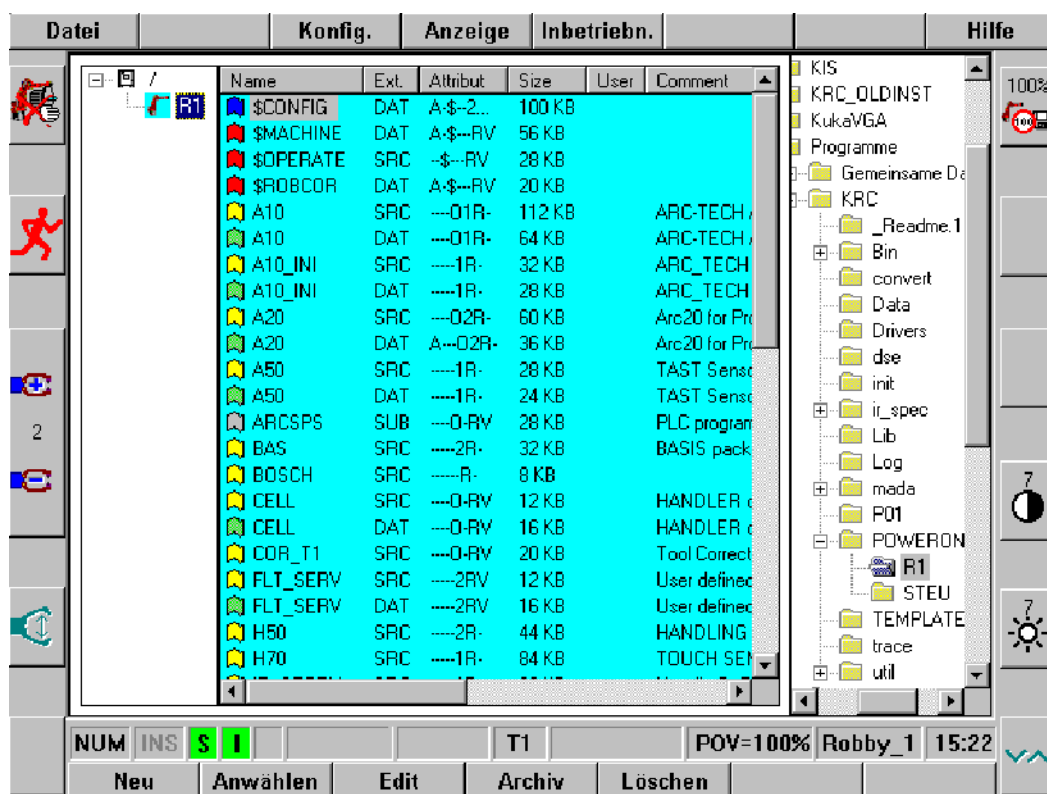


2 Allgemeines zu KRL-Programmen

2.1 Aufbau und Struktur von Programmen

2.1.1 Programmoberfläche

Sobald Sie auf die Expertenebene gewechselt haben, verändert sich die Bedienoberfläche wie nachfolgend dargestellt:



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.

In vorstehender Abbildung werden im Programmfenster die Dateien des Pfades "Roboter" angezeigt.

Standardmäßig sind nach der Installation der KRC1-Software im Verzeichnis "C:\Programme\KRC\PowerOn\R1\" die nachfolgend aufgeführten Dateien vorhanden:

Datei	Bedeutung
\$CONFIG.DAT	Systemdatenliste mit allgemeinen Konfigurationsdaten
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	Technologiepaket 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
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 Punktschweißen
WEAV_DEF.SRC	Programm für Pendelbewegungen

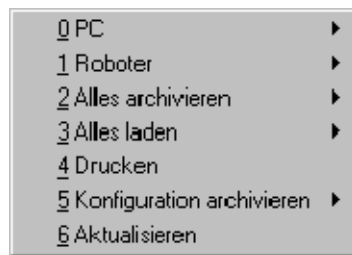
Im Verzeichnis "C:\Programme\KRC\MaDa\R1\" befinden sich folgende Dateien:

Datei	Bedeutung
\$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

2.1.2 Pfad

Datei

Nach dem Systemhochlauf ist automatisch ein Dateipfad zum Aufruf und zur Speicherung der Programme voreingestellt. Wollen Sie diesen ändern, betätigen Sie den Menükey "Datei" und wählen die Option "PC" bzw. "Roboter" aus.



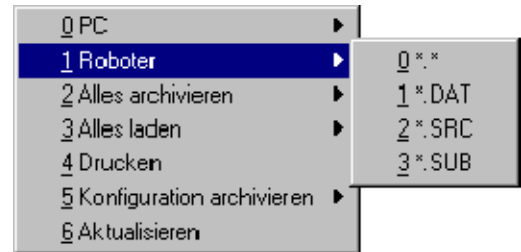
Die Programmdateien werden aus dem Verzeichnis C: \Programme\KRC\PowerOn\R1 gelesen bzw. dort gespeichert.

Die Programmdateien werden im Arbeitsspeicher des Robotersystems abgelegt bzw. von dort eingelesen.



Nach dem Systemhochlauf ist der Pfad "Roboter" voreingestellt. Diese Einstellung sollten Sie beibehalten.

