2 Allgemeines zu KRL-Programmen

2.1 Aufbau und Erstellen von Programmen

Sobald Sie in die Expertenebene gewechselt sind, verändert sich auch die Bedienoberfläche (s. Abb. 4).

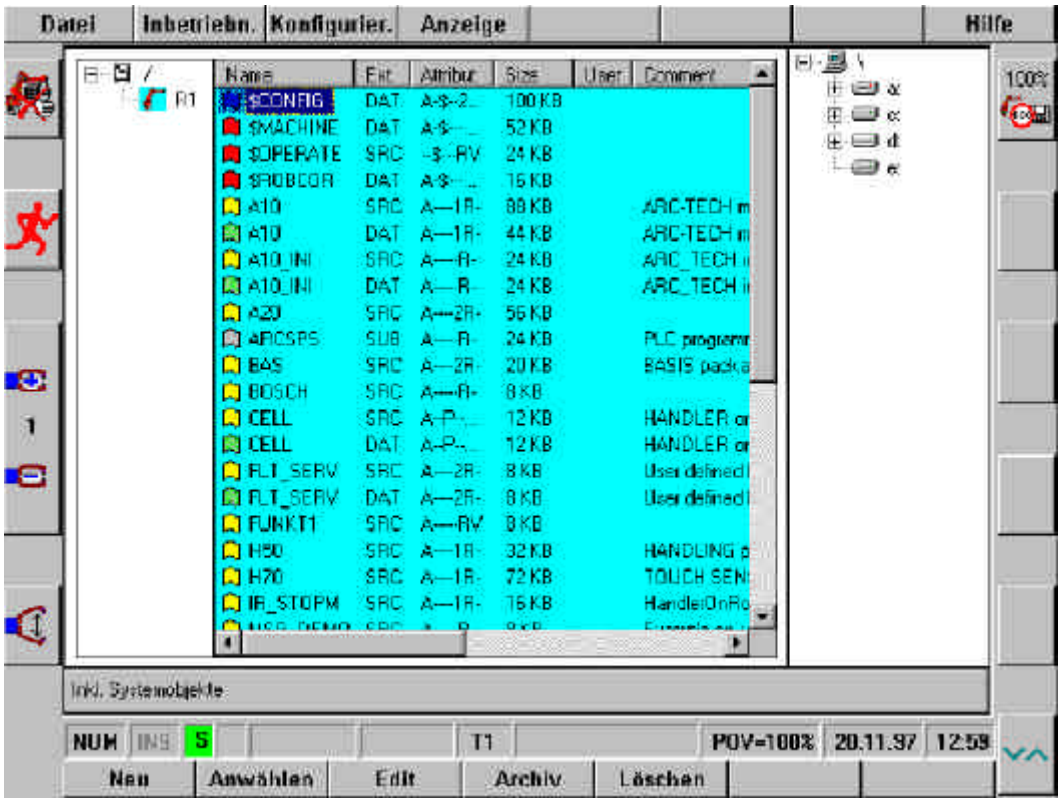


Abb. 4 Bedienoberfläche in Expertenebene

Während für den Anwender alle Systemdateien unsichtbar sind, kann der Experte diese jetzt im Programmfenster sehen und auch editieren. Ferner werden neben den Dateinamen und Kommentaren auf Expertenebene auch die Dateierweiterungen, -attribute und -größen angezeigt.

Standardmäßig sind nach der Installation der KR C1 – Software die in Tab. 1 aufgeführten Dateien vorhanden.

