AC1-PILOT

Version 1.0

WeRo 09/2022

Inhaltsverzeichnis

Was ist PILOT?[1]

Wer hierbei an die Fliegerei denkt und ggf. einen Flugsimulator erwartet, der wird enttäuscht...☹

PILOT ist die Abkürzung für *Programmed Inquiry, Learning, Or Teaching* und ist eine einfache Programmiersprache, die 1962 vom amerikanischen Psychologieprofessor *John A. Starkweather* für den Einsatz in der Schule/Lehre entwickelt wurde.

Lehrer sollten damit interaktive Programme schreiben, die dann von den Schülern verwendet wurden. Hauptanwendung waren Entscheidungsbäume (meist per Ja/Nein oder begrenzten freitextlichen Eingabemöglichkeiten), die den Bediener (Schüler) zu bestimmten Ergebnissen und "Aha"-Effekten führten. Diese computergestützte Methode fand aber kaum Verbreitung und Anwendung.

Warum kann man das nicht in auch z.B. BASIC machen? Wenn ein BASIC-Programm für die Handhabung von Dialogen mit freier Antwort interaktiv gemacht werden soll, muss der Programmierer einen großen Aufwand treiben. Speziell die Eingabe und Anordnung von Vergleichen mit möglichen Wörtern oder Wortteilen (deren Erkennung wichtig sein könnte) ist umfangreich. Die spezielle Programmiersprache PILOT erleichtert diese Verarbeitung und hält das Programm zugleich gut lesbar.

PILOT eignet sich allerdings kaum/schlecht für Berechnungen aller Arten, wie sie von Allzwecksprachen beherrscht werden. Für viele Zwecke spielt das keine Rolle, aber einige Versionen von PILOT ermöglichen sowohl Programmierung als auch gleichzeitig verfügbare minimale Rechenmöglichkeiten.

Es gab mehrere "Dialekte" mit unterschiedlichem Funktionsumfang. *John A. Starkweather* veröffentlichte 1977 sein PILOT-8080, Version 1.1 mit Quelltext in der April/Mai-Ausgabe des Journals "*Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, Box E, Menlo Park CA 94025*"[2]

***

2022 - nach 60 Jahren seit der Erfindung von PILOT - entstand auf dieser Grundlage eine Version für den AC1. Heutzutage wird kaum jemand PILOT nutzen, also warum habe ich ein solch altes Programm "neu aufgelegt"?

- Das Projekt ist (im Gegensatz beispielsweise zum umfangreicheren NEVADA-PILOT des gleichen Autors von ca. 1982) quelloffen und bietet damit die Chance der Einblicknahme und eigener Ideen zur Modifikation.
- PILOT ist weitgehend unbekannt, es gibt bislang nur wenige Portierungen der Sprache für andere Rechner.
- Man wird durch das Befassen damit nicht dümmer und als Rentner habe ich (eigentlich nie) Zeit...☺

1 siehe z.B. auch hier: http://rpilot.sourceforge.net/pilot.html
2 Scan z.B. hier: https://archive.org/details/dr_dobbs_journal_vol_02_201803/page/n181/mode/2up

Voraussetzungen und Anpassungen der AC1-Version

Voraussetzungen:

- AC1 mit 64 KiB RAM
- SCCH-Monitor
- USB-Interface
- ggf. Druckerinterface

Änderungen/Anpassungen:

- Speicherbereiche
- deutsche Hilfe-/Fehlertexte
- überarbeiteter Direktmodus
- LOAD/SAVE: Laden/Sichern von Quelltexten per USB
- DP umbenannt in LIST
- neue Kommandos:
  - HELP eingebaute Kurzhilfe
  - GO Programm starten
  - TH: Textausgabe ohne abschließende Zeilenschaltung
  - TW: Textausgabe in Sperrschrift
  - S: Steuerzeichen ausgeben
  - X: Maschinenroutine aufrufen
- EDIT ruft eingebauten Texteditor auf

Schnellstart:

- PILOT-Interpreter nach 4000h laden
- Start mit "1" Normalstart, Demoprogramm wird geladen (oder Speicher gelöscht)
- Start mit "2" Warmstart, Programm bleibt erhalten
  (zuvor muss ein Programmtext geladen worden sein)
- Nach der Eingabeaufforderung >:
  - HELP listet verfügbare Anweisungen auf
  - LIST listet Programmtext auf
  - GO startet das Programm
  - BYE Rückkehr zum Monitor

Diese Dokumentation enthält Passagen aus dem auszugsweise frei übersetzten Artikel "GUIDE TO 8080 PILOT, VERSION 1.1" (enthalten im Scan o.g. Zeitschrift), ergänzt durch die Änderungen infolge der AC1-Spezifiken.

Programmieren in PILOT

Ein PILOT-Programm ist eine Textdatei. Sie besteht aus einer Reihe von Anweisungen, die Schritt für Schritt beschreiben, was der Computer tun muss, um mit einer Person zu interagieren. Ein Anweisungscode, der aus einem oder mehreren Buchstaben gefolgt von einem Doppelpunkt besteht, definiert den Anweisungstyp und bestimmt dessen Funktion. Jede Textzeile enthält nur eine Anweisung gemäß folgender Struktur:

```
[<MARKE>] <ANWEISUNG> [<BEDINGUNG>]  :  <AUSDRUCK>[,<AUSDRUCK>...]
```

<ANWEISUNG> und : sind Pflicht, die übrigen Angaben je nach Anweisungsart optional. Jede Zeile ist mit Wagenrücklauf (0Dh) abgeschlossen. Das Dateiende ist mit dem (im Text nicht vorkommenden) Hex-Wert 01h markiert.

Beispielprogramm mit Grundanweisungen

Für die Demonstration eines einfachen PILOT-Programms werden die drei wichtigsten Anweisungen benutzt:

```
T: KENNST DU PILOT?
A:
M: JA, SICHER, FREILICH, NA KLAR
TY: DANN KANNST DU DIESE EINLEITUNG ÜBERSPRINGEN.
TN: EINLEITUNG FÜR DICH:
```

Die drei "Codes" sind Anweisungen:

T:	für "**T**ype“	-> Textausgabe
A:	für "**A**ccept"	-> Tastatur-Eingabe erwartet (Antwort des Benutzers)
M:	für "**M**atch"	-> Vergleich der Eingabe mit Liste alternativer Begriffe

Die Buchstaben Y und N sind Bedingungen. Werden sie an einen Code angehängt, so wird diese Anweisung abhängig vom Erfolg des letzten Vergleichs ausgeführt oder nicht.

Daher wird der Text "`DANN KANNST DU DIESE EINLEITUNG ÜBERSPRINGEN`" nur dann ausgegeben, wenn die Antwort auf die Frage zu PILOT die Wörter "`JA`" oder "`SICHER`" oder ... enthält. Der Text "`EINLEITUNG FÜR DICH:`" erscheint nur dann, wenn der Vergleich nicht erfolgreich war:

`KENNST DU PILOT?`	<- Ausgabe
`SICHER WEISS ICH ALLES.`	<- Eingabe Bediener
`DANN KANNST DU DIESE EINLEITUNG ÜBERSPRINGEN.`	<- Ausgabe

Das hier entscheidende Wort in der Antwort war "SICHER", denn es ist in der M:-Liste enthalten. Hätte der Benutzer allerdings z.B. statt "JA" nur ein "J" oder statt "NA KLAR" nur "KLAR" eingegeben, so wäre der Vergleich negativ ausgefallen, denn beides ist in der Liste der Antwortmöglichkeiten nicht explizit enthalten. Andererseits hätte die Eingabe von "SICHERLICH" eine Übereinstimmung ergeben, denn darin ist das Schlüsselwort SICHER enthalten.