Auf der Benutzerebene "Experte" stehen Ihnen zusätzlich Filterfunktionen zur Verfügung, die eine übersichtlichere Darstellung der Dateien ermöglichen. Dazu betätigen Sie nach Auswahl der Menüoption "PC" oder "Roboter" die "Eingabe"-Taste, bzw. die "Cursor"-Taste "→". Es öffnet sich dann ein Auswahlmenü, in dem Sie dann den anzuzeigenden Dateityp auswählen:



- G *. * alle Programme bzw. Dateien
- G DAT nur die Dateilisten im gewählten Verzeichnis
- G SRC nur die vorhandenen SRC's
- G SUB nur die Submit-Dateien



Den aktuellen Pfad sowie das aktuelle Verzeichnis finden Sie immer links neben dem Programmfenster. Wollen Sie beispielsweise eine Datei von der Festplatte oder Diskette laden, so wechseln Sie mit dem Befehl "Datei" -> "PC" auf die PC-Ebene. Die verfügbaren Laufwerke werden anschließend auf der linken Seite angezeigt, und Sie können den Inhalt des gewünschten Verzeichnisses im Programmfenster einsehen.

2.1.3 Dateikonzept

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

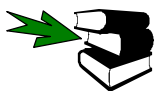
Sie können im Feld "Ext." auch explizit eine der zulässigen Dateierweiterungen (z.B. ".SRC", ".DAT") eintragen. Die entsprechende Datei wird dann erzeugt. Bei Eingabe einer unzulässigen Dateierweiterung wird eine entsprechende Hinweismeldung ausgegeben.

Das "SRC"-File ist hierbei die Datei mit dem eigentlichen Programmcode. Das "DAT"-File enthält dagegen die spezifischen Programmdateien. Diese Teilung beruht auf dem Dateikonzept von KRL: Das Programm beinhaltet neben dem Bearbeitungsablauf verschiedene Aktionen, die der Industrieroboter ausführen soll. Dies können bestimmte Bewegungsabläufe sein, das Öffnen oder Schließen eines Greiferwerkzeugs, bis hin zu komplexen Abläufen, wie beispielsweise die Steuerung einer Schweißzange unter Berücksichtigung der Randbedingungen.

Zum Testen von Programmen ist es hilfreich bzw. erforderlich, Teilaufgaben einzeln zum Ablauf bringen zu können. Das in KRL realisierte Dateikonzept wird den speziellen Bedürfnissen der Roboterprogrammierung gerecht.

2.1.4 Dateistruktur

Eine Datei ist die Einheit, welche der Programmierer erstellt und entspricht damit einer Datei 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.



Detaillierte Informationen über Unterprogramme und Funktionen finden Sie im Abschnitt **[Unterprogramme und Funktionen]**



Der innere Aufbau einer KRL-Datei besteht aus Vereinbarungsteil, Anweisungsteil sowie bis zu 255 lokalen Unterprogrammen und Funktionen.

DEF Der Objektname ohne Erweiterung ist zugleich der Name der Datei und wird deshalb in der Vereinbarung mit "DEF" angeführt. Der Name darf aus maximal 8 Zeichen bestehen und darf kein Schlüsselwort sein (siehe Abschnitt **[Variablen und Vereinbarungen]**). Jede Datei beginnt mit der Vereinbarung "DEF" und endet mit "END".

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

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.

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

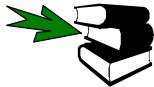
Datenliste Ein Roboterprogramm kann aus einer Programmdatei alleine oder aus einer Programmdatei mit zugehöriger Datenliste bestehen. Datenliste und Datei werden über den gemeinsamen

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 sowie Anweisungen "=" 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.



Detaillierte Informationen zum Thema finden Sie im Abschnitt **[Datenlisten]**.



NOTIZEN:

2.2 Programme erstellen und editieren

2.2.1 Erstellen eines neuen Programms

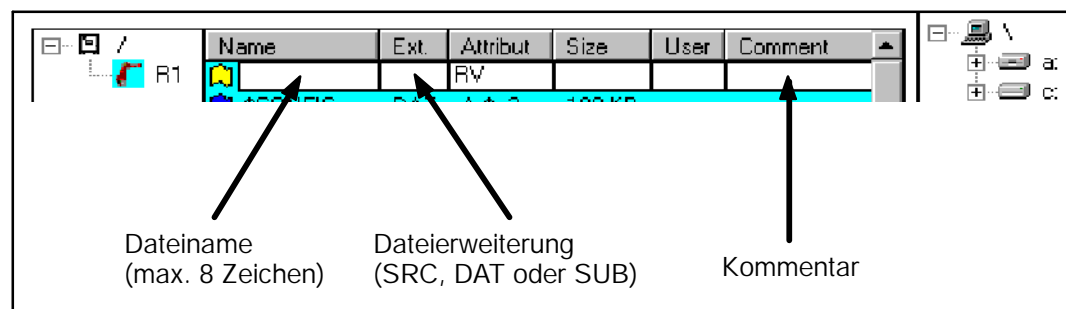
Neu

Da ein Roboterprogramm auch ohne Datenliste geschrieben werden kann, werden Datei und Datenliste auf Expertenebene nicht automatisch gleichzeitig angelegt. Um ein Programm anzulegen, betätigen Sie den Softkey "Neu". Es wird Ihnen standardmäßig die folgende Softkeyleiste angeboten:



Standard

Der Softkey "Standard" öffnet ein InLine-Formular, in dem Name, Dateierweiterung und ein Kommentar angegeben werden können.



