

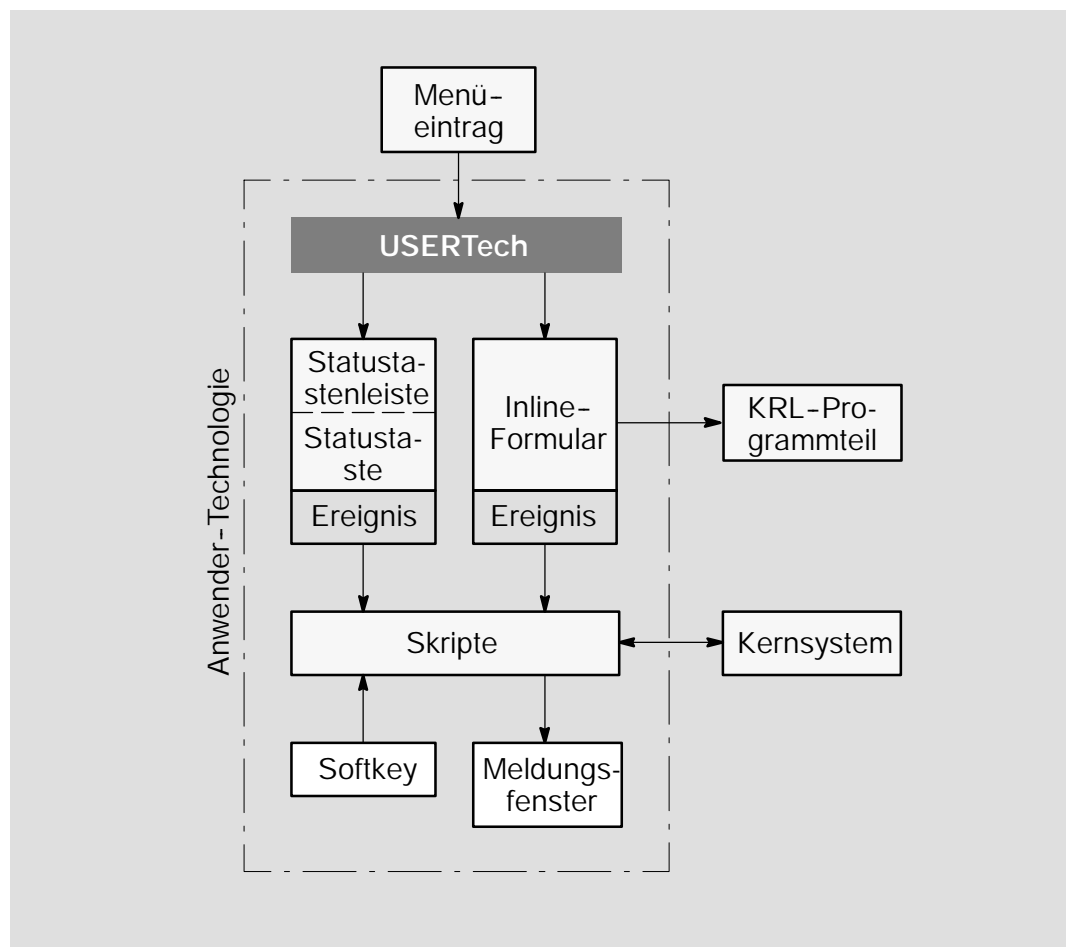


# 1 UserTech

## 1.1 Einführung

KUKA-Technologiepakete enthalten die wichtigsten Funktionen für übliche Roboteranwendungen. Spezielle Funktionen werden vom Anwender direkt in "KRL", der "KUKA-Robot Language" programmiert.

Um auch hier die Vorteile einer grafisch orientierten Bedienoberfläche nutzbar zu machen, wurde das Paket "USERTech" integriert. Mit Hilfe von Sprachelementen der "KUKA-Form Description Language", kurz "KFDL" genannt, ergänzen Sie auf einfache Weise Ihre selbst erstellten Technologiepakete um grafische Elemente.



Über Einträge im Menü werden Statustastenleisten oder Inline-Formulare aufgerufen. In die Eingabefelder der Inline-Formulare werden Daten eingegeben, die später im zu erzeugenden KRL-Programmteil Verwendung finden. Mit den Statustasten können Optionen ausgewählt werden, die den weiteren Programmablauf bestimmen. In Abhängigkeit von weiteren Bedienhandlungen, also "ereignisgesteuert", werden vom Anwender erstellte Programmteile ausgeführt, die mit dem Kernsystem Daten austauschen können.

## 1.2 Übersicht des Sprachumfangs

Technologie betreffend:	
DEFTP ... ENDTP	Kennzeichnet Beginn und Ende einer Technologie

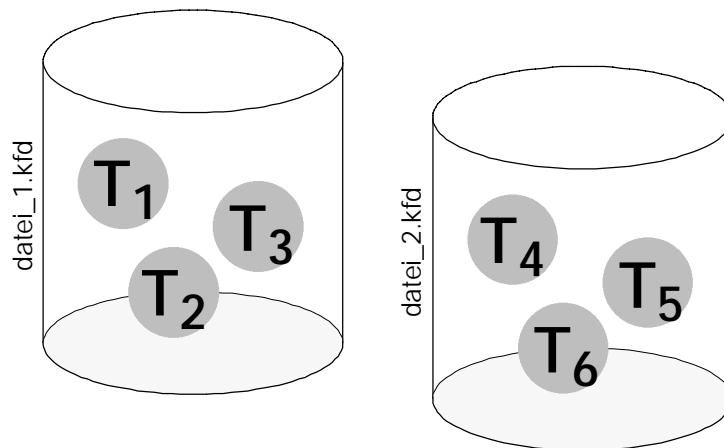
Inline-Formulare betreffend:	
DECL INLINEFORM	Beschreibung eines Inline-Formulares
DECL PARAM	Definiert Eingabefelder eines Inline-Formulares
DECL FOLD	Beschreiben eines KRL-Folds

Statustasten betreffend:	
DECL STATKEYBAR	Deklaration einer Statuskeyleiste
DECL STATKEY	Deklaration eines Statuskeys
SET	Neubelegung einer bereits vorhandenen Statuskeyleiste, bzw. bereits vorhandener Statuskeys

Skripte betreffend:	
DEFSCRIPT ... ENDSCRIPT	Kennzeichnet Beginn und Ende eines Skriptes
CASE, CASE ELSE	Kennzeichnet eine Verzweigung in der Kontrollstruktur SWITCH...ENDSWITCH
DO	Befehl zum Ausführen eines Skriptes
MESSAGE	Ausgabe von Text im Meldungsfenster
REDECL	Legt eine KRL-Variable im Kernsystem an oder überschreibt sie
SETVAR	Schreiben eines Wertes in eine KRL-Variable im Kernsystem
SHOWVAR	Auslesen des Wertes einer KRL-Variablen im Kernsystem
SWITCH ... ENDSWITCH	Kennzeichnet Beginn und Ende einer Kontrollstruktur
SWITCH DIALOG ... ENDSWITCH	Kennzeichnet Beginn und Ende einer Kontrollstruktur, die über das Meldungsfenster und die Softkeyleiste mit dem Benutzer kommuniziert

## 1.3 Anwender-Technologien

Anwender-Technologien werden in Dateien mit der Endung **KFD** beschrieben. Zu ihrer Erstellung kann jeder beliebige Text-Editor benutzt werden. Der Name der Datei kann innerhalb der Konventionen des Betriebssystems frei gewählt werden, er tritt im Technologiepaket selbst nicht im Erscheinung. KFD-Dateien werden im Verzeichnis `c:\programme\krc\template` gespeichert. USERTech hat keinen Zugriff auf den Inhalt von KFD-Dateien, die an anderer Stelle gespeichert sind. Jede KFD-Datei kann eine oder auch mehrere Technologien beinhalten. Ein Technologienname darf aber insgesamt nur einmal vorkommen. Informationen darüber, wie Technologien in das Gesamtsystem integriert werden, finden Sie in Kapitel 1.4.



### 1.3.1 Technologien beschreiben

Die Beschreibung einer Anwendertechnologie muß innerhalb der **KFD**-Datei durch die Anweisung:

**DEFTP** *Name*={ (SOC *Bool*, ) (SOT *Bool*) }

eingeleitet und durch die Anweisung:

**ENDTP**

abgeschlossen werden.