PILOT wurde entwickelt, um die Entwicklung von Konversationsprogrammen zu ermöglichen, die eine relativ freie Antwort des Benutzers ermöglichen. Das Erkennen von Benutzerantworten erfolgt, wie im obigen Beispiel angegeben, durch Durchsuchen der Antworten nach bestimmten Wörtern oder Wortstämmen. In normalen Gesprächen haben ein oder mehrere Wörter in einem Satz oft den größten Teil der Satzbedeutung. Die M: -Anweisung wird verwendet, um die Wörter, Wortteile oder Wortgruppen zu definieren, die in einer Antwort erkannt werden sollen. Programmautoren können so Antworten auf ihre gestellten Fragen zulassen und eine Liste von Wortelementen erstellen, die Schlüssel für eine angemessene Erkennung und Behandlung der Antworten sind.

Bedingte Ausführung

Die Buchstaben Y und N sind Bedingungen, unter denen die Operation eines beliebigen Anweisungstyps beim Anhängen an den Code ausgeführt wird. Die Operation hängt dann vom Vergleich der letzten A:-Antwort mit der M:-Liste ab. Y: ist als Kurzform von TY: erlaubt, desgleichen N: als Kurzform von TN:.

Eine PILOT-Anweisung kann auch vom Wert eines arithmetischen oder logischen Ausdrucks abhängig gemacht werden, der nach der Anweisung und vor dem Doppelpunkt in Klammern steht. Im vorliegenden System ist diese Methode auf einen numerischen Variablennamen beschränkt. Wenn der aktuelle Wert der referenzierten numerischen Variablen > 0 ist, wird die Anweisung ausgeführt. Andernfalls wird es ignoriert. Mit C:-Anweisungen kann der Wert der numerischen Variablen festgelegt werden. Diese diese Methode der Klammerbedingung hat Vorrang vor der Y-N-Bedingung.

Fortsetzung

Ein Doppelpunkt (:) am Anfang einer Zeile zeigt an, dass eine vorherige Textanweisung (T, Y oder N) weiterhin verwendet wird. So z.B.:

```
T: DIESER TEXT WIRD ANGEZEIGT
: UND DAS WIRD AUCH ANGEZEIGT.
```

Marken

Jede PILOT-Anweisung kann vor dem Operationscode eine Bezeichnung ("Marke") enthalten, mit der auf die Anweisung durch JUMP- (J :) oder USE- (U :) -Anweisungen verwiesen werden kann. Marken beginnen mit einem Sternchen (*) und enden mit einem Leerzeichen. Sie dürfen maximal zehn Zeichen enthalten. Eine Marke kann auch als einziges Element in einer Zeile verwendet werden, das mit dem Zeilenrücklauf endet. Beispiel:

```
*MARKE1
T: Anzeigen
J: *MARKE1
```

Die Textzeile "Anzeigen" wird nun endlos lange (Abbruch mit Strg+C) ausgegeben.

Bei der Bezugnahme auf die Marke mit J: oder U: kann das Sternchen weggelassen werden, erschwert aber dadurch etwas die Lesbarkeit des Quelltextes. In diesem System gibt es keine Fehlermeldung, wenn doppelte Marken vorhanden sind. Es wird jedoch nur die in der Reihenfolge erste Marke gefunden, wenn auf die Marke verwiesen wird.

String-Variablen

Der Name einer Stringvariablen beginnt mit dem $-Zeichen und darf maximal weitere zehn Zeichen enthalten. Groß- und Kleinschreibung wird unterschieden.

Nur durch eine ACCEPT-Anweisung kann einer Stringvariablen eine Zeichenkette zugewiesen werden. Operationen damit sind (außer Verwendung in T:- und M:-Anweisungen) nicht möglich. Später kann in einer TYPE:-Anweisung auf diese Stringvariable Bezug genommen werden. Auf die Variable folgt dabei entweder ein Leerzeichen, der Zeilenumbruch (0Dh) oder eines dieser Sonderzeichen:

,	;	:	.	?	!	"	(	)	'

Andere Folgezeichen könnten als Bestandteil des Variablennamens gewertet werden, wodurch dann die eigentlich gemeinte Variable nicht gefunden wird. In diesen Fällen wird dann der Name der Stringvariablen selbst angezeigt, genauso, wie wenn die Stringvariable zuvor nicht mit einer A:-Anweisung befüllt wurde.

Die maximale Zeichenanzahl in einer Stringvariablen ist durch die Länge der Antwort auf ACCEPT (A:) begrenzt, die am jeweiligen Punkt im Programm zulässig ist (zu setzen mit INMAX).

Die Anzahl der möglichen Stringvariablen wird durch deren jeweilige Länge und den RAM-Speicherplatz begrenzt, der zwischen Programmtext-Ende und RAM-Ende zur Verfügung steht.

Beispiel:

(Programm)	(Ausführung)
A: $NAME	"ROBERT" eingegeben
T: HALLO, $NAME	HALLO, ROBERT
T: DAS IST $UNBEKANNT	DAS IST $UNBEKANNT

Numerische Variablen

Numerische Variablennamen beginnen mit dem #-Zeichen und bestehen nur aus einem einzelnen Großbuchstaben (A...Z). Damit sind insgesamt 26 Variablen existent. Erscheint ein numerischer Variablenname in einer bedingten oder in einer C-Anweisung, wird das # nicht angegeben.

In dem vorliegenden System sind numerische Variablen auf ganzzahlige Werte von -99 bis +99 beschränkt. Sie werden zum Zeitpunkt der Initialisierung von PILOT auf Null gesetzt.

Einer numerischen Variablen kann wie folgt ein Wert zugewiesen werden:

1. händische Eingabe per A:-Anweisung — A:#X
2. rechnerisch per C:-Anweisung — C:X=1

Eine Bezugnahme auf die numerische Variable kann erfolgen:

1. per T:-Anweisung (Anzeige) — T:WERT=#X
2. mit der C:-Anweisung (Änderung/Berechnung) — C:X=X+1
3. als Bedingung — T(X):HALLO

Beispiel:

(Programm)	(Ausführung)
A: #X	"23" eingegeben
T: DER WERT IST #X	DER WERT IST 23
C: X=X+1	X wird um 1 erhöht

Operationen

Stringvariablen können nur eingegeben (A:), angezeigt (T:) oder verglichen werden (M:). Direktzuweisungen oder Verknüpfungen sind nicht zulässig.

Für *numerische Variablen* gibt es (neben Eingabe und Anzeige) folgende mögliche Operationen mit einem maximaler Wertebereich von +/-99 (vgl. C-Anweisung):

- Zuweisung C:X=55
- Addition C:X=X+1 C:X=Y+Z
- Subtraktion C:X=X-1 C:X=Y-Z
- Bedingung T(X):HALLO Ausschrift HALLO nur wenn X>0
  C(X):Y=Z+3 nur wenn X>0, dann Berechnung Y=Z+3

Numerische Vergleiche können nicht durchgeführt werden. In einigen Fällen (z.B. Test auf Gleichheit) ist jedoch der Ersatz durch Eingabe als Stringvariable verbunden mit der M:-Anweisung möglich.

Direktmodus

Nach dem Start von PILOT befindet man sich im Direktmodus. Das Zeichen > erscheint als Aufforderung zu einer Eingabe. Es kann nun ein PILOT-Direktkommando eingegeben werden, gefolgt von **ENTER ↵ .** Bei Eingabe eines nicht existenten bzw. fehlerhaft geschriebenen oder unzulässigen Kommandos wird ? angezeigt und eine neue Eingabe erwartet.