Der Dateiname ist als einziges zwingend erforderlich. Wird keine Dateierweiterung angegeben, werden sowohl eine "SRC"- als auch eine "DAT"-Datei angelegt.



Die Datenliste ist zwingend erforderlich, wenn Sie in ihrer SRC-Datei auch menügeführte Befehle einfügen wollen.

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".

Kopieren

Der Softkey "Kopieren" gestattet eine Kopie einer vorhandenen Datei unter anderem Namen zu erstellen. Anschließend kann diese Datei angewählt oder editiert werden.



Dieser Softkey beendet die Programmerstellung und stellt die vorherige Softkeyleiste wieder her.

2.2.2 Programm editieren, kompilieren und binden

Haben Sie sich eine Datei oder eine Datenliste mit "Neu" erstellt, so können Sie sie mit dem Editor bearbeiten. Hierzu dient der Softkey "Edit". Beim Schließen des Editors wird der komplette Programmcode kompiliert, d.h. der textuelle KRL-Code wird in eine für die Steuerung verständliche Maschinensprache übersetzt.

Compiler

Der Compiler überprüft den Code dabei auf syntaktische und semantische Richtigkeit. Falls Fehler vorhanden sind, wird eine entsprechende Meldung ausgegeben und eine Fehlerdatei mit der Dateierweiterung ".ERR" erzeugt.



Nur fehlerfreie Programme können angewählt und ausgeführt werden.



Weitere Informationen zur Behandlung von Editier-Fehlern finden Sie im Abschnitt **[Fehlerbehandlung]**.

Binder Beim Laden eines Programms über den Softkey "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 Kompilieren – ein Error-File mit der Erweiterung ".ERR" erzeugt.



Beim Erstellen von Programmen mittels der "Neu" -> "Standard"-Softkeys auf der Bedienoberfläche wird neben der Kopfzeile **DEF...** und **END** automatisch auch eine Initialisierungssequenz mit "INI" und das Anfahren einer HOME-Position am Anfang und Ende des Programms eingefügt.

Hierzu wird die Datei "C:\Programme\KRC\TEMPLATE\Vorgabe.src" bzw. "C:\Programme\KRC\TEMPLATE\Vorgabe.dat" als Kopiervorlage verwendet.



Sie können ein KRL-Programm auch mit jedem üblichen Text-Editor schreiben und anschließend 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:

2.3 Ändern von Programmen

Grundsätzlich gibt es zwei Möglichkeiten, ein Programm auf der Expertenebene der Bedienoberfläche zu ändern:

- G Programm-Korrektur (PROKOR)
- G Editor

2.3.1 Programm-Korrektur

Die Programm-Korrektur ist eine 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.

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

2.3.2 Editor

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

2.3.2.1 Blockfunktionen



Die Blockfunktionen stehen nur im Editor ab der Benutzerebene "Experte" zur Verfügung. Sie müssen ein Programm, dessen Inhalt Sie mit Hilfe der Blockfunktionen ändern wollen, mit dem Softkey "Edit" öffnen. Wie Sie zuvor in die Benutzerebene "Experte" wechseln, wird im Kapitel **[System konfigurieren]**, Abschnitt **"Benutzergruppe"** beschrieben.

Setzen Sie zunächst den blinkenden Editiercursor an den Anfang oder das Ende des zu verschiebenden Programnteils. Halten Sie dann die "Shift"-Taste auf der Tastatur gedrückt, während Sie den Cursor nach unten, bzw. oben bewegen. Auf diese Weise markieren Sie einen Programnteil, der im nächsten Arbeitsschritt mit den Blockfunktionen bearbeitet werden soll. Den bereits markierten Teil erkennen Sie an der farblichen Hervorhebung.

Bearbeiten

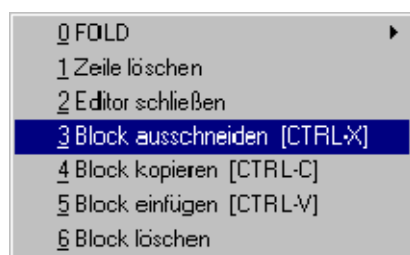
Betätigen Sie den Menükey "Bearbeiten" und wählen Sie aus dem sich öffnenden Menü die gewünschte Funktion aus.



Werden für die Blockfunktionen der Nummernblock und das Tastaturfeld verwendet, muß die NUM-Funktion ausgeschaltet sein. Drücken Sie in diesem Fall die "Num"-Taste auf dem Tastaturfeld. Die entsprechende Anzeige in der Statuszeile ist dann ausgeschaltet.

2.3.2.2 Block ausschneiden (CTRL-X)

Wählen Sie aus dem Menü die Option "Block ausschneiden" aus, wird der markierte Programnteil zwischengespeichert und aus dem Programmlisting gelöscht.





Alternativ können Sie die CTRL-Taste auf dem Nummernblock gedrückt halten und auf der Tastatur die X-Taste drücken. Lassen Sie anschließend beide Tasten los.

2.3.2.3 Block kopieren (CTRL-C)

Der markierte Programnteil wird zur weiteren Verarbeitung zwischengespeichert. Er kann anschließend an anderer Stelle eingefügt werden.