Datei	Bedeutung
$CONFIG.DAT	Systemdatenliste mit allgemeinen Konfigurationsdaten
$MASCHINE.DAT	Systemdatenliste mit Systemvariablen zur Anpassung von Steuerung und Roboter
$OPERATE.SRC	Systemdatei mit Programm– und Roboterzustandsdaten
$ROBCOR.DAT	Systemdatenliste mit Daten für das Dynamikmodell des Roboters
A10.DAT A10.SRC	Technologiepaket zum Bahnschweißen mit analogen Leitspannungen
A10_INI.DAT A10_INI.SRC	Technologiepaket zur Initialisierung des Bahnschweißens mit analogen Leitspannungen
A20.DAT A20.SRC	Technologiepaket zum Bahnschweißen mit digitalen Programmnummern
A50.DAT A50.SRC	Technolgiepaket für die Verwendung des Libo (Lichtbogen) – Sensors
ARCSPS.SUB	Submitfile für Bahnschweißen
BAS.SRC	Basis–Paket zum Initialisieren etc.
BOSCH.SRC	Programm für Punktschweißen mit serieller Schnittstelle zum Bosch Punktschweißtimer PSS5200.521C
CELL.DAT CELL.SRC	Programm zur Steuerung von Robotern über eine zentrale SPS
CLEANER.DAT CLEANER.SRC	
FLT_SERV.DAT FLT_SERV.SRC	Programm zur benutzerdefinierten Fehlerbehandlung beim Bahnschweißen
H50.SRC	Greifer–Paket
H70.SRC	Touchsensor–Paket
IR_STOPM.SRC	Programm zur Fehlerbehandlung beim Auftreten von Roboterstörungen
MSG_DEMO.SRC	Programm mit Beispielen für Benutzermeldungen
NEW_SERV.SRC	Programm zur Änderung von Startoptionen
P00.DAT P00.SRC	Programmpaket zur Steuerung über eine SPS
PERCEPT.SRC	Programm zum Aufruf des PERCEPTRON–Protokolls
SPS.SUB	Submitfile
USER_GRP.DAT USER_GRP.SRC	Programm zur benutzerdefinierten Greiferansteuerung
USERSPOT.SRC	Programmpaket zum Punktschweißem
WEAV_DEF.SRC	Programm für Pendelbewegungen

Tab. 1 Vorinstallierte Dateien im Verzeichnis R1

Pfad

Den aktuellen Pfad und das aktuelle Verzeichnis findet man stets links neben dem Programmfenster. In Abb. 4 werden im Programmfenster also gerade die Dateien des Pfades Roboter (also alle Dateien im RAM–Speicher) angezeigt. Wollen Sie z.B. eine Datei von der Festplatte oder Diskette laden, so wechseln Sie mit dem Menükey "Datei" und der Menüoption "PC" auf PC–Ebene (s. Abb. 5). Die verfügbaren Laufwerke erscheinen nun auf der linken Seite und Sie können sich die gewünschte Datei in das gewünschte Verzeichnis laden.

Abb. 5 Wechseln von PC– auf Roboterebene

Beim Erstellen eines neuen Programms auf Anwenderebene brauchten Sie nach Drücken des Softkeys "Neu" nur einen Namen für das neue Programm anzugeben. Dadurch wurden gleichzeitig zwei Dateien mit dem angegebenen Namen und den Dateierweiterungen ".SRC" und ".DAT" erzeugt.

Das SRC–File ist die Datei mit dem eigentlichen Programmcode. Das DAT–File enthält dagegen wichtige Programmdaten.

Dateikonzept

Diese Teilung beruht auf dem Dateikonzept von KRL: Das Programm eines Industrieroboters beinhaltet neben dem Bearbeitungsablauf verschiedene Aktionen, die der Industrieroboter ausführen soll. Dies kann ein Wegstück sein, das der Roboter abfahren soll, oder auch das Öffnen eines Spezialgreifers unter Berücksichtigung von gewissen Randbedingungen. Beim Testen von Programmen ist es unerläßlich, solche Teilaufgaben einzeln zum Ablauf bringen zu können. Um diesen speziellen Bedürfnissen bei der Roboterprogrammierung gerecht zu werden, wurde in KRL das Dateikonzept verwirklicht.

Datei

Eine Datei ist die Einheit, die der Programmierer programmiert. Eine Datei entspricht damit einem File auf der Festplatte oder im Speicher (RAM). Jedes Programm in KRL kann aus einem oder mehreren Dateien bestehen. Einfache Programme umfassen genau eine Datei. Komplexere Aufgaben löst man besser mit einem Programm, das aus mehreren Dateien besteht. Der innere Aufbau einer KRL–Datei besteht aus Vereinbarungsteil, Anweisungsteil sowie bis zu 255 lokalen Unterprogrammen und Funktionen (s. Abschnitt 8). Der Objektname ohne Erweiterung ist zugleich der Name der Datei und wird deshalb in der Vereinbarung mit

DEF

DEF angeführt. Der Name darf aus maximal 8 Zeichen bestehen und darf kein Schlüsselwort sein (s. Abschnitt 3.1). Jede Datei beginnt mit der Vereinbarung DEF und endet mit END:

```
DEF  NAME( )
Vereinbarungen
Anweisungen
END
```

Anweisung

Anweisungen haben im Gegensatz zu Vereinbarungen einen dynamischen Charakter: Sie werden bei der Programmbearbeitung ausgeführt.