Beispiel:
>LIST↵ zeigt das aktuelle Programm im Speicher an

Im Direktmodus und im Programmablauf sind folgende Steuerzeichen relevant:

BACKSPACE	Letztes Zeichen löschen
CTRL+U	Verwerfen der bisherigen Eingabezeile und Neueingabe
CTRL+C	Abbruch des Programms

Die Kursortasten sind nicht wirksam!

Beachte:
Der Texteditor hat seine eigene Tastenbelegung mit vielen weiteren Steuerfunktionen.

PILOT-Kernanweisungen

Zu den im o.a. Beispiel angeführten Kernanweisungen kommen noch die Buchstaben J, U, E, C und R sowie Erweiterungen je nach Programmversion.

Im folgenden Abschnitt werden Syntax, Funktion und Beispiele sowie ggf. Fehlermeldungen für die Anweisungen in dieser AC1-PILOT-Version beschrieben.

T:	TYPE	Text/Variablen anzeigen
TH:		Text/Variablen anzeigen, aber ohne Zeilenschaltung am Ende
TW:		Text/Variablen in Sperrschrift anzeigen
A:	ACCEPT	Eingeben Zahl oder String
M:	MATCH	Testen auf Übereinstimmung
J:	JUMP	bewirkt, dass das PILOT-Programm an anderer (mit einer Marke gekennzeichneten) Stelle fortfährt.
U:	USE	Unterprogramm aufrufen bewirkt, dass eine Gruppe von Anweisungen (Unterprogramm) an diesem Punkt im Programm verwendet wird. Diese Gruppe beginnt mit einer Bezeichnung, die in der U-Anweisung benannt ist und endet mit einer E-Anweisung.
E:	END	Ende eines Unterprogramms oder Ende des gesamten Programms
C:	COMPUTE	Berechnung numerischer Konstanten oder Variablen möglich ist.
R:	REMARK	Kommentar
S:	SEND	Steuerzeichen an Bildschirm senden
X:	EXECUTE	Maschinenprogramm aufrufen
INMAX:		Maximale Eingabelänge (1...63) setzen
NEW$:		alle Stringvariablen löschen

Diese Anweisungen sind im Direktmodus nicht zulässig und führen zu einer Fehlerausgabe ‚?‘.

Alle Anweisungen sind (wie auch die numerischen Variablen) groß zu schreiben!

Schlüsselworte dürfen nicht auf "N" oder "Y" enden, denn das ist das Kennzeichen für eine Bedingungsangabe. Ausnahmen sind N: und Y: selbst.

Zeichenvorrat:

- Bei der Textausgabe sowie auch der Eingabe von Strings sind lediglich Zeichen mit den Codes 20h (Leerzeichen) bis 7Eh (‚ß‘ bzw. ‚~‘) möglich.
- Pseudografikzeichen (>=80h) sowie Blockgrafik (00...1Fh) gibt es nicht.

T:	**TYPE=Anzeige auf dem Bildschirm**
TH:	***+ hang***
TW:	***+ wide***
Y: **N:**	

```
SYNTAX:
[<Marke>] T  [<Bedingung>] : <Nachricht>
[<Marke>] TH [<Bedingung>] : <Nachricht>
[<Marke>] TW [<Bedingung>] : <Nachricht>
[<Marke>] Y [(<numerische Variable>)] : <Nachricht>
[<Marke>] N [(<numerische Variable>)] : <Nachricht>
```

Beschreibung:
Zeigt eine Nachricht an. Diese kann aus einem Literal, einer Stringvariablen oder einer numerischen Variablen bestehen. Alle Zeichenpositionen rechts vom Doppelpunkt werden wörtlich wiedergegeben, mit der Ausnahme, dass die Werte von Variablen als Ersatz für ihre Namen eingefügt werden. Stringvariablennamen werden durch den Beginn mit "$" und numerische Variablennamen durch den Beginn mit "#" identifiziert.

Die TH:-Anweisung entspricht der T:-Anweisung mit der Ausnahme, dass nach Ausgabe keine Zeilenschaltung erfolgt. Der Kursor bleibt also hinter dem Text. Das ist z.B. von Nutzen, wenn mit der nächsten Anweisung (nur :) die Ausgabe an der gleichen Stelle fortgesetzt werden soll oder unmittelbar hinter dem Text eine Eingabe mittels A:-Anweisung erwartet wird.

Die TW:-Anweisung gibt den Text (oder auch eine Variable) in Sperrschrift aus (also ein Leerzeichen nach jedem Zeichen), arbeitet ansonsten wie die T:-Anweisung.

H und W (*hang* und *wide*) sind Flags, die durch die entsprechende Anweisung gesetzt werden. Bei Benutzung der Anweisung : bleiben zuvor gesetzte Flags erhalten. Erst die T:-Anweisung setzt beide Flags zurück.

Als Bedingung kann entweder Y oder N (Als Ergebnis eines vorherigen M:-Vergleichs) oder eine numerische Variable in Klammern stehen. Dabei gilt X=0 -> N und X>0 -> Y.

Die Anweisungen Y: und N: sind abgekürzte Formen von TU: und TN:. Zu der bereits von der Y/N-Bedingung kann hierbei eine weitere Bedingung berücksichtigt werden: der (logische) Wert einer numerischen Variablen X (wie oben):

T(A):FUND	"FUND" wird nur dann angezeigt, wenn A>0 ist
Y(A):TEST	"TEST" wird nur dann angezeigt, wenn die Bedingung vorher "Y" war und A>0 ist

Fehlermeldungen:
Wenn Sie auf eine Variable verweisen, der bisher kein Wert zugewiesen wurde, wird der Name der Variablen angezeigt. Dies kann natürlich absichtlich erfolgen, weist jedoch häufig auf eine falsche Schreibweise des Namens der Stringvariablen hin.

Beispiel:

```
TH:WIE IST DEIN NAME?
A:$NAME
T:HALLO, $NAME.
```

A: ACCEPT=Eingabe eines Wertes

```
SYNTAX:
[<Marke>] A [<Bedingung>] :
[<Marke>] A [<Bedingung>] : $<Stringvariable>
[<Marke>] A [<Bedingung>] : #<numerische Variable>
```

Beschreibung:
Der PILOT-Benutzer kann eine Eingabe tätigen. Das kann erfolgen:

- "ins Leere" (temporärer Eingabepuffer, wird als String behandelt),
- in eine Stringvariable oder
- in eine numerische Variable.

Die eingegebene Zeichenfolge kann bearbeitet werden, ehe sie durch ENTER übernommen wird:

- Backspace letztes Zeichen löschen
- CTRL+U gesamte Eingabe neu tätigen
- CTRL+C bricht Eingabe und Programm ab.

Bedingung:

- Y bzw. N als Ergebnis einer vorherigen M:-Anweisung
- (X) logischer Wert einer nun. Variablen X (0 -> N)

Hinweise:

- Die Länge einer Eingabe ist auf 63 Zeichen oder auf einen durch INMAX:n festgelegten Maximalwert begrenzt. Erreicht die Eingabe den eingestellten Maximalwert (INMAX), so erfolgt automatisch eine Übernahme, ohne dass ENTER gedrückt werden muss. Das ist z.B. sinnvoll, wenn z.B. nur "J" oder "N" oder eine Ziffer 0...9 einzugeben ist.
- Betätigt man bei der Eingabe nur ENTER, so wird eine Stringvariable mit einem Leerstring und eine numerische Variable mit 0 belegt.

Besonderheiten bei numerischen Variablen:

- Die Eingabe muss eine ein- oder zweistellige Ganzzahl zwischen 0 und 99 sein. Werden Zahlen >99 eingegeben, so sind nur die beiden ersten Ziffern signifikant, der Rest wird abgeschnitten.
- Die Eingabe negativer Zahlen ist nicht möglich.
- Es wird geprüft, ob das eingegebene Zeichen auch eine Ziffer war. Die Eingabe darf daher nicht mit einem Leerzeichen beginnen.