Alternativ können Sie die CTRL-Taste auf dem Nummernblock gedrückt halten und auf der Tastatur die C-Taste drücken. Lassen Sie anschließend beide Tasten los.

2.3.2.4 Block einfügen (CTRL-V)

Setzen Sie den Editiercursor an die Stelle, an welcher der zuvor "herausgeschnittene" oder "kopierte" Programnteil wieder eingefügt werden soll.



Wählen Sie jetzt die Option "Block einfügen" aus. Der vorher markierte Programnteil wird unterhalb des Editier-Cursors wieder eingefügt.



Alternativ können Sie die CTRL-Taste auf dem Nummernblock gedrückt halten und auf der Tastatur die V-Taste drücken. Lassen Sie anschließend beide Tasten los.

2.3.2.5 Block löschen

Der markierte Bereich kann aus dem Programm entfernt werden. In diesem Fall erfolgt keine Zwischenspeicherung. Der entfernte Programnteil ist damit unwiderruflich verloren.



Aus diesem Grund erfolgt eine Sicherheitsabfrage im Meldungsfenster, die über die Softkey-
leiste beantwortet werden muß.



- G **Ja** Der markierte Bereich wird unwiderruflich gelöscht;
- G **Nein** Die Funktion "Block löschen" wird abgebrochen;
- G **Alles Quitt** Löscht alle quittierbaren Meldungen im Meldungsfenster.



Wählen Sie aus dem Menü die Option "Block löschen" aus, wird der markierte Programm-
teil ohne Zwischenspeicherung aus dem Programmlisting gelöscht.



NOTIZEN:

2.4 Verstecken von Programmteilen

Anders als bei normalen Editoren gestattet der KCP-Editor eine anforderungsspezifische Anzeige der Programminhalte. So sieht zum Beispiel der Anwender nur die wesentlichen Inhalte eines Programms, während auf der Expertenebene das gesamte Programm einsehbar ist.

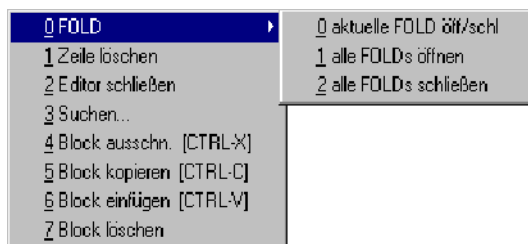
2.4.1 FOLD

Die KUKA-Bedienoberfläche benutzt eine besondere Technik zur übersichtlichen Darstellung eines Programmes. Als KRL-Kommentare markierte Anweisungen erlauben es, die Anzeige von nachfolgenden Teilen des Programmes zu unterdrücken. Das Programm wird so in sinnvolle Abschnitte unterteilt, die entsprechend ihrem Ordner-Charakter "FOLDS" genannt werden.

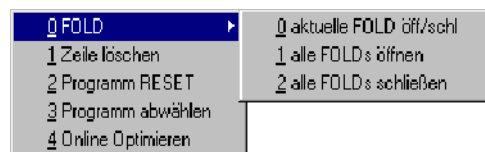
"FOLDS" sind standardmäßig "geschlossen" und können nur auf der Expertenebene "geöffnet" werden. Sie enthalten Informationen die für den Anwender auf der KUKA-Bedienoberfläche (KUKA-BOF) unsichtbar sind. Auf der Expertenebene haben Sie die Möglichkeit einen KRL-Block für den Anwender unsichtbar zu machen. Hierfür werden die betreffenden Vereinbarungen oder Anweisungen durch die Bezeichnungen **" ; FOLD "** und **" ; ENDFOLD "** eingeschlossen.

Bearbeiten

Folds in einem Programm können angezeigt oder versteckt werden, wenn Sie den Menükey "Bearbeiten" drücken und anschließend "FOLD" sowie das gewünschte Kommando anwählen.



Programm im Editor



Programm Angewählt

Folgende Optionen stehen zur Auswahl:

- | | | |
|---|-------------------------------|--|
| G | aktuelle FOLD öff/schl | öffnet bzw. schließt den FOLD der Zeile, in der sich der Editier-Cursor befindet |
| G | alle FOLDS öffnen | öffnet alle FOLDS des Programms |
| G | alle FOLDS schließen | schließt alle FOLDS des Programms |

Von der Sequenz...

; FOLD RESET OUT

```
FOR I=1 TO 16
  SOUT[I]=FALSE
ENDFOR
```

; ENDFOLD

...sind bei geschlossenen Folds auf der Bedienoberfläche nur die Worte **"RESET OUT"** zu sehen. Mit diesem Befehl können Sie beispielsweise Deklarations- und Initialisierungsteil für den Anwender unsichtbar machen.