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

Innerhalb dieser Anweisungen eingeführte Objekte und Variablen gelten lokal, d.h. nur innerhalb der Technologie. Sollen Objekte und Variablen global, d.h. in jeder Technologie der **KFD**-Datei gültig sein, müssen sie außerhalb der Anweisungen ( **DEFTP**, **ENDTP** ) eingeführt werden.

Global definierte Inlineformulare erscheinen als eigene Einträge im Listenfeld.

Werden keine zusätzlichen Parameter angegeben, so gelten die fettgedruckten Grundeinstellungen:

SOT	<b>TRUE</b>	Auswahl der Technologien über Listenfeld möglich
	<b>FALSE</b>	Auswahl der Technologien <u>nicht</u> über Listenfeld möglich
SOC	<b>TRUE</b>	Auswahl der Befehle über Listenfeld möglich
	<b>FALSE</b>	Auswahl der Befehle <u>nicht</u> über Listenfeld möglich



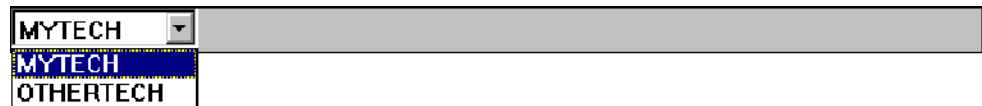
### Beispiel 1

Die Anweisungen:

```
DEFTP MyTech
ENDTP
```

```
DEFTP OtherTech
ENDTP
```

erzeugen demnach dieses Inline-Formular:



Zwischen den Technologien kann mit dem Statuskey +/- (rechts unten im Display) umgeschaltet werden:



### Beispiel 2

Die Anweisungen:

```
DEFTP MyTech={SOT FALSE}
ENDTP
```

```
DEFTP OtherTech
ENDTP
```

erzeugen dieses Inline-Formular:



Zwischen den Technologien kann nicht umgeschaltet werden. Sie müßten einzeln über Menüeinträge in der Datei **menuKUKA.INI** im Verzeichnis **C:\PROGRAMME\KRC\LIB** integriert werden. ( Siehe auch Kapitel 1.4 )

### 1.3.2 Inline-Formular erzeugen

Das Inline-Formular ist eine Eingabemaske für Parameter, die von UserTech für die Erzeugung von KRL-Programmcode benötigt werden.

Die Beschreibung eines Inline-Formulars hat folgendes Format:

```
DECL INLINEFORM Name=
    { (FOCUS Int, )
      (FOLD[ 1] Name, ) ( . . . , ) (FOLD[ n] Name, )
      (PARAM[ 1] Name, ) ( . . . , ) (PARAM[ n] Name)
      (STYLE WYSIWYG| SUB| DSUB| FCT| DFCT| ASS| ASSAGG, )
      (ONACCEPT Name, )
      (ONTOUCHUP Name) }
```



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

#### FOCUS

Nummer des Eingabefeldes ( siehe 1.3.2.1 ), auf welchem ( beim erstmaligen Erscheinen des Inline-Formulares ) der Eingabefokus liegt.

#### FOLD

Zugeordnete(r) KRL-Fold(s). Folds müssen definiert sein, bevor sie benutzt werden können. Zur Definition von Folds siehe 1.3.2.2.

Ordnen Sie dem Inlineformular keinen Fold zu, so wird sein Inhalt entsprechend der Einstellung in STYLE formatiert in das KRL-Programm eingesetzt.

#### PARAM

Eingabefelder im Inline-Formular. Eingabefelder müssen definiert sein, bevor sie benutzt werden können. Zur Definition von Eingabefeldern siehe 1.3.2.1.

#### STYLE

Die hier angegebenen Optionen beeinflussen das Aussehen des eingefügten KRL-Programmteiles wie folgt:

##### WYSIWYG

Die Grundeinstellung. Die einzufügende KRL-Programmzeile wird so formatiert, daß sie exakt dem Text des Inline-Formulares gleicht.

**LASER** **ON** Schweissdatensatz=**DataSet1**, Distance=**200** mm,  
WeavePattern=**PULSE**

resultierender KRL-Programmtext:

```
LASER.ON Schweissdatensatz=DataSet1, Distance=200mm,
WeavePattern=PULSE
```

**SUB (Veraltet: UP)**

Die Formatierung erfolgt in Form eines KRL-Unterprogrammaufrufes. Die Parameter werden auf den Inhalt der Eingabefelder reduziert, durch Kommata getrennt und von Klammern eingeschlossen. Der Trennpunkt zwischen Technologiename und Inline-Formularname wird unterdrückt.

**LASER** . **ON** (Schweissdatensatz=**DataSet1** , Distance=**200** mm, WeavePattern=**PULSE** )

resultierender KRL-Programmtext:

```
LASERON(DataSet1,200,PULSE)
```

**DSUB (Veraltet: DUP)**

Die Formatierung erfolgt in Form eines KRL-Unterprogrammaufrufes. Die Parameter werden auf den Inhalt der Eingabefelder reduziert, durch Kommata getrennt und von Klammern eingeschlossen. Der Trennpunkt zwischen Technologiename und Inline-Formularname wird unterdrückt.

Zusätzlich wird eine Parameterbeschreibung als Kommentar angehängt.

**LASER** . **ON** (Schweissdatensatz=**DataSet1** , Distance=**200** mm, WeavePattern=**PULSE** )

resultierender KRL-Programmtext:

```
LASERON(DataSet1,200,PULSE) ;Schweissdatensatz,Dis  
WeavePattern
```

**FCT**

Die Formatierung erfolgt in Form eines KRL-Funktionsaufrufes. Der Wert des ersten Feldes ( PARAM[1] ) wird unabhängig von seinem Format als Festtext dargestellt und als Funktionsname benutzt. Die anderen Parameter werden auf den Inhalt der Eingabefelder reduziert, durch Kommata getrennt und von Klammern eingeschlossen. Der Trennpunkt zwischen Technologiename und Inline-Formularname wird unterdrückt. Zwischen Formularname und Funktionsname wird ein Gleichheitszeichen eingefügt.

**LASER** . **ON** =DataSet1( Distance=**200** mm, WeavePattern=**PULSE** )

resultierender KRL-Programmtext:

```
LASERON=DataSet1(200,PULSE)
```

**DFCT**

Die Formatierung erfolgt in Form eines KRL-Funktionsaufrufes. Der Wert des ersten Feldes ( PARAM[1] ) wird unabhängig von seinem Format als Festtext dargestellt und als Funktionsname benutzt. Die anderen Parameter werden auf den Inhalt der Eingabefelder reduziert, durch Kommata getrennt und von Klammern eingeschlossen. Der Trennpunkt zwischen Technologiename und Inline-Formularname wird unterdrückt. Zwischen Formularname und Funktionsname wird ein Gleichheitszeichen eingefügt.

Zusätzlich wird eine Parameterbeschreibung als Kommentar angehängt.

**LASER** . **ON** =DataSet1( Distance=**200** mm, WeavePattern=**PULSE** )

resultierender KRL-Programmtext:

```
LASERON=DataSet1(200,PULSE) ;Distance[mm],WeavePattern
```

**ASS**

Die Formatierung erfolgt in Form einer Zuweisung. Die Parameter werden auf den Inhalt der Eingabefelder reduziert und durch Kommata getrennt. Zwischen Formularename und Parameterliste wird ein Gleichheitszeichen gesetzt. Der Trennpunkt zwischen Technologiename und Inline-Formularename wird unterdrückt.



resultierender KRL-Programmtext:

**LASERON=DataSet1.200.PULSE**

**ASSAGG**

Die Formatierung erfolgt in Form einer Aggregatzuweisung. Die Parameter werden auf den Inhalt der Eingabefelder mit dem vorangestellten Text, der unter **SHORTNAME[ ]** angegeben wurde, reduziert, durch Kommata getrennt und von geschweiften Klammern eingeschlossen. Zwischen dem Namen des Inline-Formulars und dem Parameterblock wird ein Gleichheitszeichen eingefügt.

**ONACCEPT**

Ein Ereignis des Inline-Formulars. Hier kann der Name des Skriptes angegeben werden, das nach dem Betätigen der "Enter"-Taste, bzw. des "Befehl Ok"-Softkeys ausgeführt werden soll.

**ONTOUCHUP**

Ein Ereignis des Inline-Formulars. Hier kann der Name des Skriptes angegeben werden, das nach dem Betätigen des "TouchUp"-Softkeys ausgeführt wird.

