichen
    → 5stellige Zeilennummer, Leerzeichen,
    74 Zeichen für Code
- Zeilennummern sind für den Compiler ohne Bedeutung



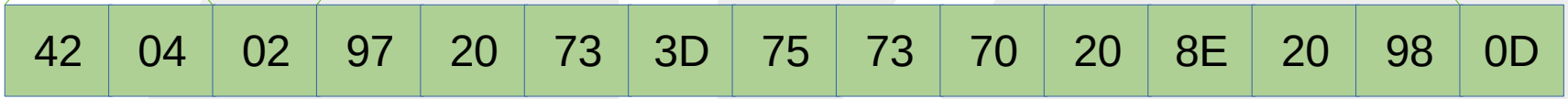
Quelltext im Speicher

Zeilennummer

Anzahl führender Leerzeichen

Quelltext mit Token für reservierte Wörter

Zeilenende



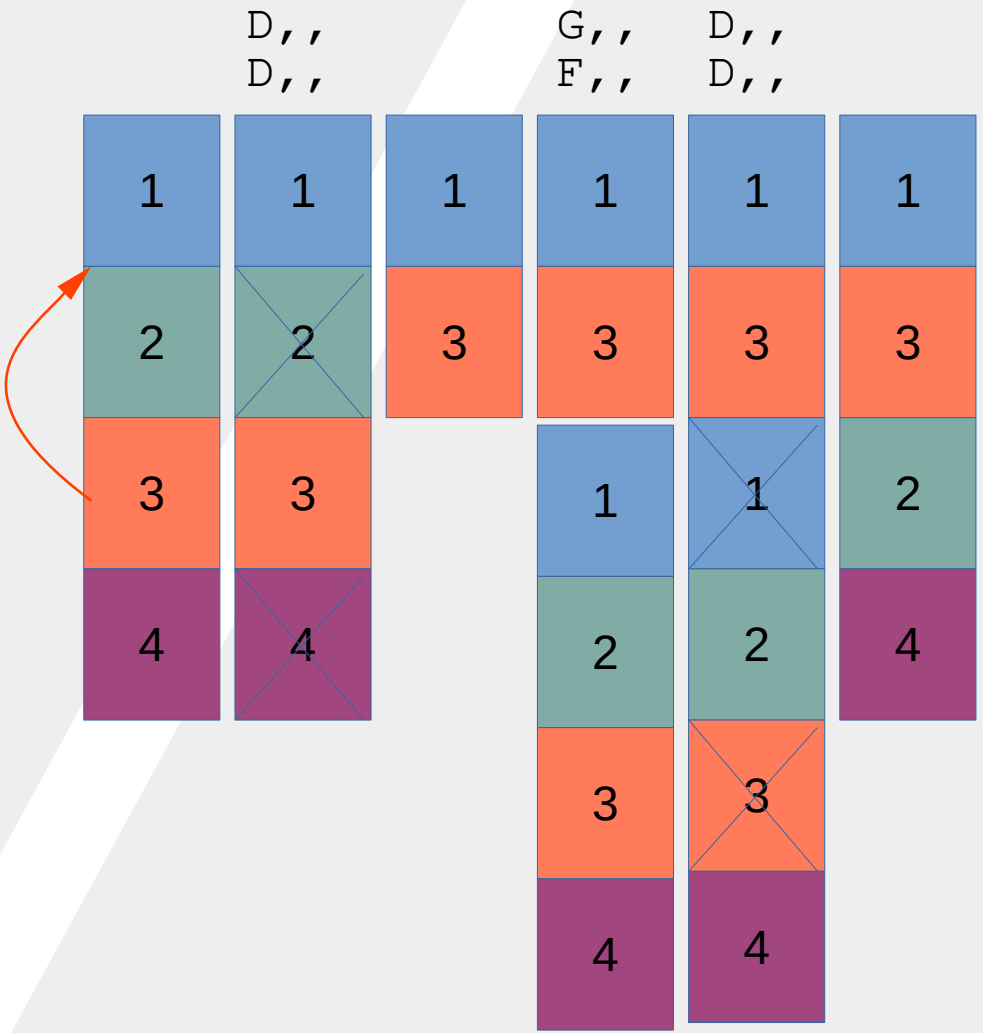

```
1080   BEGIN
1090     IF s=usp THEN BEGIN
1100       SETC(7,4);
```

Integrierter Editor (2)