2.4.1.1 Beispielprogramm



```

DEF FOLDS()

; FOLD DECLARATION; % weitere Informationen
;----- Deklarationsteil -----
EXT BAS (BAS_COMMAND :IN, REAL :IN )
DECL AXIS HOME
INT I
; ENDFOLD

; FOLD INITIALISATION
;----- Initialisierung -----
INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
INTERRUPT ON 3
BAS (#INITMOV, 0 ) ;Initialisierung von Geschwindigkeiten,
; Beschleunigungen, $BASE, $TOOL, etc.

FOR I=1 TO 16
  $OUT[I]=FALSE
ENDFOR
HOME={AXIS: A1 0, A2 -90, A3 90, A4 0, A5 30, A6 0}
; ENDFOLD

;----- Hauptteil -----
PTP HOME ; SAK-Fahrt
LIN {X 540, Y 630, Z 1500, A 0, B 90, C 0}
PTP HOME

END

```

Das Beispielprogramm sieht auf der Bedienoberfläche folgendermaßen aus:

```

1
2  DECLARATION
3
4  INITIALISATION
5
6  ;----- Hauptteil -----

```

Das gleiche Programm mit geöffneten Folds:

```

1
2  DECLARATION
3  ;----- Deklarationsteil -----
4  EXT BAS (BAS_COMMAND :IN, REAL :IN )
5  DECL AXIS HOME
6  INT I
7
8  INITIALISATION
9  ;----- Initialisierung -----
10 INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
11 INTERRUPT ON 3
12 BAS (#INITMOV, 0 ) ;Initialisierung von Geschwindigkeiten,
13 ; Beschleunigungen, $BASE, $TOOL, etc.
14 FOR I=1 TO 16
15   $OUT[I]=FALSE
16 ENDFOR
17 HOME={AXIS: A1 0, A2 -90, A3 90, A4 0, A5 0, A6 0}
18
19 ;----- Hauptteil -----

```

Im geschlossenen FOLD ist nur der Ausdruck nach dem Schlüsselwort "FOLD" sichtbar. Im geöffneten FOLD sind dagegen alle Anweisungen und Vereinbarungen zu sehen.

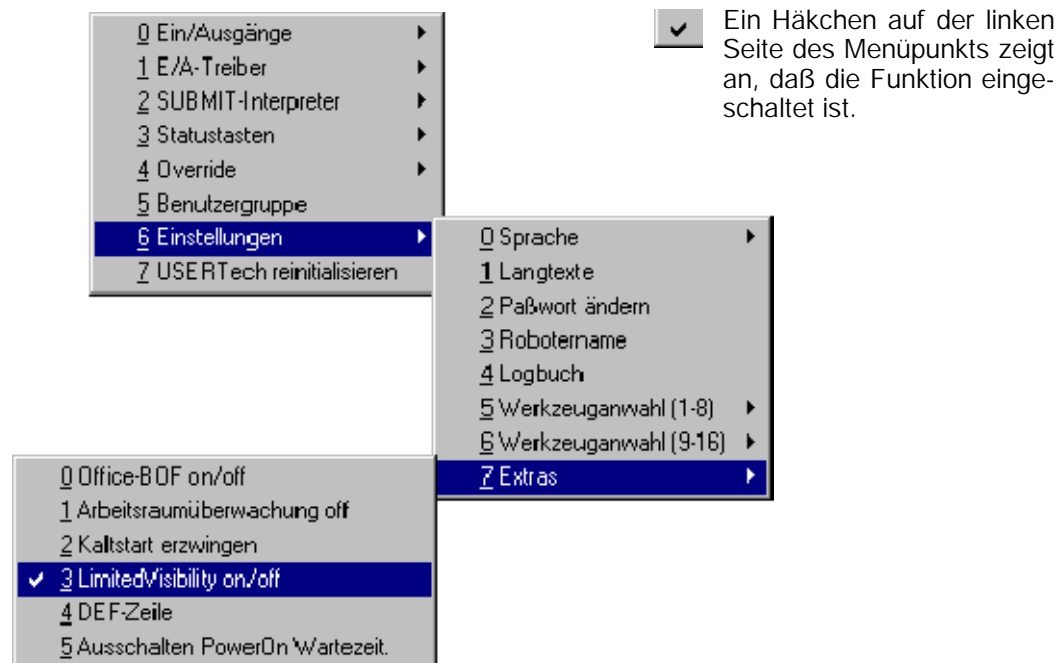


"FOLD" ist lediglich eine Anweisung für den Editor. Der Compiler interpretiert die FOLD-Anweisungen aufgrund des vorangestellten Semikolons als normalen Kommentar.

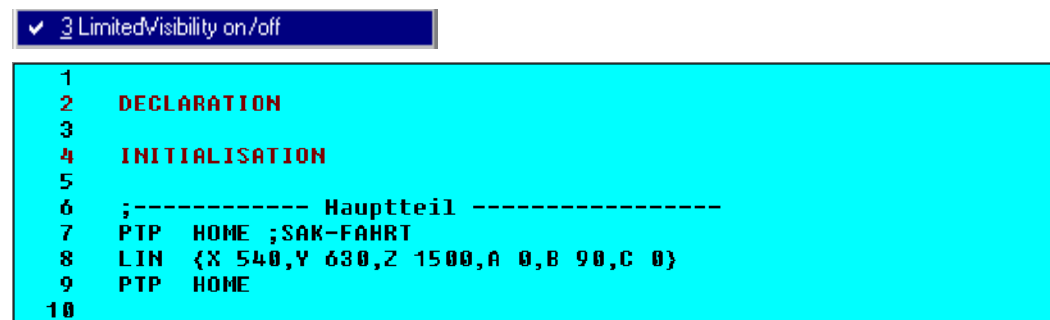
2.4.2 Beschränkung der Informationsmenge (LimitedVisibility)

Konfig.