**1.3.2.1 Eingabe-, Anzeige- und Listenfelder**

Eingabe-, Anzeige- und Listenfelder in Inlineformularen werden wie folgt beschrieben:

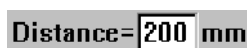
```
DECL PARAM Name={ (SHORTNAME[ ] "String",)
                  (SHORTCUT[ ] "String",)
                  (UNIT[ ] "String",)
                  (ENABLE Bool,)
                  (USERMODE Int,)
VALUE FieldDescription }
```



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

**SHORTNAME**

Hier angegebener Text erscheint vor dem Eingabefeld. Im Beispiel: "Distance=".





### SHORTCUT

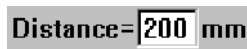
Hier angegebener Text erscheint auf dem +/- Statuskey ( rechts unten im Display ).  
Im Beispiel: "DIST".



Wird hier kein Text angegeben, erscheint der unter SHORTNAME angegebene Text auf dem Statuskey.

### UNIT

Hier angegebener Text erscheint hinter dem Eingabefeld. Im Beispiel: "mm".



### ENABLE

Über diese Eigenschaft wird das Feld aktiviert oder deaktiviert.

### USERMODE

Kennzahl der Benutzerebene, ab welcher das Feld aktiviert wird. Nähere Informationen zum Thema Benutzerebenen finden Sie in der Dokumentation [System konfigurieren].

Als Grundeinstellung gilt der Wert 0.

### VALUE

Beschreibt den Typ des Eingabefeldes. Siehe Abschnitt 1.7.1, "Field Description".

#### 1.3.2.2 FOLD-Beschreibung

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

USERTech ist in der Lage, solche "FOLDS", entsprechend Ihren Vorgaben, selbstständig zu generieren. Dazu muß das Gerüst des FOLDS, in das Ihre Daten eingetragen werden sollen, wie folgt beschrieben werden:

```
DECL FOLD      Name[ n]
                Name[ 1]="String"
                . . .
                Name[ n]="String"
```



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

Jedes Feldelement stellt eine Zeile im Inhalt des künftigen FOLD dar. Alle Platzhalter (%Name) in der Zeichenkette "String" werden durch ihre im Inline-Formular eingegebenen, bzw. aktuellen Werte ersetzt. Die Parameter Bauteilprogrammname (%MODULE), Technologiename (%TP) und Inline-Formularname (%INLINEFORM) können bei der Generierung des KRL-Folds benutzt werden.



### Beispiel

```
DECL FOLD MyFol d[ 2]
    MyFol d[ 1]="Laser( #%I NLI NEFORM , %DataSet , %Pattern )"
    MyFol d[ 2]="TRI GGER WHEN DI STANCE=%Di stanceWay DELAY=0 DO _
    LASER_ON=TRUE"
```

Ergebnis:

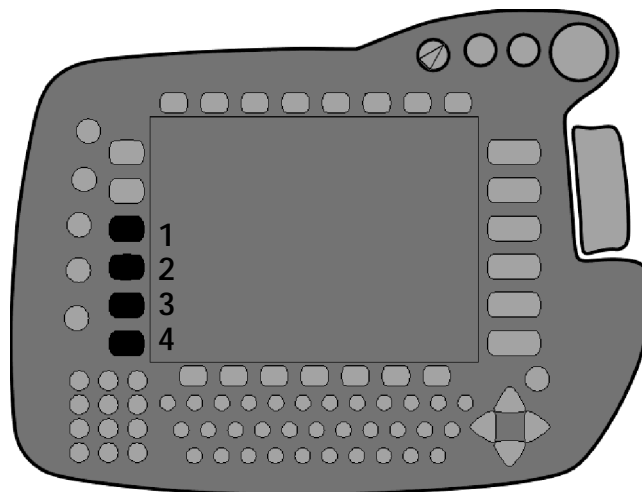
(wenn DataSet="DataSet6"; DistanceWay=210; Pattern="STEP"; Inlineform="ON" )

Laser( #ON, DatSet6, STEP)

TRI GGER WHEN DI STANCE=210 DELAY=0 DO LASER\_ON=TRUE

### 1.3.3 Statuskeys erzeugen

UserTech ermöglicht die freie Belegung von vier Statustasten.



Diese erfolgt in mehreren Schritten:

- G Beschreibung der einzelnen Statustaste(n)
- G Beschreibung der gesamten Statustastenleiste
- G Programmieren der Skipte, die mit Bedienhandlungen an den Statustasten verknüpft werden

Die Anweisung zum Erzeugen einer Statuskey-Leiste hat folgendes Format:

```
DECL STATKEYBAR Name= { (STATKEY[ 1] Name, )
                           (STATKEY[ 2] Name, )
                           (STATKEY[ 3] Name, )
                           (STATKEY[ 4] Name) }
```



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

### STATKEY [n]

Name des Statuskeys, der an Stelle [n] der Statuskey-Leiste angezeigt werden soll.

Bereits deklarierte Statustastenleisten können mit der Anweisung:

```
SET Name=      { (STATKEY[ 1] Name, )  
                  (STATKEY[ 2] Name, )  
                  (STATKEY[ 3] Name, )  
                  (STATKEY[ 4] Name, ) }
```

neu belegt werden.

Die Anweisung zum Erzeugen eines Statuskeys hat folgendes Format:

```
DECL STATKEY Name= { (TOPTTEXT[ ] String, )
                        (CENTERTEXT[ ] String, )
                        (BOTTOMTEXT[ ] String, )
                        (PICTURE[ ] String, )
                        (KEYDOWN_PICTURE[ ] String, )
                        (KEYDOWNMINUS_PICTURE[ ] String, )
                        (ENABLE Bool, )
                        (NEED_SAFETYSWITCH Bool, )
                        (NEED_DRIVESOK Bool, )
                        (NEED_PROSTATEO Int, )
                        (NEED_PROSTATE Int, )
                        (NEED_MODEOP Int, )
                        (USERMODE Int, )
                        (STYLE #SWITCH| #TOGGLE, )
                        (ONKEYDOWN Name, )
                        (ONKEYUP Name, )
                        (ONKEYSHOW Name, )
                        (ONKEYDOWNMINUS Name, )
                        (ONKEYUPMINUS Name, )
                        (ONKEYREPEAT Name, )
                        (ONKEYREPEATMINUS Name, )
                        (NEXT Name) }
```



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

## TOPTTEXT

Hier angegebener Text erscheint seitenzentriert im oberen Bereich des Statuskeys.



### CENTERTEXT

Hier angegebener Text erscheint seitenzentriert im mittleren Bereich des Statuskeys.



### BOTTOMTEXT

Hier angegebener Text erscheint seitenzentriert im unteren Bereich des Statuskeys.



### PICTURE

Pfadangabe der Bitmap-Abbildung<sup>1)</sup>, die auf dem Statuskey angezeigt werden soll.

Für den einfachen Softkey wird ein 34 × 48 Pixel großes Bitmap benötigt, für den doppelten Softkey muß das Bitmap 34 × 128 Pixel groß sein.

Zum Erstellen der Abbildungen können Sie MS-Paint verwenden, das im Lieferumfang des Betriebssystems Windows 95 enthalten ist.

### KEYDOWN\_PICTURE

Pfadangabe der Bitmap-Abbildung<sup>1)</sup>, die auf dem gedrückten "Plus"-Statuskey angezeigt werden soll.

Für den einfachen Softkey wird ein 34 × 48 Pixel großes Bitmap benötigt, für den doppelten Softkey muß das Bitmap 34 × 128 Pixel groß sein.

Zum Erstellen der Abbildungen können Sie MS-Paint verwenden, das im Lieferumfang des Betriebssystems Windows 95 enthalten ist.

Wird für KEYDOWN\_PICTURE kein Bitmap angegeben, wird das unter PICTURE angegebene Bitmap angezeigt.

### KEYDOWNMINUS\_PICTURE

Pfadangabe der Bitmap-Abbildung<sup>1)</sup>, die auf dem gedrückten "Minus"-Statuskey angezeigt werden soll.

Für den einfachen Softkey wird ein 34 × 48 Pixel großes Bitmap benötigt, für den doppelten Softkey muß das Bitmap 34 × 128 Pixel groß sein.

Zum Erstellen der Abbildungen können Sie MS-Paint verwenden, das im Lieferumfang des Betriebssystems Windows 95 enthalten ist.

Wird für KEYDOWNMINUS\_PICTURE kein Bitmap angegeben, wird das unter PICTURE angegebene Bitmap angezeigt.

### ENABLE

Über diese Eigenschaft wird der Statuskey aktiviert oder deaktiviert. Dies geschieht auch in Abhängigkeit von den Werten der Parameter NEED\_SAFETYSWITCH, NEED\_MODEOP, NEED\_DRIVESOK, NEED\_PROSTATE0 und NEED\_PROSTATE.

<sup>1)</sup> Es sollten besser Icons ( \*.ico ) verwendet werden, um keine unerwünschten Effekte beim Abblenden einer Statustaste zu erzielen.

**NEED\_SAFETYSWITCH**

Mit diesem Parameter wird die Bedienbarkeit des Statuskeys von der Betätigung der Zustimmungstaste (unten am KCP) abhängig.

Als Grundeinstellung gilt der Wert FALSE.

**NEED\_DRIVESOK**

Mit diesem Parameter wird die Bedienbarkeit des Statuskeys vom Einschalten der Antriebe abhängig.

Als Grundeinstellung gilt der Wert FALSE.

**NEED\_MODEOP**

Mit diesem Parameter wird die Bedienbarkeit des Statuskeys von der angewählten Betriebsart abhängig.

Der ganzzahlige Wert stellt eine Bitfolge dar:

Betriebsart	Extern	Automatik	T2 - Betrieb	T1 - Betrieb
Bit-Nummer	3	2	1	0
Binärer Wert (Standard)	0	1	1	1
Dezimaler Wert	3			

Der Dezimalwert wird ermittelt, indem man die Bitnummer zur Basis 2 potenziert.

Im Beispiel:

Der Wert 3 bewirkt, daß der Statuskey in den Betriebsarten T1 und T2 bedienbar ist.

Als Grundeinstellung gilt der Wert 7; der Statuskey ist dann nur in den Betriebsarten T1, T2 und Automatik verfügbar.

**NEED\_PROSTATE0**

Mit diesem Parameter wird die Bedienbarkeit des Statuskeys abhängig vom Betriebszustand des "Submit-Interpreters".

Der ganzzahlige Wert stellt eine Bitfolge dar:

Betriebsart	ACTIVE	END	RESET	STOP	FREE	UNKNOWN
Bit-Nummer	5	4	3	2	1	0
Binärer Wert (Standard)	1	0	0	0	0	0
Dezimaler Wert	32					

Der Dezimalwert wird ermittelt, indem man die Bitnummer zur Basis 2 potenziert.

Im Beispiel:

Der Wert 32 bewirkt, daß der Statuskey nur bei laufendem "Submit-Interpreter" verfügbar ist.

Als Grundeinstellung gilt der Wert 32.

## NEED\_PROSTATE

Mit diesem Parameter wird die Bedienbarkeit des Statuskeys abhängig vom Betriebszustand des "Roboter-Interpreters".

Der ganzzahlige Wert stellt eine Bitfolge dar:

Betriebsart	ACTIVE	END	RESET	STOP	FREE	UNKNOWN
Bit-Nummer	5	4	3	2	1	0
Binärer Wert (Standard)	0	1	1	1	1	0
Dezimaler Wert	30					

Der Dezimalwert wird ermittelt, indem man die Bitnummer zur Basis 2 potenziert.

Im Beispiel:

Der Wert 30 bewirkt, daß der Statuskey nur bei gelöster Starttaste verfügbar ist.

Als Grundeinstellung gilt der Wert 30.

## USERMODE

Kennzahl der Benutzerebene, ab welcher der Statuskey bedienbar wird. Nähere Informationen zum Thema Benutzerebenen finden Sie in der Dokumentation [System konfigurieren].

Als Grundeinstellung gilt der Wert 0. Dies hat zur Folge, daß der Statuskey in jeder Benutzerebene bedienbar ist.

## STYLE

Die hier angegebenen Optionen beeinflussen das Aussehen des Statuskeys.

### #SWITCH

Die Grundeinstellung. Dem Statuskey ist die links danebenliegende KCP-Taste zugeordnet.

### #TOGGLE

Der Statuskey wird als Doppeltaste dargestellt. Die obere KCP-Taste wird als "Plus"-Taste betrachtet, die untere KCP-Taste als "Minus"-Taste.

## ONKEYDOWN

Ein Ereignis der Statuskeys. Hier kann der Name des Skriptes angegeben werden, das nach dem Betätigen der "Plus"-Taste ausgeführt werden soll.

## ONKEYUP

Ein Ereignis der Statuskeys. Hier kann der Name des Skriptes angegeben werden, das nach dem Loslassen der "Plus"-Taste ausgeführt werden soll.

## ONKEYSHOW

Ein Ereignis der Statuskeys. Hier kann der Name des Skriptes angegeben werden, das nach dem Erscheinen des Statuskeys ausgeführt werden soll.

**ONKEYDOWNMINUS**

Ein Ereignis der Statuskeys. Hier kann der Name des Skriptes angegeben werden, das nach dem Betätigen der "Minus"-Taste ausgeführt werden soll.

**ONKEYUPMINUS**

Ein Ereignis der Statuskeys. Hier kann der Name des Skriptes angegeben werden, das nach dem Loslassen der "Minus"-Taste ausgeführt werden soll.

**ONKEYREPEAT**

Ein Ereignis der Statuskeys. Hier kann der Name des Skriptes angegeben werden, der nach längerer Betätigung der "Plus"-Taste ausgeführt werden soll. Dieses Ereignis wird bis zum Loslassen der Taste wiederholt ausgelöst. Die Zeitabstände werden dabei immer geringer.

**ONKEYREPEATMINUS**

Ein Ereignis der Statuskeys. Hier kann der Name des Skriptes angegeben werden, der nach längerer Betätigung der "Minus"-Taste ausgeführt werden soll. Dieses Ereignis wird bis zum Loslassen der Taste wiederholt ausgelöst. Die Zeitabstände werden dabei immer geringer.

**NEXT**

Name des nachfolgenden Statuskeys.

Bereits deklarierte Statustasten können mit der Anweisung:

```
SET Name=          { (TOPTTEXT[ ] String, )  
                      (CENTERTEXT[ ] String, )  
                      (BOTTOMTEXT[ ] String, )  
                      (PICTURE[ ] String, )  
                      (KEYDOWN_PICTURE[ ] String, )  
                      (KEYDOWNMINUS_PICTURE[ ] String, )  
                      (ENABLE Bool, )  
                      (NEED_SAFETYSWITCH Bool, )  
                      (NEED_DRIVESOK Bool, )  
                      (NEED_PROSTATE0 Int, )  
                      (NEED_PROSTATE Int, )  
                      (NEED_MODEOP Int, )  
                      (USERMODE Int, )
```



```
(STYLE #SWITCH| #TOGGLE, )
(ONKEYDOWN Name, )
(ONKEYUP Name, )
(ONKEYSHOW Name, )
(ONKEYDOWNMINUS Name, )
(ONKEYUPMINUS Name, )
(ONKEYREPEAT Name, )
(ONKEYREPEATMINUS Name, )
(NEXT Name) }
```

neu belegt werden.

Diese Anweisung kann innerhalb und außerhalb von Skripten verwendet werden. Bei der Anwendung innerhalb von Skripten können dann auch Platzhalter (z.B. %INLINEFORM) verwendet werden, die ihre Wertzuweisung bereits vor der Ausführung des Skriptes erhielten.

### 1.3.4 Skripte beschreiben

Skripte sind Programme mit speziellen Funktionen zur Kommunikation mit dem Kernsystem.

Die Beschreibung eines Skriptes muß innerhalb der KFD-Datei durch die Anweisung:

**DEFSCRIPT** *Name*

eingeleitet und durch die Anweisung

**ENDSCRIPT**

abgeschlossen werden.



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

Skripte können nicht geschachtelt werden. Es können nur die nachfolgend aufgelisteten Schlüsselwörter zur Anwendung kommen:

#### SETVAR

Setzen oder Beschreiben einer KRL-Variablen im Kernsystem

Syntax: **SETVAR** (**FULLPATH**[ ] "*String*", **VALUE**[ ] "*String*")

#### **FULLPATH**

Variablenname - Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die daraus resultierende Zeichenkette wird als Variablenname mit Pfadangabe interpretiert.

z.B. c: \programme\krc\poweron\r1\a10. dat \a\_counter

#### VALUE

Wertzuweisung - Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die daraus resultierende Zeichenkette wird als Zielwert interpretiert.

Existiert die angegebene Variable nicht oder ist der Zielwert mit dem Variablentyp unverträglich, so wird das Skript abgebrochen und eine Fehlermeldung im Meldungsfenster ausgegeben.

### SHOWVAR

Auslesen einer KRL-Variablen im Kernsystem

Syntax: **SHOWVAR (FULLPATH[] "String", PARAM Name)**

#### FULLPATH

Variablenname - Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die daraus resultierende Zeichenkette wird als Variablenname mit Pfadangabe interpretiert.

z.B. c: \programme\krc\poweron\r1\a10. dat \a\_counter

#### PARAM

Name des Parameters, in den der unter Angabe von FULLPATH gefundene Wert geschrieben wird.

Existiert die angegebene Variable nicht oder ist der Zielwert mit dem Variablentyp unverträglich, so wird das Skript abgebrochen und eine Fehlermeldung im Meldungsfenster ausgegeben.

### REDECL

Anlegen, bzw. Überschreiben einer KRL-Variablen im Kernsystem

Syntax: **REDECL (PATH[] "String", DECLARATION[] "String")**

#### PATH

Variablenname - Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die daraus resultierende Zeichenkette wird als Variablenname mit Pfadangabe interpretiert.

z.B. c: \programme\krc\poweron\r1\a10. dat \a\_counter

**HINWEIS** Um eine Variable im aktuellen Bauteilprogramm anzulegen, gibt es den vordefinierten Parameter %MODULE

#### DECLARATION

Wertzuweisung - Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die daraus resultierende Zeichenkette wird als Zielwert interpretiert.

Existiert die angegebene Variable bereits, ist aber von einem anderen Typ, bzw. ist die Deklarationszeile fehlerhaft oder unverträglich mit dem Typ des Parameters, so wird das Skript abgebrochen und eine Fehlermeldung im Meldungsfenster ausgegeben.

**DO**

Aufrufen eines Skriptes (Unterprogrammes)

Syntax: **DO** *Name*

Tritt während der Ausführung dieses Skriptes ein Fehler auf, so wird auch das aufrufende Skript abgebrochen.

**SET**

Undefinieren bereits deklarierter Statuskeys und Statustastenleisten.

Syntax: **SET** *Name*= { (TOPTEXT[ ] *String*, )  
 (CENTERTEXT[ ] *String*, )  
 (BOTTOMTEXT[ ] *String*, )  
 (PICTURE[ ] *String*, )  
 (KEYDOWN\_PICTURE[ ] *String*, )  
 (KEYDOWNMINUS\_PICTURE[ ] *String*, )  
 (ENABLE *Bool*, )  
 (NEED\_SAFETYSWITCH *Bool*, )  
 (NEED\_DRIVESOK *Bool*, )  
 (NEED\_PROSTATE0 *Int*, )  
 (NEED\_PROSTATE *Int*, )  
 (NEED\_MODEOP *Int*, )  
 (USERMODE *Int*, )  
 (STYLE #SWITCH|#TOGGLE, )  
 (ONKEYDOWN *Name*, )  
 (ONKEYUP *Name*, )  
 (ONKEYSHOW *Name*, )  
 (ONKEYDOWNMINUS *Name*, )  
 (ONKEYUPMINUS *Name*, )  
 (ONKEYREPEAT *Name*, )  
 (ONKEYREPEATMINUS *Name*, )  
 (NEXT *Name*) }

**MESSAGE**

Ausgabe eines einzeiligen Textes im Meldungsfenster

Syntax: **MESSAGE** *"String"*

Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die Zeichenkette wird dann als Schlüssel für die Sprach-Datenbank herangezogen. Für den erfolgreichen Zugriff auf diese Datenbank muß dort ein Modul mit dem Namen der Technologie vorhanden sein, in der das Skript definiert wurde. Bei globalen Skripten wird auf das Modul **KUKATPUSERG1** oba1 zugegriffen. Scheitert der Zugriff auf die Sprach-Datenbank, so wird der Schlüssel selbst im Meldungsfenster ausgegeben. In der Spalte "Absender" wird der Name der Technologie ausgegeben, in der das ausführende Skript enthalten ist.

Bei parametrisierten Ausgaben müssen die Parameter, getrennt durch das Zeichen "|", an den Schlüssel gehängt werden.

**SWITCH...CASE(ELSE)...ENDSWITCH**

Kontrollstruktur zum bedingten Aufruf von Skripten. Hinter dem Schlüsselwort SWITCH wird der Name einer Variablen angegeben, die beim Eintritt in die Kontrollstruktur abgefragt wird. Abhängig vom Ergebnis dieser Abfrage wird in einen der vorbereiteten Zweige gesprungen. Wird kein vorbereiteter Zweig gefunden, wird in den CASE ELSE-Zweig gesprungen. Eine Schachtelung von SWITCH...CASE(ELSE)...ENDSWITCH-Blöcken ist nicht möglich.

Syntax: **SWITCH** *"AbfrageString"*

**CASE** *"ErgebnisString"* DO *Name*

**CASE ELSE** DO *Name*

**ENDSWITCH**

**AbfrageString**

Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die daraus resultierende Zeichenkette wird als Variablenname mit Pfadangabe interpretiert.

z.B. c: \programme\krc\poweron\r1\ a10. dat \a\_counter

**ErgebnisString**

Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die daraus resultierende Zeichenkette wird als Variablenname mit Pfadangabe interpretiert.

z.B. c: \programme\krc\poweron\r1\ a10. dat \a\_counter

Tritt in der Abarbeitung ein Fehler auf, so wird auch das aufrufende Skript abgebrochen.

### SWITCH DIALOG...CASE...ENDSWITCH

Struktur zum Erzeugen eines Dialoges über das Meldungsfenster und die Softkeys. Innerhalb dieser Struktur müssen zwischen zwei und sieben Zweige programmiert sein, die den Softkeys in der unteren Leiste von rechts nach links entsprechen. In folgender Darstellung ist der Softkey 1 und 7 belegt, und Softkey 2 bis 5 leer.

```
Syntax:  SWITCH DIALOG "FrageString"
          CASE "AntwortString" D0 Name
          CASE
          CASE
          CASE
          CASE
          CASE
          CASE "AntwortString" D0 Name
          ENDSWITCH
```

#### FrageString

Alle Platzhalter (z.B. %PLATZHALTER) in der Zeichenkette werden durch ihre aktuellen Werte ersetzt. Die Zeichenkette wird dann als Schlüssel für die Sprach-Datenbank herangezogen. Scheitert der Zugriff auf die Sprach-Datenbank, so wird der Schlüssel selbst im Meldungsfenster ausgegeben.

#### AntwortString

Beschriftung des entsprechenden Softkeys. Die Zeichenkette wird dann als Schlüssel für die Sprach-Datenbank herangezogen. Scheitert der Zugriff auf die Sprach-Datenbank, so wird der Schlüssel selbst im Meldungsfenster ausgegeben. Um einen leeren Softkey zu erzeugen, lassen Sie einfach diese Angabe weg.

Für den erfolgreichen Zugriff auf diese Datenbank muß dort ein Modul mit dem Namen der Technologie vorhanden sein, in der das Skript definiert wurde. Bei globalen Skripten wird auf das Modul KUKATPUSER-Global zugegriffen.

In der Spalte "Absender" wird der Name der Technologie ausgegeben, in der das ausführende Skript enthalten ist.

Tritt in der Abarbeitung ein Fehler auf, so wird auch das aufrufende Skript abgebrochen.

Da sich mit Hilfe der SWITCH-Anweisung auch Schleifen realisieren lassen, achten Sie bitte darauf, Endlosschleifen zu vermeiden.

#### 1.3.4.1 Vordefinierte Skripte

##### ACCEPTINLINEFORM

Dieses Skript schließt das geöffnete Inline-Formular, übernimmt die geänderten Parameter und fügt den KRL-Programmteil ein, bzw. ersetzt diesen.

##### CANCELINLINEFORM

Dieses Skript schließt das geöffnete Inline-Formular, übernimmt die geänderten Parameter nicht und fügt auch keinen KRL-Programmteil ein.

##### END

Dieses Skript dient zum Abbruch, bzw. Beenden eines laufenden Skriptes.

**NOTHING**

Dieses Skript ist leer und hebt die ursprüngliche Zuweisung eines Skriptes auf.

## 1.4 Technologien integrieren

Die Integration von Anwender-Technologien erfolgt über Einträge in der Datei **menuKU-KA.INI** im Verzeichnis **c:\programme\krc\lib**.