- Block löschen:
  D *Zeilennummer_1*,*Zeilennummer_2*
- einzelne Zeile löschen:
  Zeilennummer <ENTER>
- leere Zeile einfügen:
  Zeilennummer <Leerzeichen> <ENTER>
- Neunummerierung der Zeilen nur für den gesamten Quelltext
- 1 Zeile verschieben → kopieren und löschen:
  M *alte_Zeilennr.*,*neue_Zeilennr.*
  alte_Zeilennr. <ENTER>

Block verschieben

- Prinzip:
  - Quelltext doppelt laden
  - unnötige Teile wegschneiden
- Hinweis:
  Zeilen wiederfinden mit Markern (Kommentare im Quelltext)



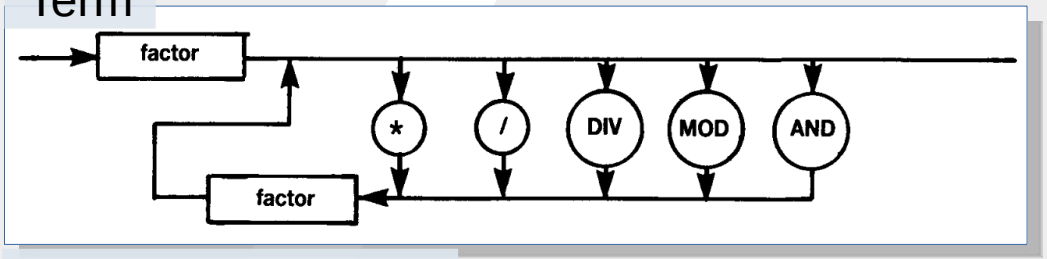
SPRACHE

Rangfolge der Operatoren

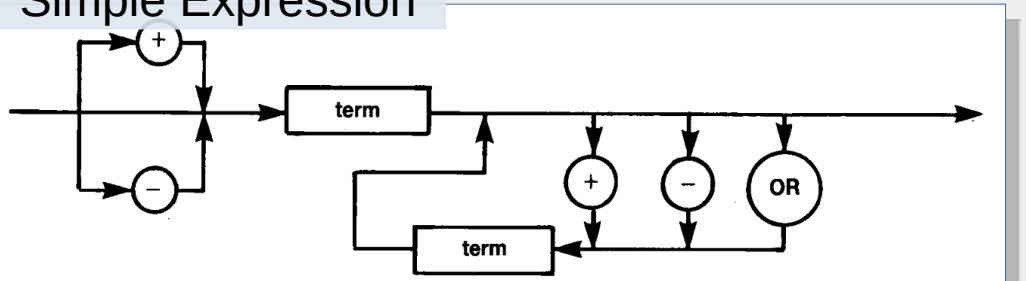
- Unterschied zu C, Perl, Python…
- logische und arithmetische auf gleicher Ebene
- logische vor Vergleichsoperatoren

```
IF (s<>0) AND (a[s]<>y) THEN
  PSetS(s,y);
```

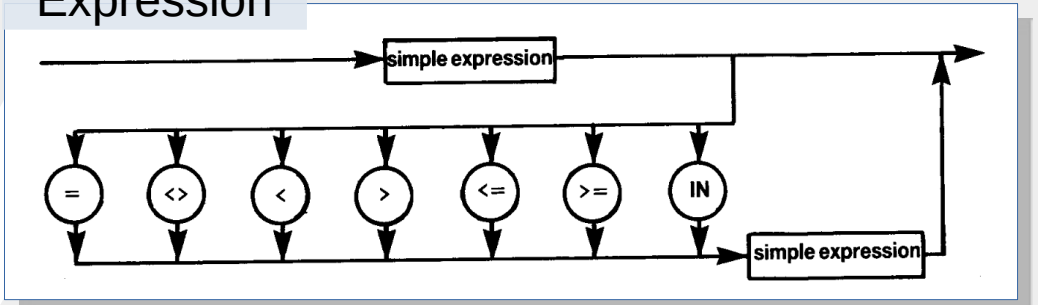
Term



Simple Expression



Expression



initialisierte Felder

- *...schmerzlich vermisst*
- z.B. Verwendung als Lookup-Tabellen
- Ersatz durch CASE-Anweisung

```
PROCEDURE WrAlgoT(al:tAlgo);
CONST L='<E:1.5>';
BEGIN
  CASE al OF
    SEL:    WRITE('Select. ');
    INS:    WRITE('Insert. ');
    BUB:    WRITE('Bubble  ');
    SHE:    WRITE('Shell   ');
    QUI:    WRITE('Quick   ');
    QUM:    WRITE('Qu/Med  ');
    QMI:    WRITE('Q/M+Ins ');
    HEA:    WRITE('Heap    ')
  END;
END;
```

abgekürzte Logikauswertung

- Beispiel:
  ```
  IF (i>0) AND (a[i]=t) THEN
  ```
  - führt bei `i` außerhalb der Feldindexe zur Fehlermeldung
- stattdessen verschachtelte IF-Anweisung