Ein weiteres Hilfsmittel um die Informationsmenge auf der Bedienoberfläche möglichst gering zu halten, ist die Funktion "LimitedVisibility on/off". Durch Drücken des Menükeys "Konfigurieren" gelangen Sie über "Einstellungen" -> "Extras" zur gewünschten Funktion.



"LimitedVisibility on/off" ist standardmäßig aktiviert und kann nur auf der Expertenebene ausgeschaltet werden. "LimitedVisibility on/off" unterdrückt z.B. alle Texte in einer FOLD-Zeile, welche nach den Zeichen ";" geschrieben werden. Diese Informationen werden aber zur Anzeige eines Inline-Formulars benötigt.



3 LimitedVisibility on/off

```

1  &ACCESS  RU0
2  DEF  FOLDS ( )
3
4  ;FOLD DECLARATION;% weitere Informationen
5
6  ;FOLD INITIALISATION
7
8  ;----- Hauptteil -----
9  PTP  HOME ;SAK-FAHRT
10  LIN  {X 540,Y 630,Z 1500,A 0,B 90,C 0}
11  PTP  HOME
12
13  END

```



Erst wenn alle FOLDS geöffnet sind **und** "LimitedVisibility on/off" ausgeschaltet ist, sind dem Programmierer alle vorhandenen Programmzeilen verfügbar. Die Darstellung auf der Bedienoberfläche entspricht dann der Darstellung in einem normalen Texteditor.

```

1  &ACCESS  RU0
2  DEF  FOLDS ( )
3
4  ;FOLD DECLARATION;% weitere Informationen
5  ;----- Deklarationsteil -----
6  EXT  BAS (BAS_COMMAND :IN,REAL :IN )
7  DECL AXIS HOME
8  INT I
9  ;ENDFOLD
10
11 ;FOLD INITIALISATION
12 ;----- Initialisierung -----
13 INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
14 INTERRUPT ON 3
15 BAS (#INITMOV,0 ) ;Initialisierung von Geschwindigkeiten,
16 ;Beschleunigungen, $BASE, $TOOL, etc.
17 FOR I=1 TO 16
18 $OUT[I]=FALSE
19 ENDFOR

```

/R1/FOLDS.SRC

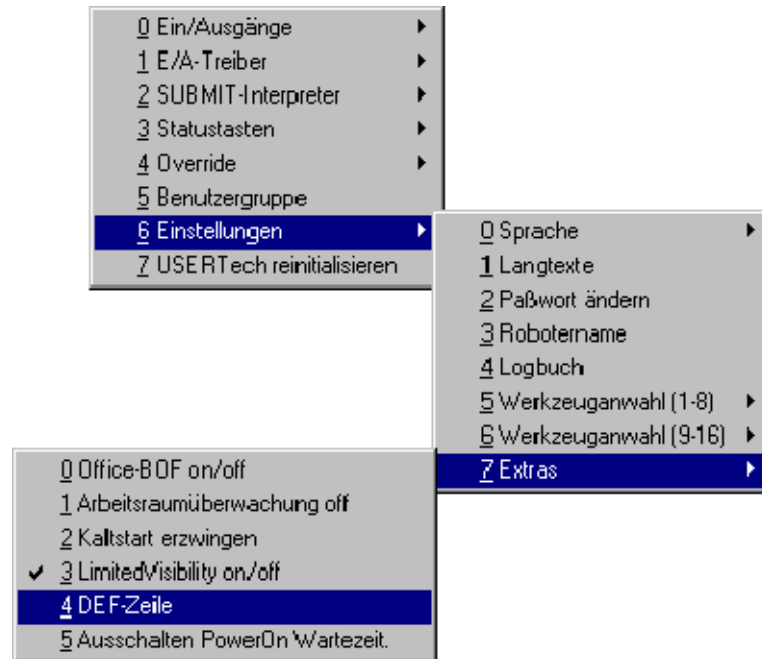
Ln 3, Col 0



2.4.3 DEF-Zeile sichtbar/unsichtbar

Konfig.

Ebenfalls hilfreich ist die Funktion "DEF-Zeile". Diese Option blendet die "DEF"- und "END"-Zeilen ein. Drücken Sie hierzu den Menükey "Konfig." und wählen die Option "Einstellungen" -> "Extras" -> "DEF-Zeile" aus.



Mit dem Beispielpogramm aus Abschnitt 2.4.1.1 sieht der Inhalt des Programmfensters folgendermaßen aus:

```

4 DEF-Zeile
1
2  DECLARATION
3
4  INITIALISATION
5
6  ;----- Hauptteil -----
7  PTP HOME ;SAK-FAHRT
8  LIN {X 540,Y 630,Z 1500,A 0,B 90,C 0}
9  PTP HOME
10

```

```

✓ 4 DEF-Zeile
1  DEF FOLDS ( )
2
3  DECLARATION
4
5  INITIALISATION
6
7  ;----- Hauptteil -----
8  PTP HOME ;SAK-FAHRT
9  LIN {X 540,Y 630,Z 1500,A 0,B 90,C 0}
10 PTP HOME
11
12 END

```




Erst wenn die DEF-Zeile sichtbar ist, können Deklarationen vorgenommen werden. Nach einem Neustart des Systems oder dem Wechsel auf die Benutzergruppe "Anwender" wird diese Funktion automatisch deaktiviert.