Fehlermeldungen:

`*ZIFFER NÖTIG, NOCHMAL:`	ein numerischer Variablenname wurde benutzt, aber keine Zahl eingegeben
`*KEIN STRINGRAUM`	nicht genügend Speicherplatz für die Speicherung von Stringvariablen

Beispiel:

```
T:WIE IST DEIN NAME?
A:$NAME
T:HALLO, $NAME.
T:GIB EINE ZIFFER 0...9 EIN:
A:#X
T:DIE ZIFFER WAR #X.
```

M: und MC: MATCH= Test auf Übereinstimmung

```
SYNTAX:
[<Marke>] M  [<Bedingung>] : <Muster>[,<Muster>...,<Muster>]
[<Marke>] MC [<Bedingung>] : <Muster>[^<pMuster>...^<Muster>]
```

Beschreibung:
Bewirkt die Vergleich der Benutzereingabe (die von der letzten ACCEPT (A :) -Anweisung empfangen wurde) mit einer kommagetrennten Musterliste.

Ein übereinstimmende Element bewirkt, dass der YN-Schalter auf *TRUE* gesetzt wird, auch wenn alle übrigen Elemente nicht passen.

War die Antwort nur ENTER, so gab die A:-Anweisung nichts (d.h. einen Leerstring) zurück und es gibt keine Übereinstimmung mit den angegebenen Mustern, das Flag ist dann immer FALSE.

Match ist eine sehr mächtige Anweisung, bei der es aber auf exakte Schreibweisen (Leerzeichen!) ankommt.

Die Muster in der Liste werden durch Kommas getrennt, wobei führende oder nachfolgende Leerzeichen als Teil des Musters betrachtet werden. Ein Komma, das das letzte Element beendet, wird ignoriert, kann jedoch dazu dienen, das Vorhandensein eines vorherigen Leerzeichens anzuzeigen.

MC: ist eine Spezialform, die die Suche nach Text mit Kommas ermöglicht. Es ist mit M: identisch, außer dass das Caret (^ = 5Eh) anstelle des Kommas als Trennzeichen zwischen den Mustern eingesetzt wird.

Wirkungsweise:

1. Der letzten Antwort auf eine A:-Anweisung wird am Ende ein Leerzeichen hinzugefügt. Mehrere Leerzeichen sind auf eins reduziert.
2. Auch bei den Mustern sind mehrere Leerzeichen auf eins reduziert. Sie kennzeichnen auch, ob ein Muster mit einem Leerzeichen beginnt oder endet.
3. Der Antwortstring wird mit jedem Muster per beweglichem "Fenster" gescannt, bis entweder eine Übereinstimmung gefunden wird oder der Antwortstring erschöpft ist.

Beispiele:

Musterliste	Ergebnis TRUE (Y) bei Eingabe	Ergebnis FALSE (N) bei
`M:A,B,C`	der Buchstaben A oder B oder C und aller Worte, in denen A, B oder C vorkommen	jedem Wort, in dem weder A noch B noch C vorkommt
`M: A, B, C`	der Buchstaben A oder B oder C und aller Worte, die mit A, B oder C beginnen	z.B. HALT oder BAHN
`M:A ,B ,C ,`	der Buchstaben A oder B oder C und alle Worte, die mit A, B oder C enden	z.B. AUS oder OBI
`M: A , B , C ,`	nur die Buchstaben A oder B oder C	allen übrigen Worten

J: JUMP=Sprung zu Marke

```
SYNTAX:
[<Marke>] J [<Bedingung>] : [*] <Ziel>
```

Beschreibung:
Springt zum angegebenen Ziel im aktuellen Programm. Das Ziel muss auf eine vorhandene Marke verweisen. Das führende * als Markenkennung ist optional. Eine nicht gefundene Marke wird angezeigt, gefolgt von: `- MARKE NICHT GEFUNDEN`. Im Gegensatz zum originalen Programm erfolgt dann ein Programmabbruch, da eine Fortsetzung nicht sinnvoll erscheint.

Beispiel:

```
 J:*START
 ...
*START T:ARE YOU READY TO BEGIN?
 A:
 M:YES
 JY:QUEST
 ...
*QUEST T:I HAVE A QUESTION FOR YOU.
 ...
```

U: USE SUBROUTINE=Aufruf Unterprogramm

```
SYNTAX:
[<Marke>] U [<Bedingung>] : [ * ] <Ziel>
```

Beschreibung:
Springt zum angegebenen Ziel (einer Unterroutine) im aktuellen Programm und merkt sich den aktuellen Speicherort. Dorthin kehrt die Steuerung zurück, wenn eine END (E:) -Anweisung im Unterprogramm auftritt. Das Ziel muss auf eine vorhandene Marke verweisen, das führende Sternchen ist jedoch optional. Wie bei J: erfolgt Fehlermeldung und Programmabbruch, wenn die angegebene Marke nicht vorhanden ist.

Grenzen:
Unterprogramme können bis zu einer maximalen Tiefe von sieben verschachtelt sein.

Fehlermeldungen:

`-MARKE NICHT GEFUNDEN`	angegebene Marke nicht definiert
`*SUB-TIEFE ZU GROSS`	zu große Verschachtelung von USE-Anweisungen

Beispiel:

```
*START T:HIER STARTEN WIR.
T:BRAUCHST DU ANLEITUNG?
A:
M: JA
UY:*INSTR
...
*INSTR T:DAS IST, WAS DU WISSEN WOLLTEST.
E:
```

E: END=Ende der Subroutine bzw. des Programms

```
SYNTAX:
[<Marke>] E [<Bedingung>] :
```

Beschreibung:
Zeigt das Ende eines Unterprogramms oder das Ende des aktuellen Programms an:

- Wenn E: nach einer USE (U:) -Anweisung festgestellt wird, wird die Steuerung an die Anweisung übergeben, die auf die USE-Anweisung folgt.
- Wenn zuvor keine USE-Anweisung ausgeführt wird, so wird das Programm beendet und PILOT geht in den Direktmodus, ersichtlich am > Zeichen als Eingabeaufforderung.
- Bei einem PILOT-Programm ohne Unterprogramme ist die E:-Anweisung am Programmende entbehrlich. Nur, wenn auf Grund von Strukturen oder Bedingungen irgendwo mitten im Programm beendet werden soll, muss die E:-Anweisung stehen.

Grenzen:
Unterprogramme können bis zu einer maximalen Tiefe von sieben verschachtelt sein. Siehe USE (U:) für mögliche Fehlermeldungen.

Beispiel:

```
*START      U:*ERSTE
            T:ZEIT
            E:
*ERSTE      U:*ZWEITE
            T:ES
            E:
*ZWEITE     U:*DRITTE
            T:IST
            E:
*DRITTE     T:JETZT
            E:
```

```
Ergibt die Ausgabe: "JETZT IST ES ZEIT"
```

C: COMPUTE =Berechnen

```
SYNTAX:
[<label>] C [<Bedingung>] : <Num. Variable> = <Num. Ausdruck>
```

Beschreibung:
Berechnet einen Wert basierend auf der Auswertung des numerischen Ausdrucks und weist das Ergebnis der numerischen Variablen links vom Gleichheitszeichen zu.

Grenzen:

- Für den numerischen Ausdruck sind nur die folgenden Formen zulässig:
  - Ganzzahl
  - num. Variable + Ganzzahl, num. Variable - Ganzzahl
  - num. Variable + num. Variable, num. Variable - num. Variable
- Ganzzahlen und Inhalte numerischer Variablen sind auf Werte von -99 bis +99 beschränkt.
- Numerische Variablennamen sind die Großbuchstaben A bis Z. Es kann ein längerer Name geschrieben werden, es wird jedoch nur der erste Buchstabe verwendet.