Schleifenabbruch

- muss mit einer zusätzlichen Variablen und IF nachgebildet werden

```
ok:=TRUE;
WHILE (j>s) AND ok AND w DO
  IF Ga(j-s,iTmp) THEN
  BEGIN
    Schiebe1(j,j-s); j:=j-s;
  END
  ELSE
    ok:=FALSE;
```

CAOS

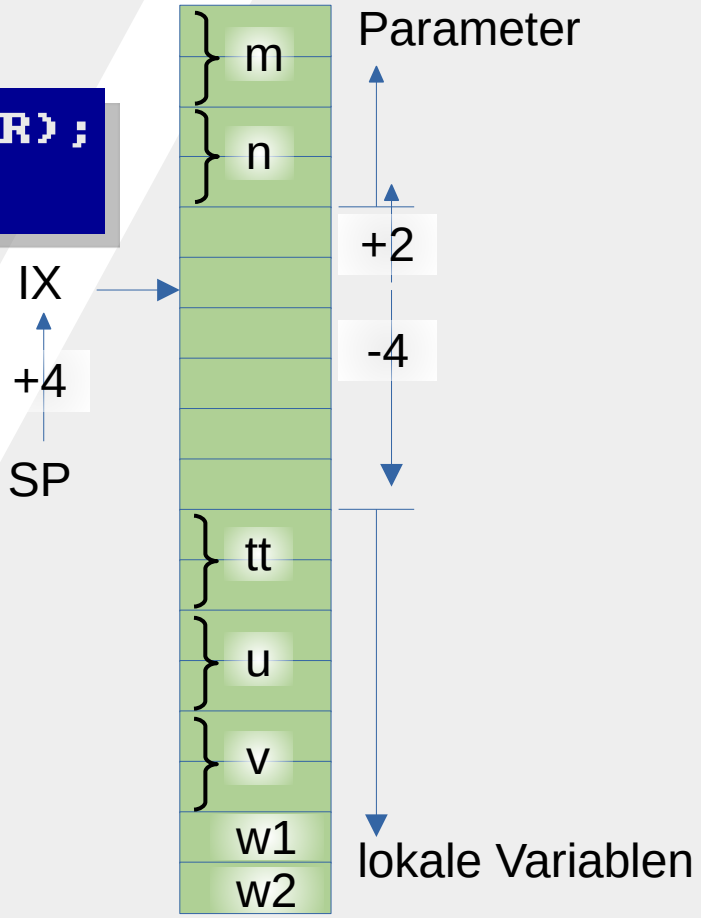
Implementierte Aufrufe

- WRITE, READ, PAGE, GOTOXY
- TOUT, TIN
  - je nach Einstellung beim Kompilieren (PASEX) für Tape oder Diskette
- SETC, GETC, PLOT, CLRPLOT, LINEPLOT, CIRCLE

IX als Framepointer

```
PROCEDURE Quick(m,n:INTEGER);
VAR l,r,t,tt,u,v:INTEGER;
    w1,w2:BOOLEAN;
```

- VAR-Parameter: 2 Byte für die Adresse
- FUNCTION: Return-Wert vor/über dem ersten Funktionsparameter



CAOS-Aufrufe

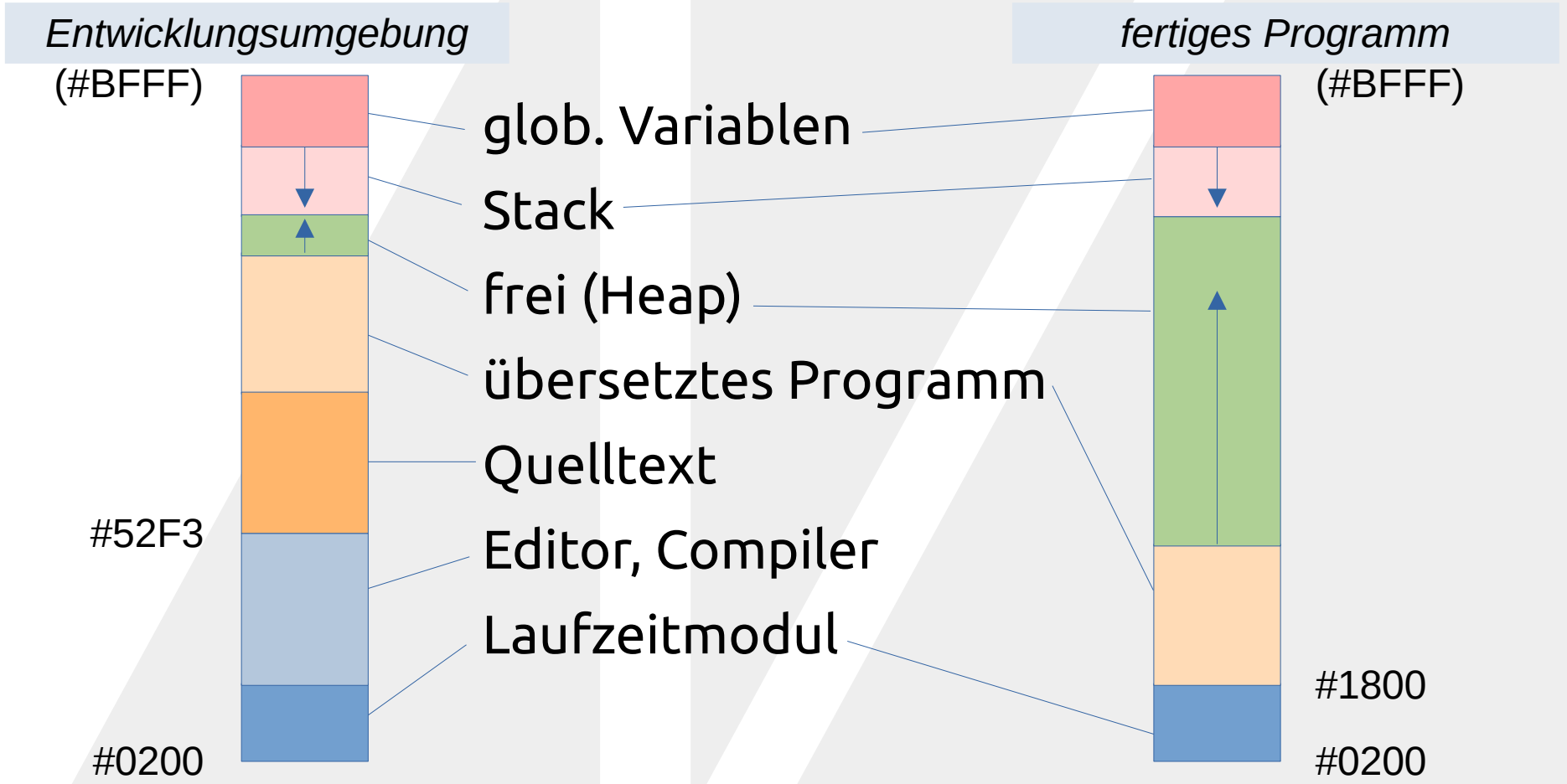
- Zugriff auf Arbeitszellen im IRM mit `GETSYS(`*`adr`*`)` und `SETSYS(`*`adr`*`,`*`wert`*`)`
- `USER()` nicht verwendbar, wenn HL für einen Parameter gebraucht wird
- IX-Register eigenständig auf CAOS-Wert setzen (steht bei KC-Pascal 5 in Adr. #06B0)
- PV6 für einzelne Aufrufe
- IRM an/aus und PV1 für feste Abfolgen von CAOS-Aufrufen

Beispiel: Window initialisieren

```
PROCEDURE WinIn(nr,za,sa,zg,sg:INT
EGER);
VAR err:BOOLEAN;
BEGIN
  err:=FALSE;
  SETSYS(ARGC,WININ);
  INLINE(#DD,#7E,#0A);
  INLINE(#DD,#66,#08);
  INLINE(#DD,#6E,#06);
  INLINE(#DD,#56,#04);
  INLINE(#DD,#5E,#02);
  INLINE(#DD,#E5);
  INLINE(#DD,#2A,#B0,#06);
  INLINE(#CD,#1E,#F0);
  INLINE(#DD,#E1);
  INLINE(#DD,#CB,#FB,#16);
  IF err THEN HALT;
END;
```

```
LD   A,(IX+#0A)
LD   H,(IX+#08)
LD   L,(IX+#06)
LD   D,(IX+#04)
LD   E,(IX+#02)
PUSH IX
LD   IX,#06B0
CALL PV6
POP  IX
RL   (IX-5)
```

Speicherbelegung



Codegröße (Beispiele)

- 500 Codezeilen, 10 KB Quelltext
  → 8 KB Kompilat
- 600 Codezeilen, 11,5 KB Quelltext
  → 9,5 KB Kompilat

➡ ...Programmgröße ohne Laufzeitmodul

Zugriff auf RAM 8

- PEEK und POKE verwenden
- Programm-Stack unter #8000 festlegen (Befehl Z)
- entweder IDE-Stack ebenfalls auf #7FFF:

```
Z 32767,32767
```

  → max. Programmgröße ca. 5 KB

- oder IDE-Stack unverändert (#BFFF):

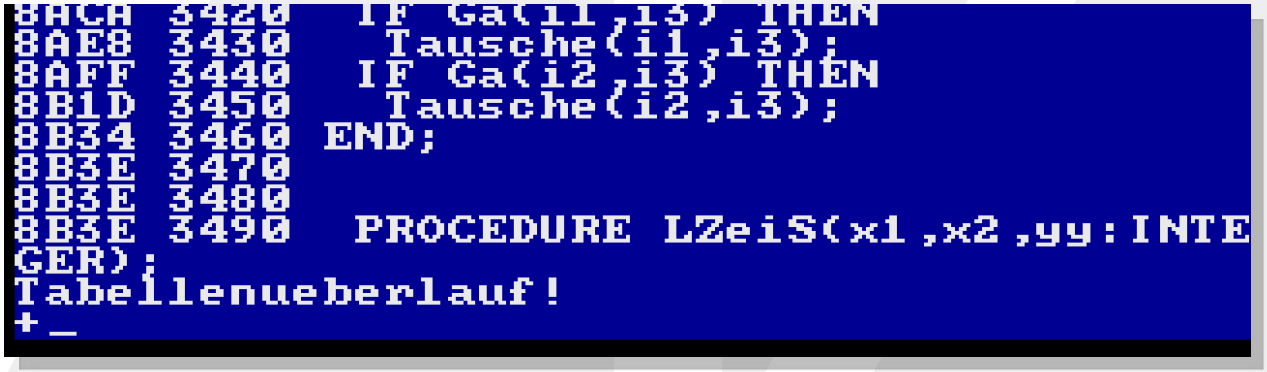
```
Z 49151,32767
```

  → Programm ist in der IDE nicht testbar!

SPEZIELLES

Größe der Symboltabelle

- ca. 2,75 KB (#1835 - #2355)
  - reicht im praktischen Bsp. für ca. 130 Bezeichner
- beim Überschreiten:



- 6 Konstanten
- 4 Typen
  - 15 Enum
  - 6 Record-Elem.
- 36 Variablen
- 25/33 Prozeduren, je nach lokalen Bezeichnern

Überlauf der Symboltabelle

- Bezeichner kürzen
- Aufzählungstypen durch `CHAR` oder `INTEGER` ersetzen
- nur einmal benutzte oder sehr kurze Prozeduren/Funktionen einbetten (auflösen)
- umsortieren zu Sub-Prozeduren in weit vorn liegenden Prozeduren (wenn möglich)

```
PROCEDURE Quick(m,n:INTEGER);
  VAR[…]
 PROCEDURE SIn;
  […]
 PROCEDURE Pu(v:INTEGER);
  […]
 FUNCTION Po: INTEGER;
  […]
 PROCEDURE So3(i1,i2,i3:INT[…]
  […]
 PROCEDURE LZeiS(x1,x2,yy:I[…]
  […]
BEGIN
 l:=1;r:=agr;SIn;w1:=TRUE;
 a[0]:=0;
 WHILE w AND w1 DO BEGIN
  […]
```

Große Programme zusammensetzen

- Programm in mehrere Teile zerlegen
  - wenig Übergänge zwischen den Teilen
  - Testbarkeit der Teile ggf. mit Dummies erhalten
  - Hauptprogramm im ersten Teil
- globale Definitionen (CONST, TYPE, VAR) in allen Teilen exakt gleich halten!
  - in eigene Datei auslagern
- compilieren, verschieben und zusammensetzen siehe github.com/duhlig

ABGESANG

Offene Themen

- Pascal 5.0 (o. PASEX) mit USB unter CAOS 4.8
- Hi-Color-Modus, Sound, Modulschaltung
- dynamischer Speicher vs. RAM 8
  - d.h. Programm-Stack im RAM 4, Heap im RAM 8
- Overlay-Programme
  - nachladen, Speicherbänke umschalten, globale Var.

Danke für Eure Aufmerksamkeit!