Vereinbarung

Vereinbarungen werden bereits vor der Programmbearbeitung, d.h. während des Übersetzens, ausgewertet. Im Vereinbarungsteil darf daher keine Anweisung stehen. Die erste Anweisung ist zugleich der Beginn des Anweisungsteils.

Datenliste

Zu jeder Datei kann genau eine Datenliste gehören. Dies ist aber nicht zwingend erforderlich. Datenliste und Datei werden über den gemeinsamen Namen als zusammengehörig gekennzeichnet. Die Namen unterscheiden sich nur in der Erweiterung, z.B.:

Datei: PROG1.SRC
Datenliste: PROG1.DAT

In Datenlisten sind nur Vereinbarungen zulässig. Wenn die Datenliste und die Datei den gleichen Namen besitzen, können Variablen, die in der Datenliste vereinbart sind, auf die gleiche Weise verwendet werden wie Variablen, die in der Datei vereinbart sind.

Abschnitt 11 Datenlisten

Da ein Roboterprogramm auch ohne Datenliste geschrieben werden kann, werden Datei und Datenliste auf Expertenebene nicht automatisch gleichzeitig angelegt. Durch den Softkey "Neu" öffnen Sie daher ein Inline-Formular, in dem Sie neben dem Namen zumindest auch die Dateierweiterung angeben müssen (s. Abb. 6). Wollen Sie neben der Datei auch eine Datenliste benutzen, so müssen Sie auf Expertenebene daher zwei Dateien anlegen. Die Datenliste ist zwingend erforderlich, wenn Sie in ihrer SRC-Datei auch menügeführte Befehle einfügen wollen.

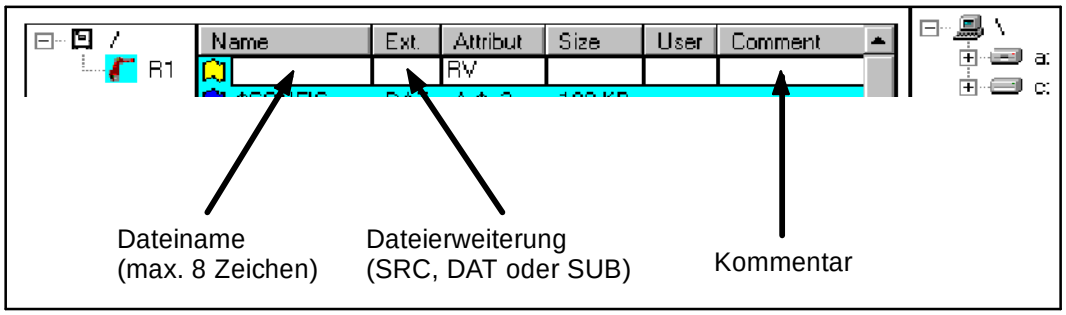


Abb. 6 Neuanlegen von Dateien oder Datenlisten

Submitdatei

Neben den Roboterdateien (SRC) gibt es auch Submitdateien. Submitdateien enthalten Programmanweisungen und können z.B. zur Steuerung der Peripherie (Greifer, etc.) genutzt werden. Submitdateien arbeiten parallel zum Betrieb des Roboters. Zur Kennzeichnung erhalten Submitdateien die Erweiterung ".SUB".

Compiler

Haben Sie sich eine Datei oder eine Datenliste mit "Neu" erstellt, so können Sie sie mit dem Editor bearbeiten. Beim Schließen des Editors wird der komplette Programmcode compiliert, d.h. der textuelle KRL-Code wird in eine für die Steuerung verständliche Maschinensprache übersetzt. Der Compiler überprüft den Code dabei auf syntaktische und semantische Richtigkeit. Sind Fehler vorhanden, so wird eine Meldung ausgegeben und eine Fehlerdatei mit der Dateierweiterung ".ERR" erzeugt (s. Abschnitt 2.4).

Binder

Nur wenn ein Programm fehlerfrei ist, kann es angewählt und ausgeführt werden. Beim Anwählen werden alle benötigten Dateien und Datenlisten zu einem Programm zusammengebunden. Beim Binden wird überprüft, ob alle Module vorhanden, analysiert und fehlerfrei sind. Außerdem überprüft der Binder bei einer Parameterübergabe die Typverträglichkeit zwischen den Übergabeparametern. Treten beim Binden Fehler auf, so wird – wie beim Compilieren – ein Error-File erstellt.

