```
50000 REM   $$$.SUB-DATEI ERSTELLEN UND AUSFÜHREN
50005 DIM HSUB$(7)
50010 HSUB$(7)=":Dateien werden auf Programmdiskette gesichert"
50020 HSUB$(6)="PIP A:=B:KUNDEN.DAT"
50030 HSUB$(5)="PIP A:=B:WAREN.DAT"
50040 HSUB$(4)=":Größe der Dateien"
50050 HSUB$(3)="STAT *.DAT"
50060 HSUB$(2)=":Das Menueprogramm wird geladen"
50070 HSUB$(1)="MENUE"

50080 OPEN "R",1,"A:$$$.SUB",128
50090 FIELD #1,128 AS SUB$

50100 FOR I=1 TO 7
50110     HSUB$(I)=CHR$(LEN(HSUB$(I)))+HSUB$(I)+CHR$(0)
50120     LSET SUB$=HSUB$(I)
50130     PUT#1,I
50140 NEXT I
50150 CLOSE

50160 END
```

Bild 3. Mit diesem Basic-Programm wird eine $$$.SUB-Datei erstellt und anschließend ausgeführt

Dazu folgende Erläuterungen: In den Zeilen 50010...50070 werden die Befehle im Feld HSUB$(I) abgelegt. Die $$$.SUB-Datei wird in den Zeilen 50080 und 50090 gebildet bzw. geöffnet. Dann werden die Befehle, versehen mit Längeninformation und Schlußbyte, in den Zeilen 50100...50150 in dieser Datei abgelegt. Der END-Befehl in Zeile 50160 schließlich löst einen Warmstart und damit die Ausführung der Befehle in der $$$.SUB-Datei aus. Zu Bemerken ist hier noch, daß auf diese Art Submit-Folgekomandos ausgeführt werden können, ohne daß SUBMIT.COM vorhanden sein muß.

Literatur:

[1] *Bernd Pol:* Vom Umgang mit CP/M, CP/M für die Praxis 1, IWT-Verlag

[2] CP/M-Betriebssystem, Funktionsbeschreibung, Triumph-Adler, Nürnberg

[3] *J.Wagner:* CP/M-Anweisung Submit, Fernmeldepraxis 3/84

Ein Fehler in der LIBF.A-Bibliothek

Mit dem Betriebssystem CP/M-68k liefert Digital Research neben dem C-Compiler auch die Bibliotheks-Datei LIBF.A, in der die Gleitkomma-Routinen für den C-Compiler enthalten sind. Dieser Compiler scheint mit Gleitkommazahlen auf Kriegsfuß zu stehen; zum Beispiel liefert der Ausdruck
float i, j;
i = 2.0 * (4.0 + 1.0) + j;
ein unsinniges Ergebnis. Ursache dafür dürfte sein, daß der Compiler die unteren 16 Bit der 32-Bit-Gleitkommazahl, die bei der Umwandlung der Gleitkommazahlen innerhalb der Klammern auftreten, als Integer- statt als Gleitkommazahl interpretiert und damit natürlich kein richtiges Ergebnis mehr liefert. Man sollte also bei der Verwendung von konstanten Klammerausdrücken vorsichtig sein: Das richtige Ergebnis erhält man nämlich, wenn man für den konstanten Teil gleich den Wert 10.0 einsetzt.
Viel schwerer als dieser wiegt jedoch ein Fehler in der LIBF.A-Bibliothek: Vergleiche von Gleitkommazahlen in C liefern ein falsches Ergebnis, wenn die beiden zu vergleichenden Zahlen negativ sind. Ursache dafür ist das Programm fpcmp.o, das in LIBF.A enthalten ist. Dieses Programm springt die fehlerhafte Routine ffpcmp an, die wiederum im Programm ffpcmp.o enthalten ist.
Bei der Programmierung dieser Routine hat sich der Programmierer ein paar Zeilen gespart. Glücklicherweise wird mit dem Betriebssystem auch das Dienstprogramm AR68.68K geliefert, das es in diesem Fall erlaubt, die fehlerhafte durch eine neue Funktion zu ersetzen.
Auf der Diskette sollten sich dazu die Dateien AS68SYMB.DAT, AS68.68K, AR68.68K und LIBF.A befinden. Zunächst gibt man nun das Programm im *Bild* mit einem Editor ein und speichert es unter dem Namen FFPCMP.S ab. Dann übersetzt man die Datei mit dem Kommando
AS68 FFPCMP.S
und ersetzt die fehlerhafte Funktion in der Bibliothek mit Hilfe des Dienstprogrammes AR68:
AR68 R LIBF.A FFPCMP.O
Nach dieser Änderung kann die geänderte Bibliothek genauso wie vorher verwendet werden, mit dem Umterschied, daß nun Vergleiche negativer Gleitkommazahlen die richtigen Resultate liefern. Zu bemerken ist noch, daß der C-Compiler anscheinend ähnliche Routinen zur Berechnung konstanter Ausdrücke benutzt – Vergleiche von Gleitkomma-Konstanten liefern daher auch nach dieser Änderung fehlerhafte Resultate. Das gilt ebenso für die Bibliothek LIBE.A, die sich von LIBF.A dadurch unterscheidet, daß sie mit Gleitkommazahlen nach dem IEEE-Format arbeitet. Hier ist jedoch die Vergleichsroutine wesentlich komplexer programmiert, so daß eine Änderung einen größeren Aufwand erfordert und daher hier nicht durchgeführt wurde.

Matthias Köfferlein

```
                .text

                .globl ffpcmp    * Die beiden Labels als global
                .globl ffptst    * definieren

ffpcmp:         tst.b d6         * Beide Zahlen negativ:
                bpl ffpcpos      * Vergleich umdrehen
                tst.b d7
                bmi ffpcneg

ffpcpos:        cmp.b d6,d7      * Das ist der Vergleich, wie er
                bne ffpcrtn      * in der Orginalroutine durchge-
                cmp.l d6,d7      * führt wird
                rts

ffpcneg:        cmp.b d7,d6      * Hier ist der Vergleich einfach
                bne ffpcrtn      * umgedreht worden
                cmp.l d7,d6
ffpcrtn:        rts

ffptst:         tst.b d7         * Diese Routine ist auch noch in dem
                rts              * Orginalprogramm enthalten

                .end
```

Wenn beide Zahlen negativ sind, wird hier einfach der Vergleich herumgedreht – und damit der Fehler behoben