NOTIZEN:

2.5 Programmbeispiel

Nachfolgend ein einfaches Programmbeispiel zur Festlegung von Achsgeschwindigkeiten und Achsbeschleunigungen:

```

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:

2.6 Programmablaufarten



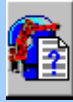
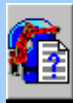
Die Programmablaufart legt fest, ob die Programmbearbeitung

G ohne Programmstop,

G schrittweise oder

G satzweise

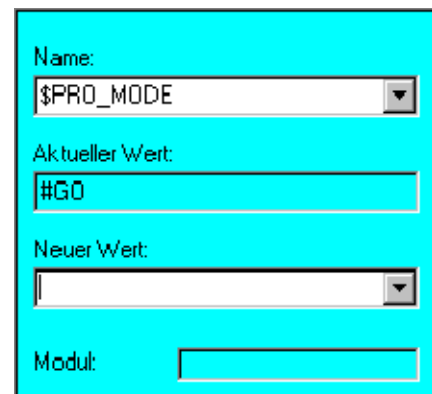
erfolgen soll. In nachfolgender Tabelle sind alle Programmablaufarten beschrieben.

Ablaufart	Beschreibung
GO 	Alle Anweisungen werden im Programm ohne STOP bis zum Programmende bearbeitet.
MSTEP 	Motion Step (Bewegungssatz) Das Programm wird schrittweise, d.h. mit einem STOP nach jedem Bewegungssatz ausgeführt. Das Programm wird ohne Vorlauf bearbeitet.
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.
PSTEP 	Program Step (Programmschritt) Unterprogramme werden komplett abgefahren. Das Programm wird ohne Vorlauf bearbeitet.
CSTEP 	Continuous Step (Bewegungssatz) Das Programm wird schrittweise, d.h. mit einem STOP nach jedem Bewegungssatz ausgeführt. Das Programm mit Vorlauf abgearbeitet, d.h. Punkte werden überschliffen.

Die Programmablaufart kann am KCP oder softwaremäßig über die Aufzählungsvariable "\$PRO_MODE" eingestellt werden. Es stehen die Werte "#GO", "#MSTEP", "#ISTEP", "#PSTEP" sowie "#CSTEP" zur Auswahl.

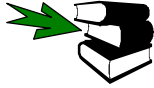
Zum Ändern des jeweiligen Zustands rufen Sie die Menüfunktion "Anzeige" -> "Variable" -> "Korrigieren" auf. Geben Sie anschließend im Eingabefeld "Name" die Variable "\$PRO_MODE" und im Feld "Neuer Wert" den gewünschten Wert ein.

Anzeige



Die Programmablaufarten "#PSTEP" und "#CSTEP" können nur über die Variablenkorrektur und nicht per Statuskey ausgewählt werden.



Näheres finden Sie im Kapitel **[Variablen und Vereinbarungen]**, Abschnitt **[Datenobjekte]** unter **"Aufzählungstypen"**.



NOTIZEN:

2.7 Fehlerbehandlung

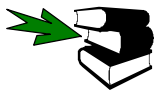


Tritt ein Fehler beim Kompilieren oder Binden auf, so wird im Meldungsfenster eine Fehlermeldung ausgegeben und eine Error-Datei mit einem Fehlerprotokoll erzeugt. Diese Datei hat den gleichen Namen wie die fehlerhafte Datei mit der Dateierweiterung ".ERR".

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) sowie eine auf die Fehlerart verweisende Fehlernummer.



Nähere Informationen zu den Fehlermeldungen finden Sie im [Anhang], Kapitel [Fehlermeldungen, Fehlerbehebung].



Als Beispiel dient die Datei "ERROR. SRC".

```

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

```

Nach Schließen des Editors erscheint nun im Meldungsfenster eine Hinweismeldung über die Anzahl der Fehler.

	Zeit	Nr.	Abs.	Meldung
	11:17:0		UserMode1	Benutzergruppe: Experte
	11:35:1326	KCP	/R1/ERROR :	3 Fehler bei Analyse

Gleichzeitig wird bei diesem Vorgang ein Fehlerprotokoll "ERROR. ERR" erzeugt.

Name	Ext.	Attribut	Size	User	Comment
ERROR	ERR	E--O-RV	4 KB		
ERROR	SRC	E--O-RV	24 KB		
ERROR	DAT	E--O-RV	28 KB		

Fehlerprotokoll

Dieses Fehlerprotokoll kann im Editor eingesehen werden.