Beim Erstellen von Programmen mittels des "Neu"-Softkeys auf der Bedienoberfläche wird neben der Kopfzeile DEF¼ und END automatisch auch eine Initialisierungssequenz mit "INI" und das Anfahren einer HOME-Position am Anfang und Ende des Programms eingefügt. Dabei wird ein Rumpfprogramm (liegt in der Datei "Programme/KRC/TEMPLATE/vorgabe.src" bzw. "Programme/KRC/TEMPLATE/vorgabe.dat") auf den neuen Namen kopiert.

Selbstverständlich können Sie jedes KRL-Programm aber auch mittels eines ganz normalen Text-Editors schreiben und dann mit dem Softkey "Laden" in den Systemspeicher laden. In diesem Fall müssen Sie allerdings selbst darauf achten, daß alle notwendigen Initialisierungen (z.B. Achsgeschwindigkeiten) vorgenommen werden.

Ein einfaches Roboterprogramm könnte dann beispielsweise so aussehen:

```
DEF PROG1()

;----- Vereinbarungsteil -----
INT J

;------ Anweisungsteil -------
$VEL_AXIS[1]=100   ;Festlegung der Achsgeschwindigkeiten
$VEL_AXIS[2]=100
$VEL_AXIS[3]=100
$VEL_AXIS[4]=100
$VEL_AXIS[5]=100
$VEL_AXIS[6]=100

$ACC_AXIS[1]=100   ;Festlegung der Achsbeschleunigungen
$ACC_AXIS[2]=100
$ACC_AXIS[3]=100
$ACC_AXIS[4]=100
$ACC_AXIS[5]=100
$ACC_AXIS[6]=100

PTP {A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}

FOR J=1 TO 5
   PTP {A1 45}
   PTP {A2 -70,A3 50}
   PTP {A1 0,A2 -90,A3 90}
ENDFOR

PTP {A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}
END
```

NOTIZEN:

2.2 Ändern von Programmen

Es gibt grundsätzlich zwei Möglichkeiten, ein Programm auf der Expertenebene der Bedienoberfläche zu ändern. Mittels

- Editor
- oder Programm–Korrektur (PROKOR).

Programm–Korrektur ist die Standard–Methode. Wird ein Programm angewählt, oder ein laufendes Programm gestoppt, befindet man sich automatisch im PROKOR–Modus.

Hier kann man nur Befehle eingeben oder bearbeiten, die sich nur auf eine Programmzeile auswirken. Also keine Kontrollstrukturen (Schleifen etc.) oder Variablendeklarationen.

Zeileninterpreter

Grund hierfür ist, daß das Programm angewählt bleibt und die einzelnen Befehle im Grundsystem von einem Zeileninterpreter überpüft werden. Im Gegensatz zum Compiler kann der Zeileninterpreter nur einzelne Zeilen überprüfen. Zum Compilieren muß das Programm jedoch abgewählt sein.

Editor

Will man daher bestimmte KRL–Befehle oder Programmstrukturen bearbeiten oder einfügen, so wird man dies über den Editor tun. Da bei Schließen des Editors der komplette Code compiliert wird, können auch Fehler erkannt werden, die erst im Zusammenhang mehrerer Zeilen auftreten (z.B. falsch vereinbarte Variablen).

NOTIZEN:

2.3 Programmlaufarten

Die Programmlaufarten legen fest, ob die Programmbearbeitung

- satzweise
- schrittweise oder
- ohne Programmstop

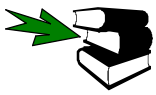
erfolgen soll. In Tab. 2 sind alle Programmlaufarten beschrieben.

Laufart	Beschreibung
ISTEP	Inkremental Step (Einzelsatz) Das Programm wird satzweise, d.h. mit einem STOP nach jeder Anweisung (auch Leerzeile) ausgeführt. Das Programm wird ohne Vorlauf bearbeitet.
MSTEP	Motion Step (Programmschritt) Das Programm wird schrittweise, d.h. mit einem STOP vor jedem Bewegungssatz ausgeführt. Das Programm wird ohne Vorlauf bearbeitet.
GO	Alle Anweisungen werden im Programm ohne STOP bis zum Programmende bearbeitet.

Tab. 2 Programmlaufarten

Die Programmlaufart kann am KCP oder softwaremäßig über die Aufzählungsvariable $PRO_MODE eingestellt werden, z.B.:

```
$PRO_MODE = #GO
```



Abschnitt 3.2.6 Aufzählungstypen

NOTIZEN:

2.4 Fehlerbehandlung

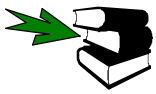
Fehlermeldung

Tritt ein Fehler beim Compilieren oder beim Binden auf, so wird im Meldungsfenster eine Fehlermeldung ausgegeben und eine Error-Datei mit einem Fehlerprotokoll erzeugt. Diese Datei hat den gleichen Namen wie die fehlerhafte Datei mit der Dateierweiterung ".ERR".

Datei: PROG2.SRC

Error-Datei: PROG2.ERR

Die Error-Datei können Sie sich mit dem Editor ansehen. Sie enthält die fehlerhafte Zeile, die dazugehörige Zeilennummer, den Dateityp (SRC, DAT oder SUB) und eine Fehlernummer, die einen Hinweis auf die Fehlerart gibt.



Dokumentation der Fehlermeldungen, Störungsbehebung

Als Beispiel diene eine Datei ERROR.SRC. In ihr wurde vergessen, die Schleifenvariable I zu deklarieren und in der Zuweisung I == I + 1 wurde ein Gleichheitszeichen zuviel gesetzt (s. Abb. 7).

```
 1   INI
 2   PTP HOME  Vel= 100 % DEFAULT
 3
 4   FOR I=1 TO  4
 5   $OUT[I]=TRUE
 6   ENDFOR
 7
 8   I == I + 1
 9
10   PTP HOME  Vel= 100 % DEFAULT
11   END
```

Abb. 7 Datei ERROR.SRC mit Fehlern

Nach Schließen des Editors erscheint im Meldungsfenster eine Fehlermeldung und es wird eine Fehlerdatei ERROR.ERR angelegt (s. Abb. 8).

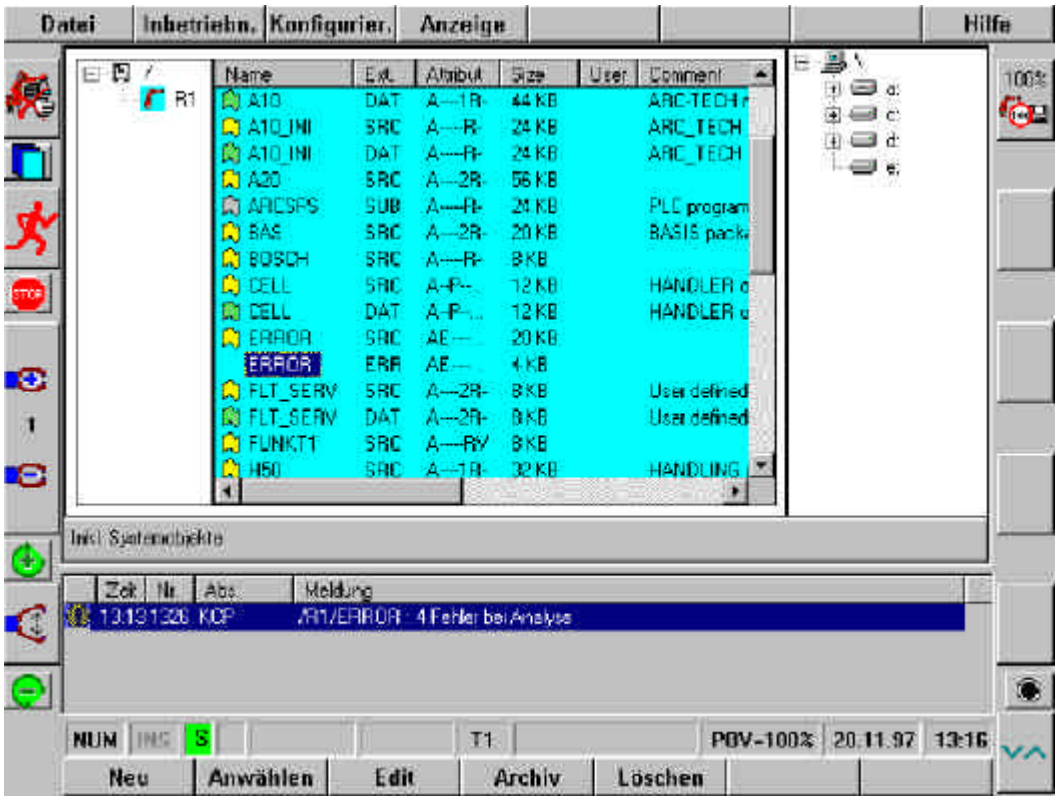


Abb. 8 Datei ERROR.ERR wurde angelegt

Fehlerprotokoll

Mit dem Editor können Sie sich nun das Fehlerprotokoll ansehen (s. Abb. 9).

```
1    SRC 58□FOR I=1 TO  4□2137(  5) □2263(  5)
2    SRC 59□$OUT[I]=TRUE□2137(  6)
3    SRC 62□***I == I + 1□2309(  6)
```

Abb. 9 Inhalt der Datei ERROR.ERR

Aus dem Fehlerprotokoll ist ersichtlich, daß

- 3 fehlerhafte Zeilen im zugehörigen SRC-File vorhanden sind,
- die Zeilennummern der fehlerhaften Zeilen 58, 59 und 62 sind,
- in Zeile 58 die Fehlermeldungen
  - 2137: Name nicht als einfache Variable deklariert
  - 2263: Typ der Laufschleifenvariable ungleich INT
- in Zeile 59 die Fehlermeldung
  - 2137: Name nicht als einfache Variable deklariert
- in Zeile 62 die Fehlermeldung
  - 2309: '(' erwartet

aufgetreten sind. Aus den Fehlermeldungen 2137 und 2263 ergibt sich sehr schnell, daß die Variable I nicht als Integer deklariert wurde. Die Meldung 2309 bedeutet: Der Compiler interpretiert die Zeile als Unterprogrammaufruf, bei dem allerdings die Klammern fehlen.

Nun kann man sich den SRC-File wieder in den Editor laden, und die entsprechenden Korrekturen vornehmen. Bei den Zeilennummern ist zu beachten, daß alle Folds geöffnet sind (Menükey "Bearbeiten/alle Folds öffnen") und die begrenzte Sichtbarkeit ausgeschaltet ist (Menükey "Hilfe/Limited Visibility"), (s. Abschnitt 12). Als weitere Hilfe werden bei bestimmten Fehlern drei Sterne "***" am Anfang der fehlerhaften Zeile eingefügt (s. Abb. 10).

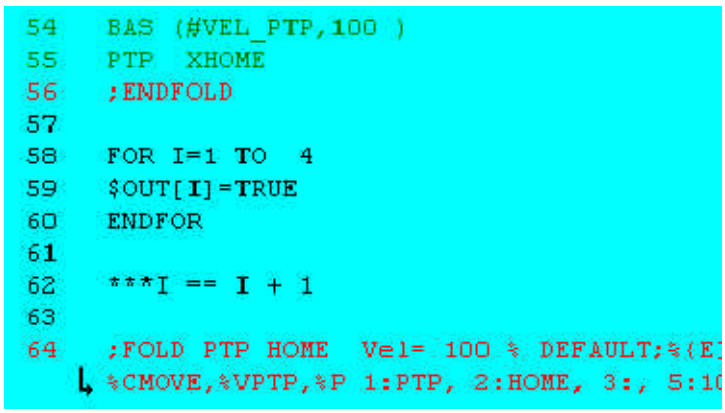


Abb. 10 Ansicht von ERROR.SRC bei geöffneten Folds und ausgeschalteter "Limited Visibility"

Fügen Sie also die Zeile

```
INT I
```

vor der INI-Zeile ein, und streichen Sie in Zeile 62 ein Gleichheitszeichen.

Das Löschen der Sterne ist nicht nötig, da diese automatisch beim Verlassen des Editors entfernt werden.

```
I = I + 1
```

Nach dem Verbessern der Fehler wird die Fehlerdatei ERROR.ERR wieder automatisch gelöscht (evtl. ist ein Auffrischen der Dateienliste mit dem Menükey "Hilfe/Refresh" erforderlich).

2.5 Kommentare

Kommentare sind eine wichtiger Bestandteil jedes Computerprogrammes. Dadurch können Sie Ihr Programm übersichtlich und auch für andere verständlich machen. Die Bearbeitungsgeschwindigkeit des Programmes wird durch Kommentare nicht beeinflußt.

Strichpunkt

Kommentare können Sie an jeder Stelle eines Programmes einfügen. Sie werden stets mit einem Strichpunkt ";" eingeleitet, z.B.:

```
¼
PTP P1                ;Bewegung zum Ausgangspunkt
¼
;--- Ausgaenge zuruecksetzen ---
FOR I = 1 TO 16
    $OUT[I] = FALSE
ENDFOR
¼
```