Fehlermeldungen:
Der den Fehler verursachende Ausdruck wird angezeigt, gefolgt von einer der folgenden Nachricht:

`* ILLEG. AUSDRUCK`	kann aufgrund einer falschen Syntax oder als Reaktion auf einen unzulässigen numerischen Variablennamen auftreten.
`* WERT > 99`	
`* WERT <-99`	

Nach der Anzeige der Nachricht erfolgt eine Verzweigung zur Tastatureingabe, und wenn "RETURN" gedrückt wird, wird die Ausführung mit der nächsten PILOT-Zeile fortgesetzt.

Bei einem Wert> 99 wird der Wert auf 99 und bei einem Wert <-99 auf -99 gesetzt.

Bei den Berechnungen ist im Gegensatz zur Eingabe (A:...) und Ausgabe (T:...) das #-Zeichen vor der numerische Variable wegzulassen!

Beispiel:
C:X = A+5
C:X = A-B

R: REMARK=Kommentar

```
SYNTAX:
[<label>] R: <text>
```

Beschreibung:
PILOT ignoriert die Zeile mit der Anweisung REMARK. Es ist ein Mittel zum Speichern potenziell nützlicher Informationen in der Programmliste. Die REMARK-Anweisung kann an jedem Punkt in einer PILOT-Programmsequenz auftreten.

Beispiel:

```
      R:..............................
      R:UNTERPROGRAMM 1
      R:..............................
*UP1
```

S: SEND=Steuerzeichen ausgeben

```
SYNTAX:
[<Marke>] S [<Bedingung>] : <HEX-Steuerbyte>
```

Beschreibung:
Sendet das als 2-stelliges HEX-Byte (keine Variable möglich) angegebene Steuerzeichen an den Bildschirm. Der Suffix 'h' ist optional.

Beispiele aus dem Vorrat der AC1-Steuerzeichen:

`S:07`	`Klingelton ausgeben`
`S:0Ch`	`Bildschirm löschen`
`S:18h` `S:19h`	`Drucker einschalten` `Drucker ausschalten`

`S:0Eh`
`S:30h`
`S:30h`
`S:31h`
`S:30h`
	`Kursorpositionierung auf 00,10`
	
	`^N zeile spalte`

X: EXECUTE=Ausführen Maschinenprogramm

```
SYNTAX:
[<Marke>] X [<Bedingung>] : <HEX-Adresse>
```

Beschreibung:
Ruft die Maschinenroutine an der als 4-stelliger HEX-Wert angegebenen Direktadresse (keine Variable!) auf. Der Suffix 'h' ist optional. Endet die Maschinenroutine mit RET, so wird die Steuerung wieder an das aufrufende PILOT-Programm zurückgegeben.

Beispiel:

`X(A):0000h`	Sprung zu Adresse 0000h = Reset des AC1, wenn Variable A>0
`X:0272h`	Klingeltonroutine aufrufen (entspricht S:07)

Anmerkungen:
Eigene Maschinenprogramme sind zuvor auf geeignete Weise im Speicher abzulegen. Dafür steht der Speicherbereich 1900h...3FFFh zur Verfügung.

Eine Parameterübergabe ist nicht implementiert. Bei Bedarf kann das durch mehrere Aufrufadressen im Maschinenprogramm nachgebildet werden, z.B.:

```
m2000:   LD   A,1      ;X:2000 übergibt Parameter=1
         JR   m200A
m2004:   LD   A,2      ;X:2004 übergibt Parameter=2
         JR   m200A
m2008:   LD   A,3      ;X:2008 übergibt Parameter=3
m200A:   …             ;Inhalt von A ist Parameter für Maschinenprogramm
```

Die Auswahl der dann nötigen X-Anweisung (X:2000, X:2004 oder X:2008) kann entweder per bedingtem Sprung oder/und Bedingung erfolgen.

INMAX: INPUT MAXIMUM NUMBER=max. Eingabelänge

```
SYNTAX:
[<label>] INMAX [<Bedingung>] : <integer>
[<label>] INMAX [<Bedingung>] : <numerische Variable>
```

Beschreibung:
Begrenzt die Anzahl der Zeichen, die von der nachfolgenden ACCEPT-Anweisungen (A :) akzeptiert werden. Das Limit wird festgelegt, indem rechts vom Doppelpunkt entweder eine Ganzzahl oder der Name einer numerischen Variablen angegeben wird.

Die nächste Eingabe bei ACCEPT wird automatisch beendet, wenn der Grenzwert erreicht ist. Ist INMAX beispielsweise auf 1 gesetzt, so erfolgt die Eingabe sofort mit einem einzelnen Zeichen und ohne Drücken der Eingabetaste.

Die Änderung gilt in einem Programmlauf für alle folgenden A:-Anweisungen, sollte also nach einer vorübergehenden Einschränkung wieder aufgehoben werden.

Grenzen:
Mögliche INMAX-Werte sind 1...63. Standardwert beim Neustart von AC1-PILOT ist 63.

Beispiel:

```
T:GIB EINEN STRING MIT 5 ZEICHEN EIN:
 INMAX:5
A:
T:DANKE.
 INMAX:63
...
```

NEW$: NEW STRING VARIABLES=Löschen Stringvariablen

```
SYNTAX:
[<label>] NEW$ [<Bedingung>] :
```

Beschreibung:
Löscht die Speicherung aller Stringvariablen, die bis zu diesem Punkt im Programm definiert wurden. Dies ermöglicht die Wiederherstellung des von ihnen verbrauchten Speicherplatzes. Die Verwendung dieser Anweisung kann z.B. bei wenig Speicherplatz erforderlich sein, wenn die Programmsequenz die mehrfache Verwendung des selben Stringvariablennamens ermöglicht.

Beispiel:

```
            T:WIE LAUTET DEINE ANTWORT?
*ANSWER     A:$TEXT
            M:JA
            N:DAS WAR FALSCH.
            N:BITTE NOCHMAL VERSUCHEN!
            NEW$N:
            JN:*ANSWER
            T:RICHTIG, WEITER IM TEXT...
            ...
```

Direktkommandos

Während die Kernanweisungen nur im Programmmodus verfügbar sind, können die folgenden normalerweise im Direktmodus benutzten Anweisungen auch in einem Programm eingesetzt werden. In diesen Fällen ist wie bei den Kernanweisungen ein : hinter dem Kommando einzutragen. Nach der Ausführung des Kommandos wird jedoch die weitere Programmausführung abgebrochen.

Übersicht:

GO	aktuelles Programm ausführen	
LIST	aktuelles Programm auflisten	
PRINT	aktuelles Programm drucken per Monitor-V.24-Schnittstelle	
EDIT	aktuelles Programm bearbeiten (externer Editor!)	
NEW	aktuelles Programm im Speicher löschen	
BYE	PILOT beenden und Rückkehr zum Monitor	
LOAD	neues Programm laden	per USB-Interface, Name angeben, ohne Name erfolgt DIR-Anzeige
SAVE	aktuelles Programm sichern	

GO — Starten des aktuellen Programms

```
SYNTAX:
[<label>] GO [:]
```

Beschreibung:
Startet das aktuell im Speicher befindliche PILOT-Programm. Bei Benutzung in einem Programm wird dieses ab Anfang neu gestartet.

Fehlermeldungen:
Ist aktuell kein PILOT-Programm im Speicher, so erfolgt die Meldung: `*KEIN PROGRAMMTEXT!`

LIST — Listen des Programmtextes

```
SYNTAX:
[<label>] LIST [:]
```

Beschreibung:
Zeigt das PILOT-Programm auf dem Bildschirm an, das sich derzeit im Speicher befindet.

Abbruchmechanismus:
Aller 20 Zeilen stoppt die Anzeige mit der Meldung `TASTE...` . Mit CTRL+C kann sowohl an diesem Punkt als auch bei laufender Ausgabe die weitere Anzeige des Listings abgebrochen werden. Jede andere Taste setzt die Ausgabe des Listings fort.

Fehlermeldungen:
Ist aktuell kein PILOT-Programm im Speicher, so erfolgt die Meldung: `*KEIN PROGRAMMTEXT!`

PRINT — Drucken Programmtext

```
SYNTAX:
[<label>] PRINT [:]
```

Beschreibung:
Druckt das aktuell geladene Programm per V.24-Schnittstelle des Monitors. Voraussetzung dafür ist eine korrekte/angepasste SCCH-Monitorversion einschließlich Handware-Interface, siehe dazu hier.

Abbruchmechanismus:
Analog zu LIST: kann eine laufende Druckausgabe mit CTRL+C abgebrochen werden. Es erfolgt jedoch kein Stopp nach 20 Zeilen.

Fehlermeldungen:
Ist aktuell kein PILOT-Programm im Speicher enthalten, so ergeht beim Druckversuch die Meldung: `*KEIN PROGRAMMTEXT!`

EDIT — Bearbeiten Programmtext

```
SYNTAX:
EDIT
```

Beschreibung:
Ruft den eingebauten Texteditor zur Bearbeitung des aktuell geladenen Programmtextes auf.

Details zur Bedienung siehe unter TEXTEDITOR.

NEW — Programm löschen

```
SYNTAX:
NEW
```

Beschreibung:
Löscht das aktuell geladene Programm aus dem Speicher. Sinnvoll nur im Direktmodus.

BYE — PILOT verlassen

```
SYNTAX:
[<label>] BYE [:]
```

Beschreibung:
Bewirkt, dass PILOT beendet wird und zum Monitor zurückkehrt.

Diese Anweisung wird normalerweise im Direktmodus verwendet, kann aber auch in einem Programm sinnvoll als letzte Anweisung benutzt werden, um das Programm und gleichzeitig PILOT zu beenden.

LOAD — Laden neues PILOT-Programm

```
SYNTAX:
[<label>] LOAD [:]
```

Beschreibung:
Lädt ein PILOT-Programm von einem USB-Speichergerät in den Programmpuffer. Der evtl. zuvor existierende aktuelle Programmtext wird automatisch gelöscht. Ein NEW-Kommando ist nicht erforderlich.

Nach Bestätigung der Anweisung mit ENTER wird das USB-Gerät initialisiert (sofern noch nicht erfolgt) und es ergeht die Aufforderung zur Eingabe eines Dateinamens. Die PILOT-Dateien tragen die Endung *.PIL, die jedoch nicht mit anzugeben ist.

Wird statt der Eingabe eines Dateinamens nur ENTER gedrückt, so bekommt man das Inhaltsverzeichnis (nur Dateien *.PIL) angezeigt.

Einschränkungen:
Der Befehl steht nur zur Verfügung, wenn der AC1 ein USB-Interface besitzt und ein USB-Speichermedium angeschlossen ist. Andernfalls ergeht eine Fehlermeldung.

Fehlermeldungen:

`... NICHT GEFUNDEN.`	Der angegebene Name der Datei wurde nicht gefunden.
`USB-FEHLER!`	Es gibt ein Problem mit dem USB-Interface...

SAVE — Sichern Programmtext

```
SYNTAX:
[<label>] SAVE [:]
```

Beschreibung:
Speichert das aktuell im Speicher befindliche PILOT-Programm auf dem USB-Medium.

Nach Bestätigung der Anweisung mit ENTER wird das USB-Gerät initialisiert (sofern noch nicht erfolgt) und es ergeht die Aufforderung zur Eingabe eines Dateinamens. Die PILOT-Dateien tragen die Endung *.PIL, die jedoch nicht mit anzugeben ist.

Einschränkungen:
Der Befehl steht nur zur Verfügung, wenn der AC1 ein USB-Interface besitzt und ein USB-Speichermedium angeschlossen ist. Andernfalls ergeht eine Fehlermeldung.

Fehlermeldungen:

`Speichern...Ueberschreiben? (J)`	Der angegebene Name der Dateiname existiert bereits. Mit 'J' wird die Datei überschrieben, jede andere Taste bricht den Vorgang ab
`USB-FEHLER!`	Es gibt ein Problem mit dem USB-Interface...

Die Befehle LOAD und SAVE werden normalerweise nur im Direktmodus verwendet. Bei Benutzung in einem Programm wird dieses beendet und ein neues geladen bzw. das aktuelle Programm gesichert. Anschließend befindet man sich im Direktmodus.

Fehlermeldungen

AC1-PILOT erkennt eine Reihe von Programmfehlern. In diesen Fällen erfolgt eine Ausschrift der den Fehler verursachenden Programmzeile, gefolgt von '?'.

Meldung	Bedeutung
`- MARKE NICHT GEFUNDEN`	die von einem Sprung oder Unterprogrammaufruf anvisierte Marke existiert nicht
`*KEIN STRINGRAUM`	Stringraum ist erschöpft
`*ILLEG.AUSDRUCK`	die Anweisung ist syntaktisch falsch geschrieben
`*WERT >  99`	Das Ergebnis einer C:-Anweisung überschreitet +/-99
`*WERT < -99`	
`*SUB-TIEFE ZU GROSS`	zu viele Unterprogramm-Verschachtelungen
`*KEIN PROGRAMMTEXT!`	eine LIST:-, PRINT:- oder GO:-Anweisung wurde versucht, ohne dass ein Programmtext vorhanden ist
`*ZIFFER NOETIG, NOCHMAL:`	falsche Eingabe (String statt Zahl)

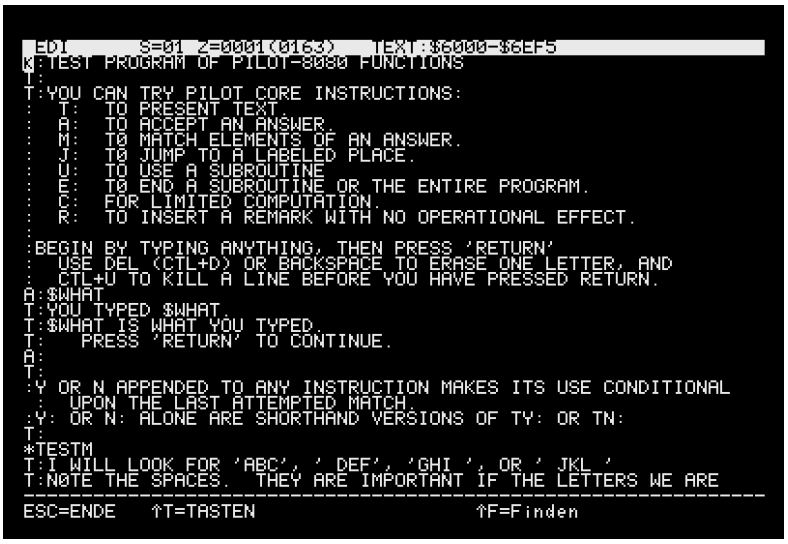
Im Fehlerfall wird das gestartete Programm abgebrochen und AC1-PILOT geht in den Direktmodus, da eine Fortsetzung in den meisten Fällen zu weiteren Fehlern führen würde. Eine Ausnahme bildet die Meldung "`*ZIFFER NOETIG, NOCHMAL:`". Bei einer erwarteten numerischen Eingabe wurde keine Ziffer eingegeben. Die Eingaben kann nun wiederholt (oder mit ^C abgebrochen) werden.

Liegt keine der o.a. Fehlermöglichkeiten vor und ist die Anweisung trotzdem nicht ausführbar, so wird das Programm abgebrochen und die fehlerhafte Programmzeile invers angezeigt, gefolgt von '?'

Tritt bei einer Eingabe im Direktmodus ein Fehler auf (Kommando falsch geschrieben bzw. nicht existent oder unzulässig), so wird dies lediglich mit '?' quittiert und eine neue Direkteingabe erwartet.

Der Texteditor

Die Originalversion von PILOT-8080, Version 1.1 enthielt keinen Text-Editor. Der Quelltext musste mit einem externen Programm bearbeitet werden. In AC1-PILOT steht nunmehr ein eingebauter kleiner Texteditor mit den wesentlichsten Funktionen einer Textverarbeitung zur Verfügung.



Kopfzeile:
Anzeige von Spalte/Zeile
und belegtem Speicherplatz

aktuelles Programm in Bearbeitung

Fußzeile: Bedienhinweise sowie
Eingabe eines zu findenden Worts

^T - > Hilfeseite mit Tastenbelegung

Rückkehr zur Textbearbeitung mit beliebiger Taste

Auf dem Bildschirm wird ein 28-Zeilen-Ausschnitt der Gesamtdatei dargestellt. In diesem Bereich kann man sich relativ schnell bewegen. In folgenden Situationen wird der aktuelle Bildschirminhalt wieder in den Gesamttext eingeordnet und ggf. das Textende korrigiert:

- Rollen nach oben oder unten, POS1 und Ende, Zeile löschen/Leerzeile einfügen
- Beenden des Editors

Bei langen Programmtexten entsteht in diesen Fällen eine geringe Verzögerung, ehe der Kursor wieder erscheint.

Es können nur die Zeichen von Leerzeichen bis ‚ß' (bzw. ‚~') eingegeben werden. Pseudografik (80h...FFh) sowie Blockgrafik (00...1Fh) gibt es nicht. Sollen Steuerzeichen (z.B. ^Q für invers ein) in auszugebenden Texten enthalten sein, so kann an geeigneter Stelle die SEND-Anweisung benutzt werden.

Die maximale Länge einer Zeile beträgt 63 Zeichen (das 64. Zeichen ist nicht nutzbar, u.a. ist es wegen eines Monitor-Fehlers bei "Entf" nicht löschbar!).

Es gibt kein "Rückgängig" und auch nur den Modus "Überschreiben".
Verfügbare Editorfunktionen:

Taste	^	Code	Funktion	
POS1	^A	01h	(A)nfang der Datei aufsuchen	
Entf	^D	04h	Löschen des akt. Zeichens, Rest rückt vor	
Einfg	^E	05h	(E)infügen eines Leerzeichens an akt. Position, Rest rückt weiter	
	^F	06h	(F)inden Zeichenkette ab Textanfang	auch Teilstrings, Kursor auf 1.gefundenes Zeichen, wird nichts (mehr) gefunden, erklingt "BEL" und der Kursor verbleibt an der letzten Position Groß-/Kleinschreibung nicht relevant
	^N	0Eh	(N)ächstes Vorkommen	
←	^H	08h	Kursor 1 Pos. links	nur innerhalb einer Zeile
→	^I	09h	Kursor 1 Pos. rechts	
↑	^J	0Ah	Kursor 1 Zeile hoch	Bewegung im gesamten Text am oberen/unteren Rand wird gescrollt
↓	^K	0Bh	Kursor 1 Zeile runter	
	^L	0Ch	(L)eerzeile **vor** aktueller Zeile einfügen (ab akt. Zeile rutscht alles runter)	
	^S	13h	(S)treichen aktuelle Zeile	
ENTER	^M	0Dh	innerhalb des Textes: Kursor an den Anfang der nächsten Zeile am Dateiende: eine neue (leere) Zeile anhängen	
TAB	^O	0Fh	tabuliert horizontal um jeweils 4 Leerzeichen	
	^T	14h	(T)astaturbelegung anzeigen (akt. BS-Inhalt wird gesichert)	
	^X	18h	geht an das akt. Zeilenende (in Leerzeile keine Reaktion)	
	^Y	19h	geht an den akt. Zeilenanfang	
Ende	^Z	1Ah	geht ans Dateiende	
ESC		1Bh	beendet den Editor und kehrt in den Direktmodus zurück	

Ein PILOT-Programm kann auch mit einem externen Editor erstellt und dann auch geeignete Weise geladen werden.

- Es ist ein Editor zu benutzen, der nur CR als Zeilenumbruch schreiben kann. Das beherrscht z.B. NOTEPAD++
- Am Dateiende ist nach dem letzten Zeilen-0Dh ein Markierungsbyte 01h anzubringen (Hex-Editor).
- Soll der Programmtext am AC1 per USB geladen werden, so ist die Dateierweiterung *.PIL zu benutzen.
- Der damit erstellte Programmtext ist dann nach dem Laden des PILOT-Interpreters auf geeignete Weise an Adresse 6000h zu laden.
- Erst dann ist AC1-PILOT mit "2" zu starten (Start mit "1" lädt entweder die Demo oder löscht ein vorhandenes Programm).

Anhang

Dateiformat

- Textdatei
- Zeilenendemarkierung: 0Dh (kein 0D/0A!)
- Dateiendemarkierung : 01h
- für USB:
  - Dateierweiterung *.PIL (ist beim Laden/Speichern nicht mit anzugeben!)
  - Dateiname max. 8+3 Zeichen lang, keine Sonderzeichen benutzen

Sonderzeichen im Programmtext

Zeichen	**Bedeutung**	**Beispiel**
#	num. Variable	#A
$	Stringvariable	$X
*	Marke	*%
, ^	Trennzeichen zwischen Mustern	M: A,B,C bzw. M: A^B^C

Speicherbelegung

- 0000...17FF Monitor und BWS
- 1800...18FF Systemarbeitszellen
- 1900...3FFF frei
- 4000...5FFF PILOT: Interpreter+Arbeitszellen+Stack+Demoprogramm
- 6000...xxxx PILOT-Programmtext
- rückwärts ab EFFF: PILOT-Stringbereich

Patchzellen:

Adresse	Standardwert	Bedeutung
4005h	31h	"1" = Kaltstartziffer aus Monitor
400Bh	32h	"2" = Warmstartziffer aus Monitor
4010h	01h	0/1 1=lädt Demo
4011h	04h	Tabulatorweite
4012h	3Fh	max. Länge einer Eingabezeile (63)

Drucken

Steht die nötige Hardware zur Verfügung (PIO2 und Pegelwandler), kann mit der monitorinternen V24-Ausgabe auf einem Drucker mit serieller Schnittstelle gedruckt werden, ohne dass ein extra Treiber erforderlich wird. Dabei wird die aktuelle Monitoreinstellung benutzt.

Achtung:
Bei nicht angeschlossenem oder eingeschaltetem Drucker hängt der AC1 bei Druckausgaben fest!

Hinweise:

1. Der Monitor muss dafür angepasst sein, d.h. die entsprechende Timing-Korrektur für 4800 und 9600 Bd. vorliegen. Ansonsten erfolgt kein korrekter Druck!

Monitor 10/88	Mon. 8 und Abkömmlinge	Mon. 10 und Mon. 11
orig. modif. 00F9: E5 6E 00FA: E1 DD 00FB: 7F 23 0E39: E5 6E 0E3A: E1 DD 0E3B: 7F 23	orig. modif. 0F9: E5 BE 0FA: E1 DD 0FB: 7F 23 E39: E5 BE E3A: E1 DD E3B: 7F 23	keine Modifikation erforderlich! Achtung! Der AC1 darf keine WAIT-Zyklen benutzen, diese verfälschen das Timing! Von Mon.11 gibt es aber eine angepasste Version, die auch mit WAIT-Einschüben korrekt arbeitet.

2. Das V24-Steuerbyte im Monitor ($1820) muss unbedingt auf "X-Draht" eingestellt sein (z.B. 04h für 9600 Bd), ansonsten wird die Rückmeldung "Drucker beschäftigt" (Signal CTS am AC1) ignoriert und der Text verstümmelt gedruckt (Pufferüberlauf).

3. Am Drucker ist die Einstellung "Seitenformatierung entsprechend Formularlänge" (automatische Trennlinien) abzuschalten und "Autolinefeed" zu aktivieren.

Originales Testprogramm

```
T:TEST PROGRAM OF PILOT-8080 FUNCTIONS
T:
T:YOU CAN TRY PILOT CORE INSTRUCTIONS:
:  T:  TO PRESENT TEXT.
:  A:  TO ACCEPT AN ANSWER.
:  M:  T0 MATCH ELEMENTS OF AN ANSWER.
:  J:  T0 JUMP TO A LABELED PLACE.
:  U:  TO USE A SUBROUTINE
:  E:  T0 END A SUBROUTINE OR THE ENTIRE PROGRAM.
:  C:  FOR LIMITED COMPUTATION.
:  R:  TO INSERT A REMARK WITH NO OPERATIONAL EFFECT.
:
:BEGIN BY TYPING ANYTHING, THEN PRESS 'RETURN'
:  USE DEL (CTL+D) OR BACKSPACE TO ERASE ONE LETTER, AND
:  CTL+U TO KILL A LINE BEFORE YOU HAVE PRESSED RETURN.
A:$WHAT
T:YOU TYPED $WHAT.
T:$WHAT IS WHAT YOU TYPED.
T:   PRESS 'RETURN' TO CONTINUE.
A:
T:
:Y OR N APPENDED TO ANY INSTRUCTION MAKES ITS USE CONDITIONAL
 :  UPON THE LAST ATTEMPTED MATCH.
:Y: OR N: ALONE ARE SHORTHAND VERSIONS OF TY: OR TN:
T:
*TESTM
T:I WILL LOOK FOR 'ABC', ' DEF', 'GHI ', OR ' JKL '
T:N0TE THE SPACES.  THEY ARE IMPORTANT IF THE LETTERS WE ARE
T:TRYING TO MATCH ARE NOT AT EITHER END OF A LINE.
A:
M:ABC, DEF,GHI , JKL ,
TY:MATCH
TN:NO MATCH
   U:*ASK
   JY:*TESTM
   J:*TESTC
*ASK
T:AGAIN? (Y OR N)
A:
M:Y
E:
R:................................
*TESTC
T:TYPE A NUMBER FROM 0 TO 99
T:I USE THE FIRST TWO DIGITS ENTERED.
  A:#A
T:ANOTHER NUMBER PLEASE
  A:#B
T:YOUR NUMBERS ARE #A AND #B.
T:I WILL SUBTRACT THEM.
  C:C=A-B
T:#A-#B=#C
T:
T:    TYPE 'NEXT' TO GO ON
A:
M:NEXT
  JN:*TESTC
T:
*TESTPLUS
    C:A=0
    C:B=0
T:TESTS CAN BE CONDITIONAL ON A NUMERIC VALUE > 0
T:A NUMERIC VARIABLE IS APPENDED AS IN 'J(X):*LABEL'
T:TYPE 'ABC' OR 'DEF'
A:
```

```
M: ABC
   CY:A=1
M: DEF
   CY:B=1
T(A):ABC FOUND
T(B):DEF FOUND
   C:C=A+B
   J(C):*END
T:NO MATCH OF EITHER ABC OR DEF
*END
T:END OF TEST
   U:*ASK
   JY:*TESTPLUS
T:
:NINE OTHER INSTRUCTIONS ARE AVAILABLE IN THIS SYSTEM:
:   MC:     TO LOOK FOR TEXT WITH COMMAS IN IT.
:   INMAX:  TO LIMIT THE NUMBER OF CHARACTERS ACCEPTED.
:   NEW$    TO ERASE STORED $TEXT.
:   LIST:   TO DISPLAY THE PROGRAM AT THE CONSOLE.
:   LOAD:   TO LOAD A NEW PROGRAM ($TEXT REMAINS).
:   SAVE    TO SEND THE PROGRAM TO A STORAGE DEVICE.
:   PRINT:  TO SEND THE PROGRAM TO A  LISTING DEVICE.
:   EDIT:   TO TO LEAVE PILOT AND USE AN EDITOR
:   BYE:    TO LEAVE PILOT AND GO TO A MONITOR.
    T:
    T:YOU CAN LOOK FOR TEXT WITHOUT COMMAS IN IT.
    T:A CARET '^' (5Eh) IS USED AS A SEPARATOR FOR MC:
  *TESTMC
    T:I WILL LOOK FOR 'ABC,D' OR 'A,BC'
A:$ENTRY
MC:ABC,D^A,BC
Y:YES, $ENTRY IS OK.
N:NO, YOU TYPED $ENTRY  INSTEAD OF THE ABOVE.
   U:*ASK
   JY:*TESTMC
T:
*TESTIN
T:INMAX: I WILL BE USED TO LOOK FOR 'A'.
T:PRESS ONLY A SINGLE KEY. DO NOT PRESS 'RETURN'
T:      ----                   ---
   INMAX:1
   A:
M:A
Y:YOU TYPED 'A'
N:THAT WASN'T 'A'
T:AGAIN? (Y OR N)
  A:
  INMAX:72
  M:Y
  JY:*TESTIN
T:
*ERRORS

T:I WILL SHOW SOME ERROR MESSAGES.  AFTER EACH, PRESS 'RETURN'
T:I WILL SHOW SOME ERROR MESSAGES.  AFTER EACH, PRESS 'RETURN'
T:I WILL TRY TO JUMP TO AN UNKNOWN LABEL '*HERE'
T:PRESS 'RETURN'
A:
R:--------------------
R:   J:*HERE
R:--------------------
T:
T:A C: INSTRUCTION HAS BAD SYNTAX.
T:    PRESS 'RETURN'
A:
      C:X NOT Y
T:THE EXPRESSION FIELD IS DISPLAYED + ERROR MESSAGE.
T:    PRESS 'RETURN'
```

```
A:
T:
T:THE VALUE OF X WILL BECOME GREATER THAN 99.
T:PRESS 'RETURN'
A:
    C:X=99
    C:X=X+10
T:THE STATEMENT CAUSING OVERFLOW IS DISPLAYED.
T:THE VALUE OF X WAS LEFT AT #X
T:A SIMILAR MESSAGE APPEARS WHEN X BECOMES SMALLER THAN -99.
T:PRESS 'RETURN'
A:
T:TYPE 'R' TO REPEAT ERROR MESSAGES.
   INMAX:1
A:
   INMAX:72
M:R
   JY:*ERRORS
T:
T:AN ILLEGAL STATEMENT IS SIMPLY DISPLAYED.
   MW:*
T:MW: IS NOT LEGAL IN THIS SYSTEM.
T:
T:YOU CAN OBTAIN IMMEDIATE OPERATION OF SOME INSTRUCTIONS BY
T:TYPING AFTER >. FOR EXAMPLE, '>LIST'
T:  '>LIST'     WILL DISPLAY THE CURRENT PROGRAM IN MEMORY.
T:  '>J:*LABEL' WILL JUMP IMMEDIATELY TO *LABEL.
T:CTL+C BY ITSELF WILL RETURN PILOT TO ITS STARTUP POINT.
T:
T:TESTS ARE NOW COMPLETE.
```

nicht original, an AC1-Version angepasst!