### Untermenü

Um ein Untermenü in der Menüleiste zu erzeugen, arbeiten Sie bitte nacheinander die folgenden Schritte ab:

- G Ergänzen Sie den Dateiabchnitt [SOFTKEYS] um Ihren Eintrag im Format:

***Untermenüname = Beschriftung, , , POPUP, Liste\_der\_Einträge***

- G Definieren Sie die Eintragsliste im Dateiabchnitt [MENU]:

***Liste\_der\_Einträge=Erste\_Option, Zweite\_Option, Dritte\_Option***

- G Tragen Sie die Menüoptionen im Dateiabchnitt [SOFTKEYS] ein:

- a) Inline-Formular

Um eine Inline-Formular durch diesen Untermenüeintrag aufzurufen

***Erste\_Option = Beschriftung, 11, TECHPACK, KUKATPUSER; \_  
Technologie; Inl i ne- Formul arname***

- b) Statustastenleiste

Um eine Statustastenleiste durch diesen Untermenüeintrag aufzurufen

***Zweite\_Option = Beschriftung, 11, USERSTATKEYBAROCX, \_  
KUKATPUSER; Technol ogi ename. Statustastenlei stenname***

- c) Statustaste

Um eine Statustaste durch diesen Untermenüeintrag aufzurufen

***Dritte\_Option = Beschriftung, 11, USERSTATKEYBAROCX, \_  
KUKATPUSER; Technol ogi ename. Statustastennamen; Posi ti on***

- G Fügen Sie Ihr selbsterstelltes Menü dem "Technologie"-Menü zu. Suchen Sie zu diesem Zweck im Dateiabchnitt [MENU] den Eintrag **mTECHNOLOGIE** und hängen den Namen Ihres Menüs an:

***mTECHNOLOGIE = ARCTech, A20Tech, SPOTTech, . . . , \_  
Untermenüname***

Nach dem Speichern der geänderten Datei und einem Neustart des Systems steht der neue Untermenüpunkt zur Verfügung.

### 1.4.1 Ein Beispiel zur Integration



Im Menü "Technologie" soll ein neuer Untermenüpunkt mit der Bezeichnung "Untermenü" erscheinen. Nach dessen Auswahl soll ein Untermenü erscheinen, bei dem aus den Optionen "Eins", "Zwei" und "Drei" ausgewählt werden kann.

Nach Öffnen der Datei `c:\programme\krc\lib\menuKUKA.ini` wird das Ende des Abschnitts `[SOFTKEYS]` gesucht. Hier werden folgende Einträge gemacht:

```
erster_eintrag = Eins, 2010, TECHPACK, KUKATPUSER; MYTECH
zweiter_eintrag = Zwei, 2010, TECHPACK, KUKATPUSER; MYTECH
dritter_eintrag = Drei, 2010, TECHPACK, KUKATPUSER; MYTECH
neues_untermenue = Untermenue, , , POPUP, liste_der_eintraege
```

Am Anfang des Abschnittes `[MENU]` wird folgender Eintrag gemacht:

```
liste_der_eintraege = erster_eintrag, zweiter_eintrag, dritter_eintrag
```

Anschließend wird der Eintrag `mTECHNOLOGIE` gesucht und der Name des neuen Menüs angehängt:

```
mTECHNOLOGIE = ARCTech, A20Tech, SPOTTech, ..., neues_untermenue
```

Nach dem Speichern der veränderten Datei und einen Neustart des Systems steht der neue Untermenüpunkt zur Verfügung.





## 1.5 Programmierbeispiel

### 1.5.1 Statuskeys



Zweck dieses Programmes soll sein, Helligkeit und Kontrast des KCP-Displays über Statustasten einstellen zu können. Dazu müssen die Systemvariablen \$PHGBRIGHT und \$PHGCONT verändert werden.

Zuerst muß eine Technologie-Datei erstellt werden, die dann im Verzeichnis c: \programme\krc\template gespeichert wird. Als Dateiname wird DISP\_SET. KFD gewählt.

Die Technologie selbst soll ebenfalls DISP\_SET genannt werden:

```
DEFTP disp_set  
ENDTP
```

Im nächsten Schritt wird beschrieben, aus welchen Statustasten die Statustastenleiste bestehen soll:

```
DEFTP disp_set  
DECL STATKEYBAR leiste = { STATKEY[1] helligkeit,  
STATKEY[3] kontrast }  
  
ENDTP
```

Die Statustasten selbst müssen bereits vor der Statustastenleiste beschrieben werden:

```
DEFTP disp_set  
DECL STATKEY helligkeit = { PICTURE[] "c:\programme\krc\ _  
template\phgbright.bmp", _  
ENABLE TRUE, _  
USERMODE 10, _  
STYLE #TOGGLE, _  
ONKEYDOWN bright_hi, _  
ONKEYDOWNMINUS bright_lo}  
  
DECL STATKEY kontrast = { PICTURE[] "c:\programme\krc\ _  
template\phgcont.bmp", _  
ENABLE TRUE, _  
USERMODE 10, _  
STYLE #TOGGLE, _  
ONKEYDOWN cont_hi, _  
ONKEYDOWNMINUS cont_lo}  
  
DECL STATKEYBAR leiste = { STATKEY[1] helligkeit,  
STATKEY[3] kontrast }  
  
ENDTP
```

Vor den Beschreibungen der Statustasten und der Statustastenleiste muß die Beschreibung der Skripte erfolgen, die durch die Ereignisse ONKEYDOWN, bzw. ONKEYDOWNMINUS aufgerufen werden:

```
DEFTP disp_set  
DEFSCRIPT bright_hi  
SHOWAR(FULLPATH[] "$PHGBRIGHT", PARAM bright)  
SETVAR(FULLPATH[] "$PHGBRIGHT", VALUE[] "%bright +1")  
SHOWAR(FULLPATH[] "$PHGBRIGHT", PARAM bright)
```

```

SHOWAR(FULLPATH[] "$PHGCONT", PARAM cont)
SHOWAR(FULLPATH[] "$PHGTEMP", PARAM temp)
MESSAGE "Helligkeitsstufe: %bright Kontraststufe: _
%cont Display-Temperatur: %temp °C"
ENDSCRIPT
DEFSCRIPT bright_lo
SHOWAR(FULLPATH[] "$PHGBRIGHT", PARAM bright)
SETVAR(FULLPATH[] "$PHGBRIGHT", VALUE[] "%bright -1")
SHOWAR(FULLPATH[] "$PHGBRIGHT", PARAM bright)
SHOWAR(FULLPATH[] "$PHGCONT", PARAM cont)
SHOWAR(FULLPATH[] "$PHGTEMP", PARAM temp)
MESSAGE "Helligkeitsstufe: %bright Kontraststufe: _
%cont Display-Temperatur: %temp °C"
ENDSCRIPT
DEFSCRIPT cont_hi
SHOWAR(FULLPATH[] "$PHGCONT", PARAM cont)
SETVAR(FULLPATH[] "$PHGCONT", VALUE[] "%cont +1")
SHOWAR(FULLPATH[] "$PHGCONT", PARAM cont)
SHOWAR(FULLPATH[] "$PHGBRIGHT", PARAM bright)
SHOWAR(FULLPATH[] "$PHGTEMP", PARAM temp)
MESSAGE "Helligkeitsstufe: %bright Kontraststufe: _
%cont Display-Temperatur: %temp °C"
ENDSCRIPT
DEFSCRIPT cont_lo
SHOWAR(FULLPATH[] "$PHGCONT", PARAM cont)
SETVAR(FULLPATH[] "$PHGCONT", VALUE[] "%cont -1")
SHOWAR(FULLPATH[] "$PHGCONT", PARAM cont)
SHOWAR(FULLPATH[] "$PHGBRIGHT", PARAM bright)
SHOWAR(FULLPATH[] "$PHGTEMP", PARAM temp)
MESSAGE "Helligkeitsstufe: %bright Kontraststufe: _
%cont Display-Temperatur: %temp °C"
ENDSCRIPT
DECL STATKEY helligkeit = { PICTURE[] "c:\programme\krc\ _
template\phgbright.bmp", _
ENABLE TRUE, _
USERMODE 10, _
STYLE #TOGGLE, _
ONKEYDOWN bright_hi, _
ONKEYDOWNMI NUS bright_lo}
DECL STATKEY kontrast = { PICTURE[] "c:\programme\krc\ _
template\phgcont.bmp", _
ENABLE TRUE, _
USERMODE 10, _
STYLE #TOGGLE, _
ONKEYDOWN cont_hi, _
ONKEYDOWNMI NUS cont_lo}

```

```
DECL STATKEYBAR leiste = {  STATKEY[1] helligkeit,
                           STATKEY[3] kontrast      }

ENDTP
```

Zum Abschluß werden noch die in den Skripten verwendeten Parameter vor den Skripten beschrieben:

```
DEFTP di sp_set
DECL PARAM bright = {VALUE {NUMBER: }}
DECL PARAM cont = {VALUE {NUMBER: }}
DECL PARAM temp = {VALUE {NUMBER: }}
DEFSRIPT bright_hi
  SHOWVAR(FULLPATH[] "SPHGBRIGHT", PARAM bright)
  SETVAR(FULLPATH[] "SPHGBRIGHT", VALUE[] "%bright +1")
  SHOWVAR(FULLPATH[] "SPHGBRIGHT", PARAM bright)
  SHOWVAR(FULLPATH[] "SPHGCONT", PARAM cont)
  SHOWVAR(FULLPATH[] "SPHGTEMP", PARAM temp)
  MESSAGE "Helligkeitsstufe: %bright Kontraststufe: _
%cont Display-Temperatur: %temp °C"
ENDSCRIPT
DEFSRIPT bright_lo
  SHOWVAR(FULLPATH[] "SPHGBRIGHT", PARAM bright)
  SETVAR(FULLPATH[] "SPHGBRIGHT", VALUE[] "%bright -1")
  SHOWVAR(FULLPATH[] "SPHGBRIGHT", PARAM bright)
  SHOWVAR(FULLPATH[] "SPHGCONT", PARAM cont)
  SHOWVAR(FULLPATH[] "SPHGTEMP", PARAM temp)
  MESSAGE "Helligkeitsstufe: %bright Kontraststufe: _
%cont Display-Temperatur: %temp °C"
ENDSCRIPT
DEFSRIPT cont_hi
  SHOWVAR(FULLPATH[] "SPHGCONT", PARAM cont)
  SETVAR(FULLPATH[] "SPHGCONT", VALUE[] "%cont +1")
  SHOWVAR(FULLPATH[] "SPHGCONT", PARAM cont)
  SHOWVAR(FULLPATH[] "SPHGBRIGHT", PARAM bright)
  SHOWVAR(FULLPATH[] "SPHGTEMP", PARAM temp)
  MESSAGE "Helligkeitsstufe: %bright Kontraststufe: _
%cont Display-Temperatur: %temp °C"
ENDSCRIPT
DEFSRIPT cont_lo
  SHOWVAR(FULLPATH[] "SPHGCONT", PARAM cont)
  SETVAR(FULLPATH[] "SPHGCONT", VALUE[] "%cont -1")
  SHOWVAR(FULLPATH[] "SPHGCONT", PARAM cont)
  SHOWVAR(FULLPATH[] "SPHGBRIGHT", PARAM bright)
  SHOWVAR(FULLPATH[] "SPHGTEMP", PARAM temp)
```

```

MESSAGE "Helligkeitsstufe: %bright Kontraststufe: _
%cont Display-Temperatur: %temp °C"

ENDSCRIPT

DECL STATKEY helligkeit = { PICTURE[] "c:\programme\krc\ _
template\phgbright.bmp", _
ENABLE TRUE, _
USERMODE 10, _
STYLE #TOGGLE, _
ONKEYDOWN bright_hi, _
ONKEYDOWNMINUS bright_lo}

DECL STATKEY kontrast = { PICTURE[] "c:\programme\krc\ _
template\phgcont.bmp", _
ENABLE TRUE, _
USERMODE 10, _
STYLE #TOGGLE, _
ONKEYDOWN cont_hi, _
ONKEYDOWNMINUS cont_lo}

DECL STATKEYBAR leiste = { STATKEY[1] helligkeit,
STATKEY[3] kontrast }

ENDTP

```

In der Datei `menuKUKA.ini` im Verzeichnis `c:\programme\krc\lib` müssen dann noch folgende Ergänzungen vorgenommen werden:

```

...
;TechnologieMenu
...
DISPSET = Display-Einstellungen, 11, USERSTATKEYBAROCX, KUKATPU-
SER;disp_set.leiste; 1

...
mTECHSTATUSKEYS= H50STATKEYS, A10STATKEYS, A20STATKEYS, DISPSET
...

```

Nach einem erneuten Systemhochlauf steht dieses Programm zur Verfügung:



## 1.5.2 Inline-Formular



## Programmtext

DEFTP LASER

```
DECL PARAM DataSet={SHORTNAME[] "Schweisdatensatz=", _
VALUE {NAME: DEFAULT[] "DataSet1"}, _
SHORTCUT[] "DATA"}
```

```
DECL PARAM DistanceWay={ SHORTNAME[] "Distance=", _
VALUE {NUMBER: DEFAULT 200, MIN 0, MAX 500, _
STEP 10}, _
SHORTCUT[] "DIST", UNIT[] "mm"}
```

```
DECL PARAM DelayTime={ SHORTNAME[] "Delay=", _
VALUE {REAL: DEFAULT 0.8, MIN 0, MAX 20.3, _
STEP 0.3}, _
UNIT[] "ms", SHORTCUT[] "DLY"}
```

```
DECL PARAM Pattern={ SHORTNAME[] "WeavePattern=", _
VALUE {LIST: _
ITEM[1] {ITEM: VALUE[] "PULSE"}, _
ITEM[2] {ITEM: VALUE[] "STEP"}, _
ITEM[3] {ITEM: VALUE[] "CONT"} _
}, SHORTCUT[] "WPTN"}
```

DECL FOLD LasOn[2]

```
LasOn[1]="Laser(#ON,%DataSet,%Pattern)"
```

```
LasOn[2]="TRI GGER WHEN DI STANCE=%Di stanceWay _
DELAY=0 DO LASER_ON=TRUE"
```

DECL FOLD LasOff[2]

```
LasOff[1]="Laser(#OFF)"
```

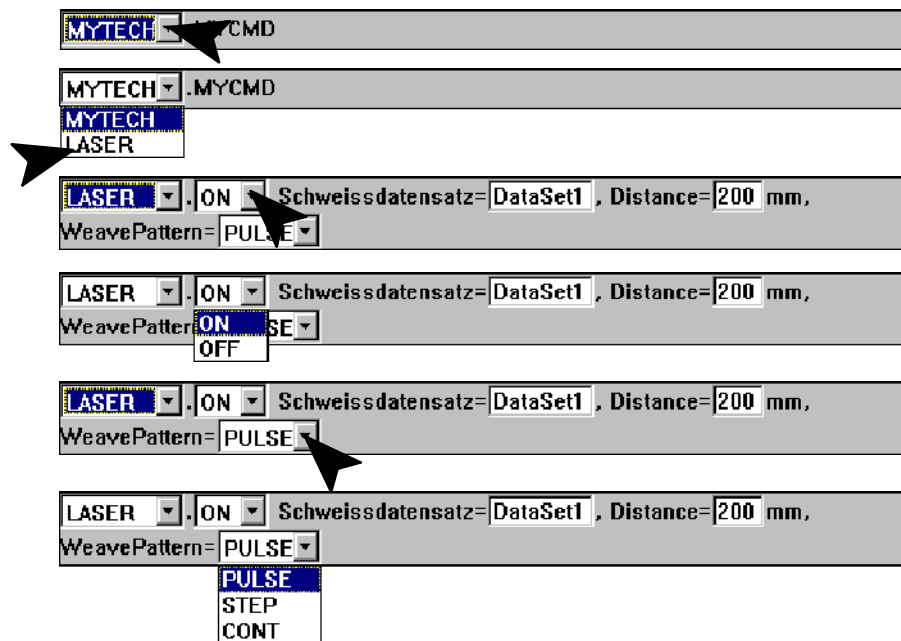
```
LasOff[2]="TRI GGER WHEN DI STANCE=0 _
DELAY=%Del ayTi me _
DO LASER_ON=FALSE"
```

```
DECL InlineForm On={PARAM[1] DataSet,PARAM[2] _
DistanceWay, PARAM[3] Pattern, FOLD[1] LasOn}
```

```
DECL InlineForm Off={PARAM[1] DelayTime, FOLD[1] LasOff}
```

ENDTP

## Erscheinungsbild

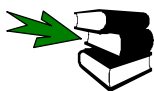


## 1.6 UserTech reinitialisieren

Änderungen und Ergänzungen an bereits initialisierten Technologiepaketen bedingen keinen erneuten Systemstart. Über die Funktion "UserTech reinitialisieren" des Menüpunktes "Konfigurieren" können Sie die geänderten Daten neu einlesen lassen.



Diese Funktion kann nur dann ausgeführt werden, wenn Sie sich in der Benutzergruppe "Experte" befinden.



Wie Sie in diese Benutzergruppe wechseln können, ist in der Dokumentation [System konfigurieren] beschrieben.

## 1.7 Anhang

### 1.7.1 Gültige Ausdrücke

In KFDL-Ausdrücken werden folgende Typen benutzt:

#### Bool

Dieser Typ beschreibt einen binären Zustand, kann also nur zwei Werte annehmen. In KFDL ( und KRL ) werden diese Zustände mit **TRUE** oder **FALSE** bezeichnet.

#### Int

Dieser Typ beschreibt eine Ganzzahl.

#### Real

Dieser Typ beschreibt einen Fließkommawert.

Beispiele für syntaktisch korrekte Ausdrücke:

3. 12

1. 2

#### Char

Dieser Typ beschreibt ein einzelnes alphanumerisches Zeichen.

#### Name

Dieser Typ beschreibt einen gültigen Namen für eine in KRL zu verwendete Variable.

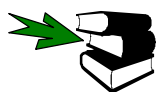
Die Länge dieses Ausdrucks liegt zwischen 1 und 11 Zeichen.

Das erste Zeichen muß Element der Gruppen *a...z* oder *A...Z* sein. Das Zeichen "\$" kann auch verwendet werden. Alle folgenden Zeichen müssen Element der Gruppen *a...z*, *A...Z* oder *0...9* sein. Die Zeichen "\$" und "\_" können ebenfalls verwendet werden.

Die erste Zahl von rechts kann während der Laufzeit mit dem Statuskey +/- verändert werden. Dies ist bei Punktbezeichnungen hilfreich.



Die Verwendung reservierter KRL-Schlüsselwörter ist nicht zulässig und führt zu einer Fehlermeldung.



Eine Liste reservierter KRL-Schlüsselwörter finden Sie im [KRL Reference Guide].

Beispiele für syntaktisch korrekte Ausdrücke:

"A"

"Hallo"

"D47g11"

## String

Dieser Typ beschreibt eine Kette von alphanumerischen Zeichen.



Nachfolgend aufgelistete Zeichen führen bei ihrer Verwendung zu einer Fehlermeldung oder Fehlfunktion.

"," – Komma

Beispiele für syntaktisch korrekte Ausdrücke:

"A"

"Hallo"

"D47g11"

"1aBc"

" "

## FieldDescription

### Der Typ {static: }

Die korrekte Syntax lautet:

**{static: default[] "String"}**



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

#### Beispiel

```
decl param field_sta = {value _
{static: default[] "This can't be changed"}}}
```

erzeugt dieses Ausgabefeld im Inline-Formular:

**This can't be changed**

Der ausgegebene Text kann nicht geändert werden.

### Der Typ {free: }

Die korrekte Syntax lautet:

**{free: (default[] "String")}**



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

#### Beispiel

```
decl param field_fre = {shortname[] "Programmer: ", _
{free: default[] " Alfred E. Neumann "}}}
```

erzeugt dieses Eingabefeld im Inline-Formular:

**Programmer:** **Alfred E. Neumann**

Der vorgegebene Text kann geändert werden. Das Vorgeben eines Textes ist optional.



**Der Typ {name: }**

Die korrekte Syntax lautet:

**{name: default[] "Name"}**



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

**Beispiel**

```
decl param field_nam = { shortname[] "Welding-point-nr. : ", _
                        shortcut[] "WPT", value _
                        {name: default[] "WPT1"}} }
```

erzeugt dieses Eingabefeld im Inline-Formular:

**Welding-point-nr.:**

Der vorgegebene KRL-Variablenname kann geändert werden.



Der Statuskey "+/-" trägt dabei die mit "shortcut[]" festgelegte Beschriftung "WPT". Durch Drücken dieses Statuskey kann die erste Zahl von rechts schrittweise erhöht bzw. verringert werden. Dies ist bei Punktbezeichnungen hilfreich.

**Der Typ {number: }**

Die korrekte Syntax lautet:

**{number: (min Int,) (max Int,) (step Int,) (default Int,) (autolimit Bool)}**



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

**AUTOLIMIT**

Setzt einen falsch eingegebenen Wert automatisch auf den minimalen, bzw. maximalen Wert. Der Defaultwert ist TRUE.

**Beispiel**

```
decl param field_num = { shortname[] "Distance: ", _
                        shortcut[] "DIST", _
                        unit[] "mm", _
                        value _
                        {number: min 0, max 100, step 2, default 50,}} }
```

erzeugt dieses Nummernfeld im Inline-Formular:

**Distance:**  mm



Der Statuskey "+/-" trägt dabei die mit "shortcut[]" festgelegte Beschriftung "NUM". In diesem Nummernfeld kann ein Wert zwischen 0 und 100 eingegeben werden (als Standardwert erscheint 50). Durch Drücken des Statuskey "+/-" wird dieser Wert um jeweils 2 erhöht bzw. verringert.

**Der Typ {real: }**

Die korrekte Syntax lautet:

**{real: (min Real,) (max Real,) (step Real,) (default Real,) (autolimit Bool)}**



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

## AUTOLIMIT

Setzt einen falsch eingegebenen Wert automatisch auf den minimalen, bzw. maximalen Wert. Der Defaultwert ist TRUE.

### Beispiel

```
decl param field_rea = { shortname[] "Delay: ", _
                        shortcut[] "DELAY", _
                        unit[] "secs", _
                        value _
{real: min 0.5, max 5, step 0.5, default 2}}
```

erzeugt dieses Eingabefeld im Inline-Formular:

Delay:  secs



Der Statuskey "+/-" trägt dabei die mit "shortcut[]" festgelegte Beschriftung "DELAY". In diesem Feld kann ein Wert zwischen 0.5 und 5 eingegeben werden (als Standardwert erscheint 5). Durch Drücken des Statuskey "+/-" wird dieser Wert um jeweils 0,5 verringert bzw. erhöht.

## Der Typ {list:}

Die korrekte Syntax lautet:

```
{list: (default[] "String",) _
      (pos Int,) _
      item[1] ItemDescription, _
      (item[2] ItemDescription,)
      (...,)
      (item[n] ItemDescription)}
```



Die Angabe fettgedruckter Passagen ist obligatorisch. In runde Klammern gefaßte Ausdrücke sind optional. Kursiv gedruckte Ausdrücke müssen ersetzt werden.

*ItemDescription* (Listeneintrag)

Die korrekte Syntax lautet:

```
{item value[] "String" (, disp[] "String")}
```

Mit dem Parameter "disp" können Sie eine Unterscheidung zwischen angezeigtem und tatsächlich verarbeitetem Text treffen. Sie können also einen anderen als den angezeigten Text verarbeiten.

### Beispiel

```
decl param field_lis = { shortname[] "Weave pattern: ", _
                        shortcut[] "PATT", value _
                        {list: pos 1, _
                        item[1] _
                        {item value[] "Triangle"}, _
                        item[2] _
                        {item value[] "Trapezoid"}, _
                        item[3] _
                        {item value[] "Sinus"}}}
```

erzeugt dieses Listenfeld im Inline-Formular:

Weave pattern:

- Triangle
- Trapezoid
- Sinus



Der Statuskey "+/-" trägt dabei die mit "short cut [ ]" festgelegte Beschriftung "PATT". In diesem Listenelement kann eine der vorgegebenen Optionen ausgewählt werden (als Standardwert erscheint das erste Listenelement "Triangle"). Durch Drücken des Statuskey "+/-" wird zwischen den vorgegebenen Elementen des Listenelementes umgeschaltet.

### 1.7.2 Sonderzeichen

";" kennzeichnet den Rest der Programmzeile als Kommentar.

z. B.

; Dies ist ein Kommentar

"\_" führt die Programmzeile trotz der Unterbrechung durch eine neue Zeile fort.

z. B.

decl \_

int \_

zahl

entspricht:

decl int zahl

"/" hebt die Sonderfunktion eines nachfolgenden Zeichens auf. Das Zeichen selbst wird in der resultierenden Zeichenkette unterdrückt.

z. B.

/%

um das Zeichen "%" anzugeben, das eine Sonderfunktion hat.

"%" kennzeichnet einen Platzhalter.

z. B.

%I NLI NEFORM