```

1  &ACCESS RUO
2  SRC 51■FOR I=1 TO 4■2137( 5) ■2263( 5)
3  SRC 52■$OUT[I]=TRUE■2137( 6)
4  SRC 55■***I == I + 1■2309( 6)

```

Fehlernummern

Zeilennummer *1

Inhalt Programmzeile



HINWEIS

*1 Die angezeigten Zeilennummern entsprechen den absoluten Zeilennummern im Programm, wie sie ein normaler ASCII-Editor anzeigen würde. Die im KCP-Editor angezeigten Zeilennummern weichen aufgrund unterschiedlicher Konfigurationseinstellungen (z.B. Fold, LimitedVisibility usw.) voneinander ab, weil die Nummernfolge für die aktuell angezeigten Programmzeilen jeweils neu generiert wird.

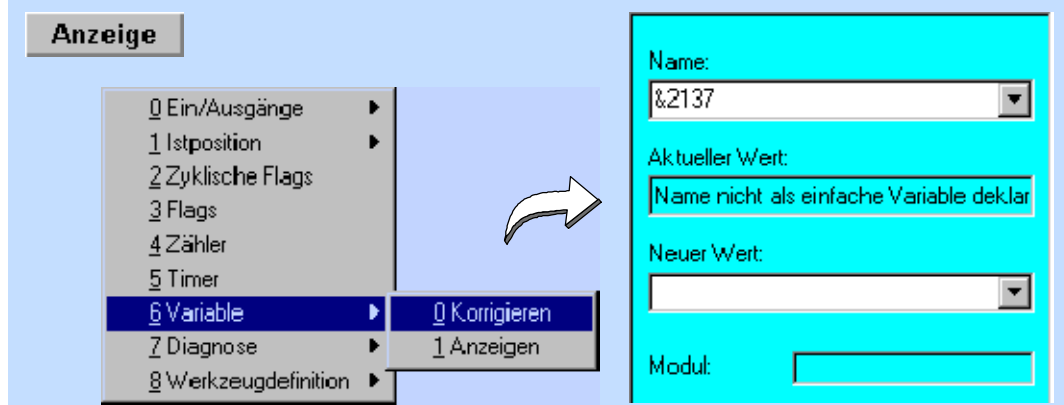
Aus dem Fehlerprotokoll ist ersichtlich, daß folgende Fehler aufgetreten sind:

- G 3 Zeilen im SRC-File sind fehlerhaft;
- G die Zeilennummern der fehlerhaften Zeilen lauten 51, 52 und 55;
- G in Zeile 51 die Fehlernummern
 - 2137: Name nicht als einfache Variable deklariert;
 - 2263: Typ der Laufschleifenvariable ungleich INT;
- G in Zeile 52 die Fehlernummer
 - 2137: Name nicht als einfache Variable deklariert;
- G in Zeile 55 die Fehlermeldung
 - 2309: das Zeichen "(" erwartet;

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.



Die Bedeutung der Fehlernummern können Sie Online mit Hilfe der Menüfunktion "Anzeige" -> "Variable" -> "Korrigieren" anzeigen lassen. Geben Sie hierzu im Zustandsfenster im Eingabefeld "Name" das Zeichen "&" gefolgt von der Fehlernummer ein. In diesem Fall beispielsweise "&2137" und betätigen die Eingabe-Taste.



The screenshot shows the 'Anzeige' (Display) menu with the following options: 0 Ein/Ausgänge, 1 Istposition, 2 Zyklische Flags, 3 Flags, 4 Zähler, 5 Timer, 6 Variable, 7 Diagnose, and 8 Werkzeugdefinition. The '6 Variable' option is selected, and a sub-menu is open showing '0 Korrigieren' and '1 Anzeigen'. An arrow points from the '0 Korrigieren' option to a dialog box. The dialog box has a 'Name:' field containing '&2137', an 'Aktueller Wert:' field containing 'Name nicht als einfache Variable deklar', a 'Neuer Wert:' field, and a 'Modul:' field.

Wenn Sie nun die SRC-Datei (in diesem Falls "ERROR.SRC") in den Editor laden, können Sie die entsprechenden Korrekturen vornehmen. Beachten Sie, daß die begrenzte Sichtbarkeit ausgeschaltet (Menübefehl "Konfig." -> "Einstellungen" -> "Extras" -> "LimitedVisibility on/off") sowie die DEF-Zeile sichtbar ist (Menübefehl "Konfig." -> "Einstellungen" -> "Extras" -> "DEF-Zeile"). Näheres finden Sie in Abschnitt 2.4.

Im vorliegenden Beispiel brauchen die Folds nicht geöffnet zu werden. Wird dies gewünscht, verwenden Sie den Menübefehl "Bearbeiten" -> "Folds" -> "alle FOLDS öffnen".



Als Hilfe werden bei bestimmten Fehlern drei Sterne "***" am Anfang der fehlerhaften Zeile eingefügt.

```

1  DEF  ERROR ( )
2  INT I
3  INI
4  PTP HOME  Ue1= 100 % DEFAULT
5
6  FOR I=1 TO 4
7  $OUT[I]=TRUE
8  ENDFOR
9
10 ***I = I + 1
11
12 PTP HOME  Ue1= 100 % DEFAULT
13 END

```

Diese Zeile hier einfügen

Ein Gleichheitszeichen entfernen



Die im ursprünglich erstellten Programm fehlende Zeile "INT I" muß **vor** der Zeile "INI" eingefügt werden. Dies ist nur möglich, wenn die Zeile "DEF ERROR ()" sichtbar ist.

Fügen Sie also die Zeile

INT I

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

I = I + 1

Nach Schließen des Editors wird die Fehlerdatei "ERROR.ERR" automatisch gelöscht. Unter Umständen muß die Dateiliste mit dem Menübefehl "Datei" -> "Aktualisieren" neu eingelesen werden.



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



NOTIZEN:

2.8 Kommentare

Kommentare sind ein 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.

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

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



NOTIZEN: