OrangeApps

myHMI

Benutzerdefinierte HMI per XML

Version 1.2

Anwender Dokumentation

Stand: 07.07.2021, Version 1.7

OrangeApps GmbH
Arnikaweg 1
87471 Durach
Deutschland
www.orangeapps.de

Gültigkeit der Dokumentation

Dokument Version	Software Version		Verfasser	Datum
	Von	Bis		
1.7	1.2		Mayer	27.01.2021

Historie der Dokumentenversionen

Version	Datum	Autor	Änderungsgrund / Bemerkung
1.0	22.09.2016	Christian Mayer	Ersterstellung
1.1	20.2.2018	Christian Mayer	Implementation Steuerelement „Picture“
1.2	06.03.2018	Christian Mayer	Kleinere Überarbeitungen der Dokumentation
1.3	13.3.2018	Christian Mayer	Kapitel 5.5 überarbeitet
1.4.	23.11.2018	Christian Mayer	AreYouSure Element bei Button entfernt
1.5.	26.9.2019	Christian Mayer	Kap. 7.2 myHMI-Designer eingefügt
1.6	20.8.2020	Christian Mayer	Kap.8 KRL-Funktionen eingefügt
1.7	07.07.2021	Christian Mayer	Steuerelement Switch IO, Installation als KOP über WoV

Inhalt

1 Einleitung

1.1 Zielgruppe

Diese Dokumentation richtet sich an Anwender mit folgenden Kenntnissen:

- Kenntnisse über die Software-Struktur des KUKA Robotersystems

1.2 Darstellung von Hinweisen

Diese Hinweise bedeuten, dass Tod oder schwere Körperverletzungen sicher oder sehr wahrscheinlich eintreten werden, wenn keine Vorsichtsmaßnahmen getroffen werden.

Diese Hinweise bedeuten, dass Tod oder schwere Körperverletzungen eintreten **können**, wenn keine Vorsichtsmaßnahmen getroffen werden.

Diese Hinweise bedeuten, dass leichte Körperverletzungen eintreten **können**, wenn keine Vorsichtsmaßnahmen getroffen werden.

Diese Hinweise bedeuten, dass Sachschäden eintreten **können**, wenn keine Vorsichtsmaßnahmen getroffen werden.

Diese Hinweise enthalten nützliche Tipps oder besondere Informationen für das aktuelle Thema.

1.3 Verwendete Begriffe

Begriff	Beschreibung
HMI	Die Human-Machine Interface (HMI) ist eine Schnittstelle, über die ein Mensch mit einer Maschine kommuniziert.
KSS	KUKA Systemsoftware
smartPad	Roboter Bediengerät
SmartHMI	Bedienoberfläche der KRC4 Robotersteuerung
KOP	KUKA Option Package
WoV	KUKA Work Visual

2 Übersicht

2.1 Produktbeschreibung

Mit myHMI können anwenderspezifische HMI‘s auf dem SmartPad angezeigt werden. Diese diesen zur Anzeige und Manipulation aller auf dem Robotersystem bekannten KRL-Variablen werden. Im Vordergrund steht die Anzeige von BOOL, INT, REAL, ENUM und CHAR Variablen unter Verwendung von grafischen Elementen wie Textfelder, Schalter, LEDs, Dropdown-Steuerelemente und Bildsteuerelementen.

Die Erstellung einer HMI erfolgt über eine XML-Datei in der mittels eines normalen Texteditors die spezifischen Einträge erstellt werden. Ein Plugin interpretiert die Einträge in der XML-Datei und stellt die Elemente in tabellarischer Form auf einer HMI dar. Der Anwender benötigt somit keinerlei Kenntnisse in der Programmierung von Plugins und HMI‘s.

Es können beliebig viele XML-Dateien und somit beliebig viele HMI’s erzeugt werden. Der Aufruf jeder einzelnen HMI erfolgt über einen Menüeintrag im Hauptmenü des Roboters. Dabei kann zwischen einer halb- oder vollseitigen Anzeige gewählt werden. Die HMI bettet sich nahtlos in das Robotersystem ein. Alle standardmäßigen Menü- und Bedienelemente bleiben vollständig bedienbar. Zur einfachen Erstellung der Menüeinträge steht ein komfortabler Menüassistent zur Verfügung.

Um eine thematische Trennung innerhalb einer HMI zu ermöglichen, können die Inhalte auf maximal 5 Registerkarten verteilt werden. Jede Registerkarte kann 32 Elemente darstellen.

Jedes Element kann dynamisch nach angemeldeter Benutzerebene dargestellt (sichtbar) bzw. editierbar (schreibgeschützt) sein.

Die gesamte HMI unterstützt Mehrsprachigkeit, sodass eine dynamische Sprachumschaltung problemlos möglich ist.

Durch den Bediener geänderte Werte, werden im KUKA Logbuch aufgezeichnet. (Diagnose/Logbuch).

Zur Erstellung der Menüaufrufe der einzelnen HMI’s steht ein Menüassistent zur Verfügung. Somit sind keinerlei Kenntnisse zur Menüerstellung notwendig.

2.2 Merkmale

- HMI zur Anzeige und Manipulation von KRL Variablen
- Angezeigte Inhalte werden per XML definiert
- Inhalte werden thematisch mit Registerkarten dargestellt
- Die Steuerelemente werden untereinander in tabellarischer Form mit bis zu 3 Spalten dargestellt
- Steuerelemente: Schalter, Taster, Textfeld, Nummernfeld, LED, Dropdown-Liste, Schieberegler, Fortschrittsanzeige, Überschrift, Beschriftungsfeld und Bild
- Neues Steuerelement:
- Die Steuerelemente können durch Angabe optionaler Abhängigkeiten aktiviert oder deaktiviert werden (Benutzergruppe, Submit u. Programm-Status, Antriebe, Zustimmschalter, Betriebsart usw.)
- Dynamische Bildanzeige
- Bilddateien können sowohl lokal als auch auf einem Netzlaufwerk gespeichert sein

- Inhalte können unter Verwendung von KXR-Dateien mehrsprachig dargestellt werden
- Benutzereingaben werden im Logbuch gespeichert
- Die Anzahl der erstellten HMI ist theoretisch unbegrenzt
- Bis zu 5 Registerkarten können je HMI definiert werden
- Bis zu 32 Elemente können je Registerkarte definiert werden
- Jede HMI wird aus dem Hauptmenü geöffnet
- die Menüeinträge können mit einem Menüassistenten erstellt werden
- einfache Installation der Software über die Standard KUKA Installationsroutine
- **Neue Funktion ab V1.2: Öffnen und Schließen der HMI's direkt über KRL, automatischer Start beim Start des Controllers und abhängig von der Änderung des Benutzerlevels und der Betriebsart**
- **Installation über WorkVisual möglich**

2.3 Lieferumfang

Die Lieferung erfolgt als Technologiepaket zur direkten Installation am Roboter als Zusatzsoftware. Darin sind alle zur Installation und Betrieb notwendigen Komponenten enthalten:

- Plugin myHMI
- Anwender-Dokumentation zur Installation und Betrieb der Software
- Menü Assistent

Um dem Anwender den Einstieg zu erleichtern, ist im Setup-Paket eine Beispiel-HMI mit folgenden Dateien enthalten:

- DemoMyHMI.xml → enthält Beispiele zur Verwendung von Registerkarten und Steuerelementen
- DemoMyHMI.kxr → Beispiel zum Erstellen einer Sprachdatenbank für Mehrsprachigkeit

2.4 Einsatzgebiet / -umgebung

Die Software ist lauffähig auf allen KUKA Roboter mit KSS8.3.23 oder höher ohne CPC-Schutz.

Auskunft über den Einsatz der Software auf älteren Maschinen sind unter info@orangeapps.de erhältlich.

3 Installation, Deinstallation, Update

Ab der Version 1.2.5 wird die Software als KUKA Optionspaket (KOP) ausgeliefert.

Die Installation kann über Work Visual (WoV) oder über die Option *Zusatzsoftware* erfolgen. Die Option *Zusatzsoftware* befindet sich im Hauptmenü unter *Inbetriebnahme.*

Ältere Version als 1.2.5 müssen vor der Installation des KOP deinstalliert werden. Dabei müssen auch alle Menüeinträge zum Aufruf benutzerdefinierter HMI's über den Menü-Assistenten gelöscht werden.

3.1 Systemvoraussetzungen für die Ausführung

Mindestanforderungen Software

- KUKA System Software 8.3.23
- Bei Installation über WorkVisual: WorkVisual 5.x oder höher

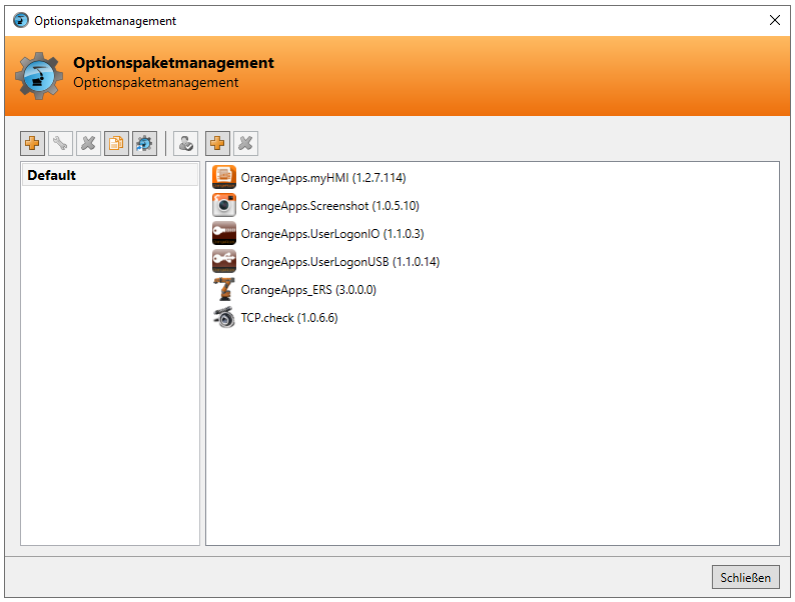
Soll die Technologie auf KRC4 Robotern mit KSS Version älter als 8.3.23 installiert werden, ist diese Version bei uns erhältlich. Sprechen Sie uns dazu an.

Falls auf dem Roboter KUKA.CPC verwendet wird, wird zur Installation des Plugin ein Softwarezertifikat benötigt.
Bitte halten Sie in diesem Fall, vor dem Kauf der Software, Rücksprache mit unserem Kundenservice (Email an info@orangeapps.de)

3.2 Installation über Work Visual

3.2.1 myHMI installieren oder updaten

Die Installation des KOP erfolgt wie ein normales KUKA Optionspaket und muss dazu über das Optionspaketmanagement in WoV installiert werden. Es steht dann als Katalogelement zur Verfügung.



In WoV wird dann das Optionspaket dem Projekt hinzugefügt und beim Übertragen des Projekts automatisch auf der Robotersteuerung installiert.

Bei einem Update muss die vorherige Version des Optionspakets in WoV zuerst deinstalliert werden. Vor einem Update sollten alle zugehörigen Daten archiviert werden.

Überblick Schritte Installation über WoV

- Optionspaket in WoV als Katalogelement installieren
- Projekt von Roboter ziehen
- Option einfügen
- Am Roboter als Experte anmelden und Projekt übertragen

Voraussetzung

- Mindestens Benutzergruppe Experte
- Betriebsart T1 oder T2
- Es ist kein Programm angewählt.
- Netzwerkverbindung zur Robotersteuerung
- Optionspaket liegt als KOP-Datei vor.

Vorgehensweise

1. **Nur bei einem Update**: Die vorherige Version des Optionspakets MyHMI in WorkVisual deinstallieren.
2. Das Optionspaket MyHMI in WorkVisual installieren.
3. Das Projekt von der Robotersteuerung laden.
4. Das Optionspaket MyHMI in das Projekt einfügen.
5. Das Projekt von WorkVisual auf die Robotersteuerung übertragen und aktivieren.
6. Auf der smartHMI wird die Sicherheitsabfrage ***Wollen Sie die Aktivierung des Projektes […] zulassen?*** angezeigt. Bei der Aktivierung wird das aktive Projekt überschrieben. Wenn kein relevantes Projekt überschrieben wird: Die Abfrage mit ***Ja*** bestätigen.
7. Auf der smartHMI wird eine Übersicht mit den Änderungen und einer Sicherheitsabfrage angezeigt. Diese mit ***Ja*** beantworten. Das Optionspaket wird installiert und die Robotersteuerung führt einen Neustart durch.

Informationen zu Abläufen in WorkVisual sind in der Dokumentation zu WorkVisual zu finden..

LOG-Datei

Es wird eine LOG-Datei unter C:\KRC\ROBOTER\LOG erstellt.

Eintrag im Hauptmenü

-

Eintrag im Infofenster

Nach erfolgreicher Installation wird unter ***Hilfe→Info→Optionen*** der Eintrag „OrangeApps.myHMI“ angezeigt.

Veränderte Robotersystemdateien

-

3.2.1.1 myHMI deinstallieren

Vor einer Deinstallation sollten alle zugehörigen Daten archiviert werden.

Überblick Schritte Deinstallation über WoV

- Projekt von Roboter ziehen
- Option entfernen

- Am Roboter als Experte anmelden und Projekt übertragen

Voraussetzung

- Mindestens Benutzergruppe Experte
- Betriebsart T1 oder T2
- Es ist kein Programm angewählt
- Netzwerkverbindung zur Robotersteuerung
- Optionspaket liegt als KOP-Datei vor.

Vorgehensweise

1. Das Projekt von der Robotersteuerung laden.
2. Das Optionspaket MyHMI in das Projekt einfügen.
3. Das Projekt von WorkVisual auf die Robotersteuerung übertragen und aktivieren.
4. Auf der smartHMI wird die Sicherheitsabfrage ***Wollen Sie die Aktivierung des Projektes […] zulassen?*** angezeigt. Bei der Aktivierung wird das aktive Projekt überschrieben. Wenn kein relevantes Projekt überschrieben wird: Die Abfrage mit ***Ja*** bestätigen.
5. Auf der smartHMI wird eine Übersicht mit den Änderungen und einer Sicherheitsabfrage angezeigt. Diese mit Ja beantworten. Das Optionspaket wird deinstalliert und die Robotersteuerung führt einen Neustart durch.

Informationen zu Abläufen in WorkVisual sind in der Dokumentation zu WorkVisual zu finden..

LOG-Datei

Es wird eine LOG-Datei unter C:\KRC\ROBOTER\LOG erstellt.

3.2.2 Installation über smartHMI

3.2.2.1 MyHMI installieren oder updaten

Voraussetzung

- Mindestens Benutzergruppe Experte
- Betriebsart T1 oder T2
- Kein Programm angewählt
- USB-Stick mit dem Optionspaket (KOP-Datei)
- KSS 8.3 oder höher

Vorgehensweise

Die Installation erfolgt über ***Inbetriebnahme → Zusatzsoftware*** im Hauptmenü.

1. Kopieren sie die KOP-Datei entweder auf einen USB-Stick oder direkt auf ein Laufwerk des Zielsystems (z.B. d:\).
2. Bei der Installation von einem USB-Stick, schließen sie diesen an den Steuerungs-PC oder das smartPad an.
3. Wählen Sie im Hauptmenü unter ***Inbetriebnahme*** → ***Zusatzsoftware*** aus.
4. Klicken Sie auf die Schaltfläche ***Neue Software***.
5. Sie erhalten eine Liste für die Installation zur Verfügung stehender Software. Sollte in der Liste kein Eintrag mit ***OrangeApps.MyHMI*** aufgeführt sein, klicken Sie auf ***Aktualisieren.*** Wird nun der Eintrag angezeigt, machen Sie weiter mit Schritt 8.
6. Sollte der Eintrag nicht angezeigt werden, muss das Laufwerk von dem installiert werden soll, erst konfiguriert werden. Wählen Sie dazu ***Konfiguration*****.** In einem neuen Fenster haben Sie nun die Möglichkeit den Pfad auswählen unter dem der die Option ***OrangeApps.MyHMI*** zu finden ist.
7. Markieren Sie im Bereich ***Installationspfade für Optionen*** eine leere Zelle und wählen Sie ***Pfadauswahl.*** Die vorhandenen Laufwerke werden angezeigt. Markieren Sie das Laufwerk an dem die Option ***OrangeApps.MyHMI*** zur Verfügung steht und speichern Sie Ihre Auswahl mit ***Speichern.*** Das Fenster schließt sich wieder. In der Liste sollte nun ein Eintrag ***OrangeApps.MyHMI*** erscheinen. Ist dies nicht der Fall, drücken Sie auf ***Aktualisieren*** und/oder wiederholen Sie die Schritte 7 und 8.
8. Markieren Sie den Eintrag ***OrangeApps.MyHMI*** und drücken Sie auf ***Installieren*****.** Bestätigen Sie den Installationshinweis mit ***OK***
9. Auf der smartHMI wird die Sicherheitsabfrage ***Wollen Sie die Aktivierung des Projektes […] zulassen?*** angezeigt. Bei der Aktivierung wird das aktive Projekt überschrieben. Wenn kein relevantes Projekt überschrieben wird: Die Abfrage mit ***Ja*** bestätigen.
10. Auf der smartHMI wird eine Übersicht mit den Änderungen und einer Sicherheitsabfrage angezeigt. Diese mit ***Ja*** beantworten. Das Optionspaket wird installiert und die Robotersteuerung führt einen Neustart durch.
11. Ziehen Sie gegebenenfalls den USB-Stick ab.

LOG-Datei

Es wird eine LOG-Datei unter C:\KRC\ROBOTER\LOG erstellt.

Eintrag im Hauptmenü

-

Eintrag im Infofenster

Nach erfolgreicher Installation wird unter ***Hilfe→Info→Optionen*** der Eintrag „OrangeApps.MyHMI“ angezeigt.

Veränderte Robotersystemdateien

-

3.2.2.2 MyHMI deinstallieren

Voraussetzung

- Mindestens Benutzergruppe Experte
- Betriebsart T1 oder T2
- Kein Programm angewählt

Vorgehensweise

Die Deinstallation erfolgt über ***Inbetriebnahme → Zusatzsoftware*** im Hauptmenü.

1. Wählen Sie im Hauptmenü unter ***Inbetriebnahme → Zusatzsoftware*** aus.
2. Setzen Sie ein Häkchen bei ***OrangeApps.MyHMI*** uund drücken Sie ***Deinstallieren.***
3. Auf der smartHMI wird die Sicherheitsabfrage ***Wollen Sie die Aktivierung des Projektes […] zulassen?*** angezeigt. Bei der Aktivierung wird das aktive Projekt überschrieben. Wenn kein relevantes Projekt überschrieben wird: Die Abfrage mit ***Ja*** bestätigen.
4. Auf der smartHMI wird eine Übersicht mit den Änderungen und einer Sicherheitsabfrage angezeigt. Diese mit ***Ja*** beantworten. Das Optionspaket wird deinstalliert und die Robotersteuerung führt einen Neustart durch.

LOG-Datei

Es wird eine LOG-Datei unter C:\KRC\ROBOTER\LOG erstellt.

3.3 Installierte Dateien

Zum Betrieb der Software werden folgende Dateien installiert:

Ordner	Dateien	Funktion
C:\KRC\TP\OrangeApps.myHMI\SmartHmi	OrangeApps.myHMI.dll SmartHMI.exe.myHMI.config OrangeApps.myHMIViewService.dll SmartHMI.exe.myHMIViewService.config OrangeApps.myHMIMenuConfigurator.dll SmartHMI.exe.myHMIMenuConfigurator.config	Plugin myHMI Plugin Menü Assistent
C:\KRC\TP\OrangeApps.myHMI\kxr	myHMI.kxr myHMIMenuConfigurator.kxr	Sprachdaten-bank für myHMI und Menü Assistent
R1\TP\myHMI	myHMI_user.dat myHMI_lib.src and .dat	Funktionen und Variablen zum Öffnen / Schließen von HMIs innerhalb von KRL, beim Hochlauf und abhängig von Benutzerebene und Betriebsart

Folgende Dateien werden für die Beispiel-HMI installiert:

Ordner	Dateien	Funktion
C:\KRC\DATA	DemoMyHMI.kxr	Sprachdatenbank für Demo HMI
C:\KRC\USER\myHMI	DemoMyHMI.xml myHMIDemo.src und .dat	HMI KRL Modul für Demo
C:\KRC\USER\myHMI\PicsDemo	verschiedene Bilddateien	

Die Beispiel HMI ist voll lauffähig und kann somit als Grundlage für weitere HMI's genutzt werden.

Im Auslieferungszustand ist kein Menüaufruf für die Demo-HMI vorhanden. Dieser muss über den Menüassistenten erzeugt werden.

4 Lizenzierung

myHMI ist für die produktive Nutzung lizenzierungspflichtig. Die Lizenzierung erfolgt über eine Lizenzdatei. Zu Testzwecken sind kostenlose Test-Lizenzen unter www.orangeapps.de erhältlich.

Hinweis

- Für jeden Roboter ist eine Lizenz notwendig.
- Testlizenzen sind kostenlos und zeitlich begrenzt.
- **Datumsmanipulationen** am System werden erkannt, myHMI deaktiviert die Lizenz automatisch

Testlizenzen können direkt auf www.orangeapps.de bezogen werden. Laufzeitlizenzen erhalten Sie nach Eingang der Lizenzgebühr.

4.1 Roboterlizenz

Um eine gültige Lizenz zu erhalten, benötigen Sie die Seriennummer des Roboters. Diese finden Sie auf dem Typenschild des Roboters oder in der Steuerungssoftware im Menü ***Hilfe→Info→Roboter→Seriennummer***.

4.2 Lizenz für KUKA OfficePC/OfficeLite

Für OfficePC / OfficeLite wird keine Lizenz benötigt.

4.1 Lizenz installieren

Methode 1

- Stecken Sie einen USB-Stick mit darauf gespeicherter Lizenz an einem USB Port der Steuerung ein.
- Beim Hochlauf der Steuerung oder dem Öffnen einer HMI wird bei Vorhandensein einer gültigen Lizenz auf dem USB-Stick diese automatisch in den Lizenzordner kopiert und aktiviert. **Hinweis:** Eine Laufzeitlizenz im Lizenzordner wird dabei nicht durch eine Testlizenz überschrieben
- Entfernen Sie den USB-Stick

Methode 2

- Kopieren Sie die erhaltene Lizenz in den Ordner C:\KRC\TP\OrangeApps.myHMI\lic.

5 Elemente einer HMI

Die darzustellende HMI wird vollständig in einer XML-Datei definiert.

Die XML-Datei muss im Verzeichnis C:\KRC\User\myHMI gespeichert werden.

Auf dem Robotersystem können beliebig viele HMI erstellt und angezeigt werden. Jede HMI benötigt dazu folgende Dateien (der Platzhalter *HMI_Name* steht dabei für den Namen der HMI)

- *HMI_Name*.xml
- (optional) *HMI_Name*.kxr

In der XML-Datei wird folgendes definiert:

- Anzahl und Reihenfolge der Registerkarten
- Anzahl und Reihenfolge der Steuerelemente
- Gruppierung der Steuerelemente
- Eigenschaften der Steuerelemente
- Überschrift des Anzeigefensters

In der kxr-Datei wird folgendes definiert:

- Übersetzungstexte

Beim Aufruf aus dem Hauptmenü wird die XML-Datei interpretiert und die HMI entsprechend dargestellt. Die Darstellung der Steuerelemente erfolgt spaltenweise auf Registerkarten, entsprechend der Reihenfolge in der XML-Datei. Je nach Bedarf kann eine ein-, zwei- oder dreispaltiger Darstellung gewählt werden.

Die (optionale) Verwendung einer kxr-Übersetzungsdatei ermöglicht eine automatische Übersetzung der dargestellten Texte in die eingestellte HMI-Sprache.

Die Menüeinträge werden in der Datei SmartHMI.User.config im Verzeichnis C:\KRC\User erstellt. Dazu steht der Menüassistent zur Verfügung.

Tabellarische Zusammenfassung

Datei	Beschreibung	Speicherort
HMI_Name.xml	Beschreibung der darzustellenden Inhalte über Registerkarten und vordefinierte Steuerelemente	C:\KRC\User\myHMI
HMI_Name.kxr	(Optional) für Mehrsprachigkeit der Text- und Beschriftungsinhalte	C:\KRC\Data

SmartHMI.User.config	Eintragung der Menüaufrufe. Erstellt durch den Menüassistenten.	C:\KRC\User

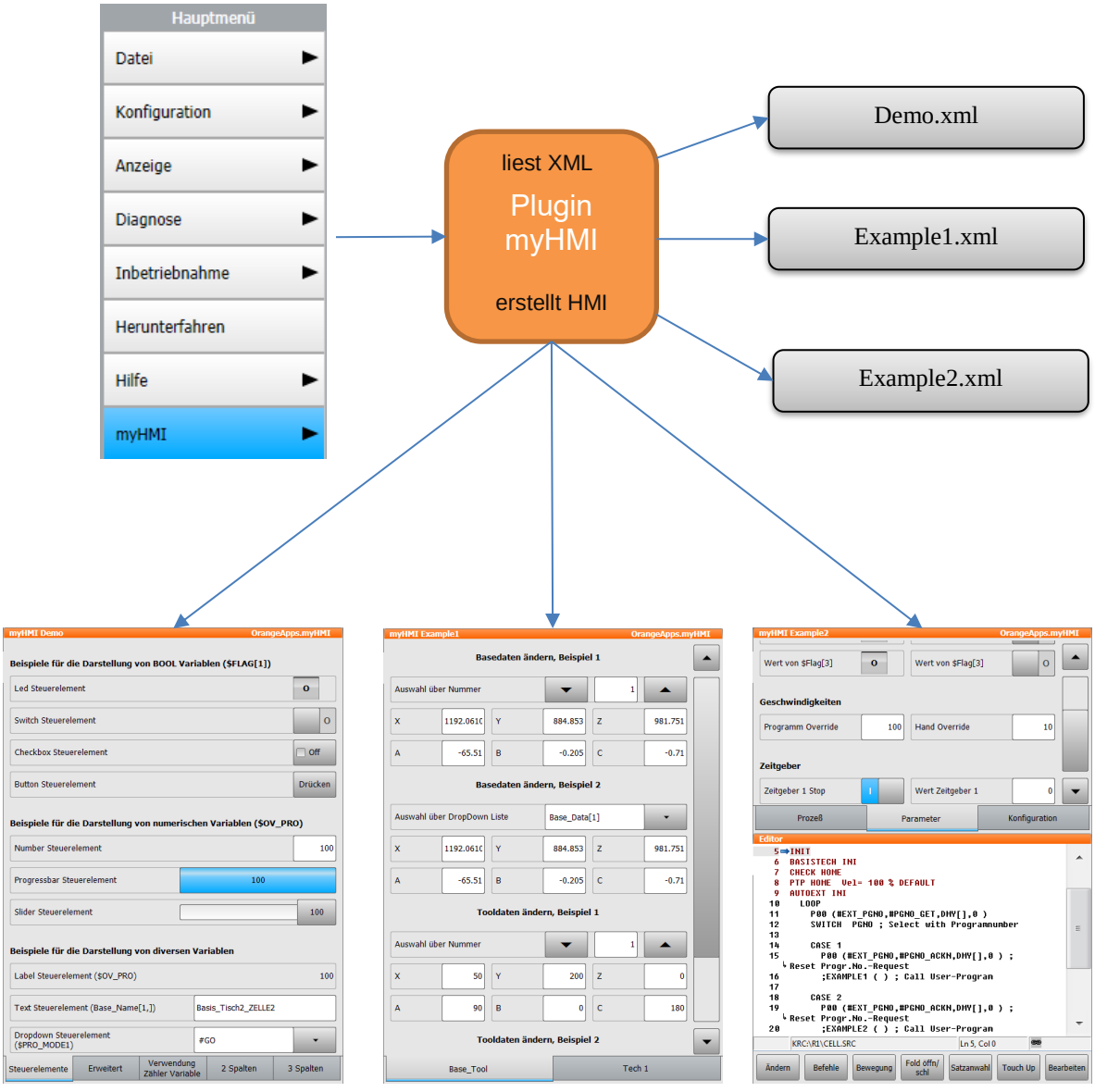
HMI_Name = Platzhalter für Namen der HMI, frei wählbar

Zur Erstellung und zur Bearbeitung von XML-Dateien wird „Notepad++“ empfohlen. Zum einen werden die XML-Inhalte farblich gut lesbar dargestellt und zum anderen werden grundsätzliche XML-Fehler von dem in Notepad++ enthaltenen XML-Parser bereits beim Speichern der Datei erkannt.

Um eine aktualisierte XML-Datei in myHMI darzustellen, ist es ausreichend das Anzeigefenster der HMI zu schließen und wieder zu öffnen. Änderungen in einer KXR-Sprachdatei erfordern einen Neustart des SmartHMI, oder alternativ einen Kaltstart der Roboter-Steuerung.

5.1 Schematische Darstellung des Funktionsprinzips

Nach dem Aufruf einer HMI aus dem Hauptmenü interpretiert das Plugin die dazugehörige XML-Datei und erstellt die jeweilige HMI.



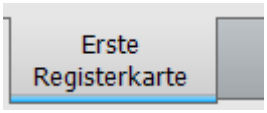
5.2 Gruppierung der Steuerelemente auf Registerkarten

Die Gruppen dienen zur thematischen Trennung von Inhalten und werden auf der HMI als Registerkarten angezeigt. Es muss mindestens eine Gruppe angegeben werden, und es können bis zu fünf Gruppen angegeben werden. Eine Gruppe wird durch den Knoten **<Group…> …. </Group>** definiert.

Beispiel XML

```
<Configuration Text="MeinHmi_Titel" Module="MeinHmi">
<Group Text="Erste Registerkarte">
  . . .
</Group>
</Configuration>
```

Beispiel Darstellung



Wird innerhalb einer XML-Datei nur eine Registerkarte definiert, wird in der HMI **keine** Registerkarte dargestellt. Stattdessen kann der Platz für ein weiteres Steuerelement genutzt werden.

5.3 Steuerelemente

Steuerelemente werden durch den Knoten **<Control ….>** definiert.

Zur Darstellung und Manipulation der KRL-Variablen stehen verschiedene Steuerelemente zur Verfügung. Diese Steuerelemente werden in der XML-Datei mit den KRL-Variablen verknüpft. Die Darstellung und das Verhalten können über verschiedene Argumente beeinflusst werden. Im einfachsten Fall benötigt ein Steuerelement die Argumente ***Type, Text und KrlVar.*** Die nachfolgende Beschreibung der Steuerelemente zeigt die mindestens notwendigen Argumente. Weitere Argumente für jedes Steuerelement werden in Kapitel 0 beschrieben.

Um Steuerelemente innerhalb einer Registerkarte zu definieren, muss die Angabe innerhalb des Knotens <Group ….> … </Group> erfolgen.

```
<Group Text="…">
<Control Type="Number"  Text="Path Velocity" KrlVar="$Vel.CP"/>
  . . .
</Group>
```

5.3.1 Schreibweise in der XML-Datei

Die Schreibweise in der XML-Datei richtet sich nach den gängigen Formatierungsregeln für XML-Dateien.

Ein Steuerelement wird innerhalb des Knotens **<Control**…./**>** angegeben. Der Typ des Steuerelements wird durch das Argument ***Type*** gekennzeichnet. Die Angabe der Wertzuweisung zu den einzelnen Argumenten wird innerhalb Anführungszeichen (") geschrieben.

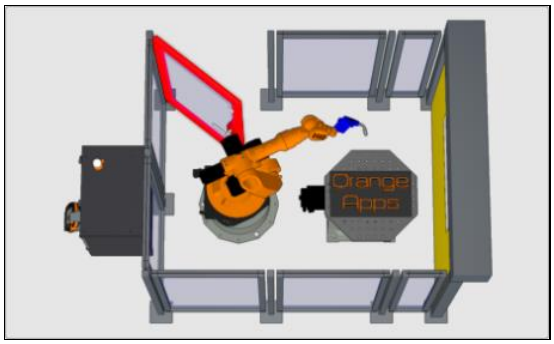
Beispiel für ein Steuerelement vom Typ "Number" zur Darstellung von INT- oder REAL-Variablen

```
<Control Type="Number"  Text="Bahngeschwindigkeit" KrlVar="$Vel.CP"/>
```

5.3.2 Übersicht verfügbare Steuerelemente

Folgende Steuerelemente stehen zur Verfügung:

- Picture



- Number

- LED

- Switch

- **SwitchIO (neu in V1.2)**

- Checkbox

- Button

- DropDown

- Text

- Label

- Progressbar

- Slider

- Headline

Beispiele für die Darstellung von numerischen Variablen ($OV_PRO)

Alle Steuerelemente können optional durch eine Vielzahl von Argumenten in Funktion und Aussehen beeinflusst werden.

5.3.3 Darstellungsgröße der Steuerelemente

Außer dem Steuerelement „Picture“ werden alle Steuerelemente mit einer Höhe von 40 Pixel dargestellt.

5.3.4 Steuerelement "Number"

Dient zur Darstellung von INT- oder REAL-Variablen. Das Steuerelement ermittelt automatisch den Variablentyp (INT oder REAL) und setzt den entsprechenden Keyboardtyp zur Werteeingabe.

Beispiel 1

Eintrag in der XML-Datei

```
<Control Type="Number"  Text="Bahngeschwindigkeit" KrlVar="$Vel.CP"
```

Beispiel 2

Eintrag in der XML-Datei

```
<Control Type="Number" Text="Geschwindigkeit" KrlVar="$OV_PRO"
VisibleLevel="0" EditLevel="20" Min="0" Max="100" Step="10"/>
```

Die Angabe des Min- und Max-Wertes kann auch über eine KRL-Variable erfolgen.

```
<Control Type="Number" Text="Geschwindigkeit" KrlVar="$OV_PRO"
VisibleLevel="0" EditLevel="20" Min="iMinValue" Max="iMaxValue" Step="10"/>
```

5.3.5 Steuerelement "LED”

Dient zur Darstellung von booleschen Variablen.

Beispiel

Beispiel Eintrag in der XML-Datei

```
<Control Type="Led" Text="Motor Drehtisch 1" KrlVar="$FLAG[1]"
```

Durch Angabe des Arguments "Negate" kann die Anzeige der LED negiert werden.

5.3.6 Steuerelement "Switch”

Stellt den Zustand boolescher Variablen dar und toggelt deren Wert zwischen TRUE und FALSE.

Funktion: Schalter, rastend

Beispiel

Eintrag in der XML-Datei

```
<Control Type="Switch" Text="Flag 1" KrlVar="$FLAG[1]"
```

Durch Angabe des Arguments "Negate" kann die Funktion des Schalters negiert werden.

5.3.7 Steuerelement "SwitchIO”

Stellt den Zustand boolescher Variablen dar und toggelt deren Wert zwischen TRUE und FALSE. Die LED zeigt den Zustand des Schalters an. LED aus = Schalter 0, LED ein = Schalter 1

Funktion: Schalter, rastend

Beispiel

Eintrag in der XML-Datei

```
<Control Type="SwitchIO" Text="Flag 1" KrlVar="$FLAG[1]"
```

Durch Angabe des Arguments "Negate" kann die Funktion des Schalters negiert werden.

```
<Control Type="SwitchIO" Text="Flag 1" KrlVar="$FLAG[1]" Negate="TRUE"
```

5.3.8 Steuerelement "Checkbox”

Stellt den Zustand boolescher Variablen dar und toggelt deren Wert zwischen TRUE und FALSE.

Funktion: Schalter, rastend

Beispiel

Eintrag in der XML-Datei

```
<Control Type="Checkbox" Text="Checkbox Steuerelement" KrlVar="$Flug[1]"/>
```

Durch Angabe des Arguments "Negate" kann die Funktion des Schalters negiert werden.

5.3.9 Steuerelement "Button”

Setzt den Wert einer booleschen Variable auf TRUE oder FALSE.

Funktion: Taster, nicht rastend

Standardwert beim Betätigen: TRUE

Beispiel

Eintrag in der XML-Datei

```
<Control Type="Button" Text="Button Steuerelement" TextButton="Drücken"
KrlVar="$Flag[1]"/>
```

Durch Angabe des Arguments "Negate" kann die Funktion des Tasters negiert werden.

5.3.10 Steuerelement "DropDown”

Dient zur Darstellung und Manipulation von Variablen vom Typ ENUM.

Beispiel

Eintrag in der XML-Datei

```
<Control Type="Dropdown" Text="Interpolations Modus" KrlVar="$IPO_MODE" >
    <DropDownItem Text="Basis" Value="#Base"/>
    <DropDownItem Text="Werkzeug" Value="#TCP"/>
 </Control>
```

Die Einträge im Dropdownmenü erfolgen über das Element "DropDownItem" und dessen Argumente "Value" und "Text" (optional).

Argumente des Elements "DropdownItem"

Argument	Beschreibung
Text	Angezeigter Text im Dropdown Menü (optional)
Value	Wert, welcher der verbundenen KRL-Variable zugewiesen wird

Neben ENUM-Variablen, lassen sich auch KRL Variablen vom Typ CHAR, INT, REAL, BOOL mit diesem Steuerelement verknüpfen. Es muss dann darauf geachtet werden, dass die verfügbaren Werte im DropDown kompatibel zur Ziel-KRL-Variable sind.

5.3.11 Steuerelement "Text"

Dient zur Darstellung und Manipulation von CHAR-Variablen.

Beispiel

Eintrag in der XML-Datei

```
<Control Type="Text" Text="Bezeichnung Base 1" KrlVar="Base_Name[1,]"/>
```

5.3.12 Steuerelement "Label"

Dient zur Darstellung beliebiger Variablentypen.

Beispiel

Eintrag in der XML-Datei

```
<Control Type="Label" Text="Label Steuerelement ($OV_PRO)"
KrlVar="$OV_PRO"/>
```

5.3.13 Steuerelement "Progressbar"

Dient zur Darstellung von INT- oder REAL-Variablen

Beispiel

Eintrag in der XML-Datei

```
<Control Type="Progressbar" Text="Progressbar Steuerelement"
KrlVar="$OV_PRO" Min="0" Max="100"/>
```

Die Angabe des Min- und Max-Wertes kann auch über eine KRL-Variable erfolgen.

```
<Control Type="Progressbar" Text="Progressbar Steuerelement"
KrlVar="$OV_PRO" Min="iMinValue" Max="iMaxValue"/>
```

5.3.14 Steuerelement "Slider”

Dient zur Darstellung und Manipulation von INT- oder REAL-Variablen.

Beispiel

Eintrag in der XML-Datei

```
<Control Type="Slider" Text="Slider Steuerelement" KrlVar="$OV_PRO" Min="0"
Max="100"/>
```

Die relative Platzierung des Schiebereglers erfolgt mittels der Angabe der Argumente "Min" und "Max". Diese sind standardmäßig 0 bzw. 100. Befindet sich der Wert der KRL-Variable immer nur innerhalb dieser Grenzen, ist die Angabe von "Min" und "Max" nicht notwendig. Nimmt die KRL-Variable Werte außerhalb des Wertebereichs von "Min" und "Max" an, wird der Hintergrundbalken rot dargestellt.

Die Angabe des Min- und Max-Wertes kann auch über eine KRL-Variable erfolgen.

```
<Control Type="Slider" Text="Slider Steuerelement" KrlVar="$OV_PRO"
Min="iMinValue" Max="iMaxValue"/>
```

5.3.15 Steuerelement "Headline”

Dient zur Darstellung von Beschriftungen und Überschriften

Beispiel

Beispiele für die Darstellung von numerischen Variablen ($OV_PRO)

Eintrag in der XML-Datei

```
<Control Type="Headline" Text="Beispiele für die Darstellung von numerischen
Variablen ($OV_PRO)"/>
```

5.3.16 Steuerelement "Picture”

Das Steuerelement „Picture“ dient zur statischen und dynamischen Darstellung von Grafiken. Zur dynamischen Darstellung wird die Angabe eine KRL-Variablen und eines Platzhalters benötigt.

Es besteht auch die Möglichkeit einzelne Grafiken zu einer Gesamtgrafik zu überlagern. Dazu muss die KRL-Variable eine Struktur sein.

Der Speicherort der Grafikdateien kann sowohl lokal als auch auf einem Netzlaufwerk (Dateifreigabe muss beachtet werden) sein.

5.3.16.1 Zulässige Grafikformate

Zulässige Grafikformate sind:

- BMP
- GIF
- JPEG
- PNG
- TIFF

Bei Verwendung eines falschen Grafikformats wird eine Fehlermeldung ausgegeben.

5.3.16.2 Darstellungsgröße der Grafiken

Die Grafiken werden standardmäßig (ohne Größenangabe) in Originalgröße dargestellt. Über das optionale Argument „Width“ kann die Größe skaliert werden.

Alle anderen Steuerelemente werden standardmäßig mit einer Höhe von 40 Pixel dargestellt.

5.3.16.3 Argumente des Steuerelements „Picture“

Über verschiedene Argumente können das Aussehen und das Verhalten des Steuerelements beeinflusst werden.

5.3.16.3.1 Argument „Path“ - Pfadangabe zur ladenden Grafikdatei

Das Argument „Path“ legt den Namen der zu ladenden Grafikdateien fest. Dieser Pfad kann sowohl absolut als auch relativ angegeben werden. Bei relativer Angabe ist der Bezugsordner c:\KRC\User\myHMI.

Beispiel: Die Grafiken liegen im Ordner C:\KRC\User\myHMI\myPics

Absolute Angabe:

```
<Control Type="Picture" Text="myHMI is powered by OrangeApps"
Path="C:\KRC\User\myHMI\myPics\logo_orangeapps.png" Border="False"/>
```

Relative Angabe:

```
<Control Type="Picture" Text="myHMI is powered by OrangeApps"
Path="myPics\logo_orangeapps.png" Border="False"/>
```

5.3.16.3.2 Argument „KrlVar“ (optional) – KRL-Variable zur dynamischen Darstellung

Durch Angabe einer KRL-Variablen und des Platzhalters {0} kann der Name der anzuzeigenden Grafik dynamisch generiert werden.

```
<Control Type="Picture" Text="Power Laser enabled" Path="D:\IO_{0}.png"
KrlVar="$IN[1]"/>
```

Der Wert der KRL-Variable wird überwacht, ausgelesen und in einen String gewandelt. Der Platzhalter wird durch den String ersetzt und somit der Dateiname gebildet.

5.3.16.3.3 Argument „Width“ (optional) – Darstellungsgröße einer Grafik

Ohne die Angabe der des Elements „Width“ wird die Grafik in Originalgröße dargestellt. Über das Element „Width“ wird die Grafik in der Breite skaliert. Die Anpassung der Höhe erfolgt entsprechend dem originalen Seitenverhältnis. Die Angabe von „Width“ erfolgt in Pixeln.

Beispiel:

```
<Control Type="Picture" Text="Power Laser enabled" Path="D:\IO_{0}.png"
KrlVar="$IN[1]" Width="80"/>
```

Angenommen die Originalgröße einer Grafik beträgt 50x80 Pixel. Width wird mit 80 angegeben

➔ Der Vergrößerungsfaktor beträgt 1,6. Die Grafik wird mit 80x128 Pixel dargestellt

5.3.16.3.4 Argument „Border“ (optional) – Grafiken umranden

Das Arguments „Border“ bestimmt ob ein Rahmen um die Grafik gezeichnet werden soll.

Bei Border=“True“ wird ein Rahmen gezeichnet, bei „False“ wird der Rahmen nicht gezeichnet. Die Voreinstellung ist „TRUE“

Beispiel:

```
<Control Type="Picture" Path="D:\IO_{0}.png" KrlVar="$IN[1]" Width="80"
Border="FALSE"/>
```

5.3.16.3.5 Argument „Alignment“ (optional) – Grafiken ausrichten

Mit dem Argument „Alignment“ kann die Ausrichtung des Bildes zwischen links, mittig und rechts geändert werden. Ohne Angabe von „Alignment“ werden die Grafiken am rechten Rand ausgerichtet.

Beispiel:

```
<Control Type="Picture" Path="C:\KRC\User\myHMI\myPics\logo_orangeapps.png"
Alignment="Left"/>
```

5.3.16.3.6 Argument „Text“ (optional) – Text einblenden

Mit dem Argument „Text“ kann zusätzlicher Text eingeblendet werden. Dieser Text kann auch mehrsprachig angezeigt werden (siehe Kapitel „Mehrsprachigkeit“).

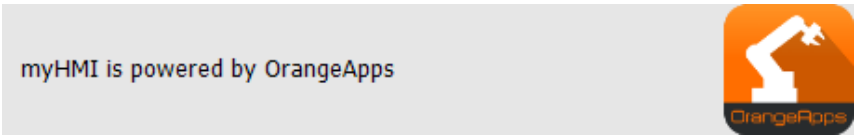
Beispiel:

```
<Control Type="Picture" Text="Power Laser enabled" Path="D:\IO_{0}.png"
KrlVar="$IN[1]" Width="80" Border="FALSE"/>
```

5.3.16.4 Statische Darstellung

Bei der statischen Darstellung immer dasselbe Bild angezeigt.

Beispiel statisch



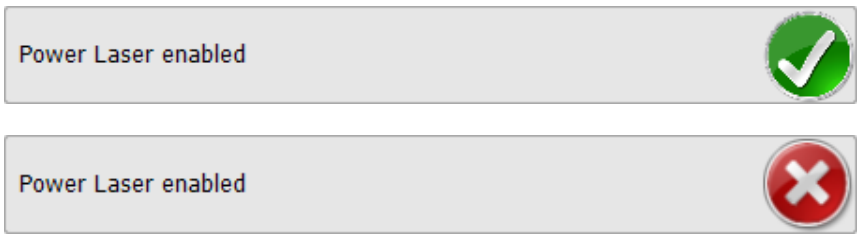
Eintrag in der XML-Datei

```
<Control Type="Picture" Text="myHMI is powered by OrangeApps"
Path="Pics\logo_orangeapps.png" Border="False"/>
```

5.3.16.5 Dynamische Darstellung

Zur dynamischen Anzeige von Bilden wird der Pfad zur Bilddatei aus der Angabe im Element „Path“ und dem Wert einer KRL-Variablen zusammengebaut. Im Dateinamen muss dazu der Platzhalter {0} verwendet werden. Der Wert der Variable wird als String interpretiert und an der Stelle des Platzhalters {0} in den Dateinamen eingebaut. Ändert sich der Wert der KRL-Variable so ändert sich automatisch das angezeigte Bild.

Beispiel dynamisch



Eintrag in der XML-Datei

```
<Control Type="Picture" Text="Power Laser enabled" Path="D:\IO_{0}.png"
KrlVar="$IN[1]"/>
```

➔ Zu ladende Grafik:

Wert der Variable $IN[1]	Zu ladende Bilddatei
FALSE	D:\IO_FALSE.PNG
TRUE	D:\IO_TRUE.PNG

5.3.16.6 Dynamische Überlagerung von Einzelgrafiken zu einer Gesamtgrafik

Ähnlich wie bei der dynamischen Darstellung wird zur überlagerten Anzeige von Grafiken der Platzhalter **{0}** im Element „Path“ verwendet. Zur Darstellung von überlagerten Grafiken zu einem Gesamtgrafik wird eine KRL-Variable vom Typ „Struktur“ benötigt. myHMI ermittelt nacheinander alle Strukturelemente und deren Wert, setzt diese zu einem String zusammen und ersetzt den Platzhalter im Dateinamen durch den jeweiligen String nach folgendem Prinzip:

Erste Grafik: String aus Path + Name des 1. Strukturelements + Wert des 1. Strukturelements

Zweite Grafik: String aus Path + Name des 2. Strukturelements + Wert des 2. Strukturelements

usw.

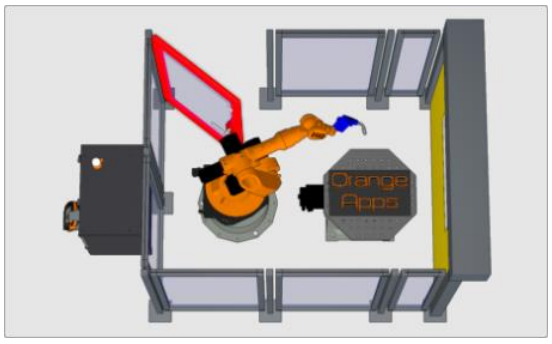
letzte Grafik: String aus Path + Name des letzten Strukturelements + Wert des letzten Strukturelements

Es werden so viele Grafiken angezeigt wie es Strukturelemente gibt. Die Reihenfolge der Bildüberlagerung wird bestimmt durch die Reihenfolge der Definition der Strukturelemente.

Der Name der Strukturvariablen und deren Elemente ist frei wählbar. Nicht erlaubt ist Struktur in Struktur.

Ändert sich der Wert der KRL-Variable so ändert sich automatisch das angezeigte Gesamtbild.

Beispiel mit Überlagerung, die Strukturvariable heißt „myHMIPicDemo“ mit den boolschen Elementen Gate, Background, Door, Turntable und Robot



Das Gesamtbild setzt sich aus fünf verschiedenen Einzelgrafiken zusammen.

Eintrag in der XML-Datei (Annahme: Die Grafiken sind im Ordner D:\myHMI\Pics gespeichert)

```
<Control Type="Picture" Path="D:\myHMI\Pics\myHMI_{0}.png"
KrlVar="myHMIPicDemo"/>
```

Die Variable myHMIPicDemo ist eine Struktur mit folgenden boolschen Elementen:

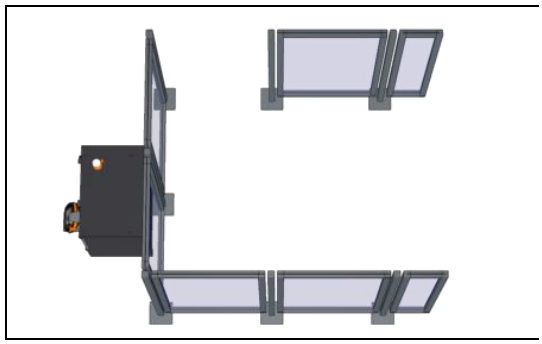
```
GLOBAL STRUC strPicDemo BOOL Gate,BackGround,Door,TurnTable,Robot

DECL GLOBAL strPicDemo myHMIPicDemo={Gate TRUE,BackGround TRUE,Door
TRUE,TurnTable FALSE,Robot FALSE}
```

Wie ergeben sich die Bildnamen?

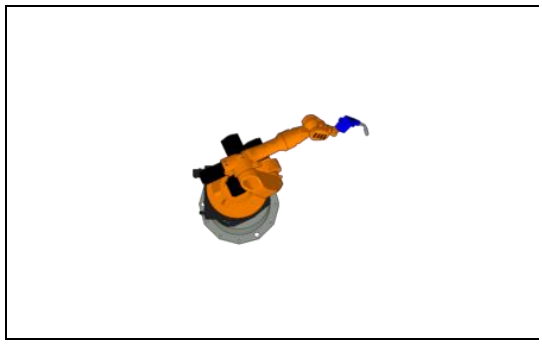
1. myHMI sucht nach den Elementen der Variable myHMIDemo. Diese sind Gate, Background, Door, TurnTable und Robot
2. myHMI ermittelt die aktuellen Werte der Elemente
3. myHMI setzt den Namen jedes Elements und dessen Wert zu einem String zusammen, also z.B. GateTrue
4. Der Platzhalter {0] im Pfad D:\myHMI\Pics\myHMI_{0}.png wird durch den ermittelten String ersetzt, also z.B. D:\myHMI\Pics\myHMI_GateTrue.png
5. Dieses Bild wird nun angezeigt

myHMI führt diese Schritte für jedes Strukturelement aus. Die Reihenfolge der Elemente in der Definition der Struktur bestimmt die Reihenfolge der Bildanzeige.

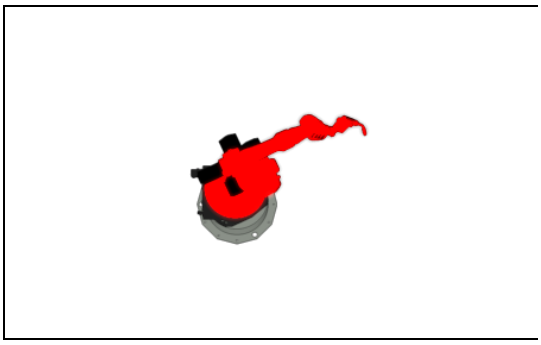
Im Ordner D:\myHMI\Pics befinden sich folgende Grafiken:



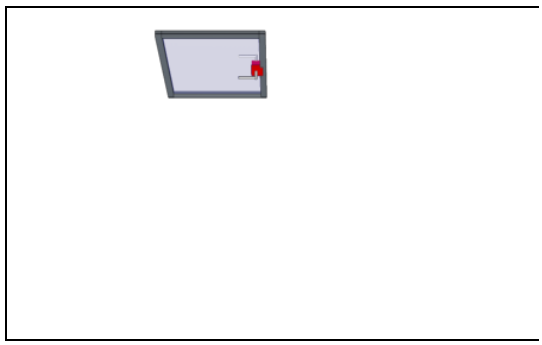
myHMI_BackgroundTrue.png



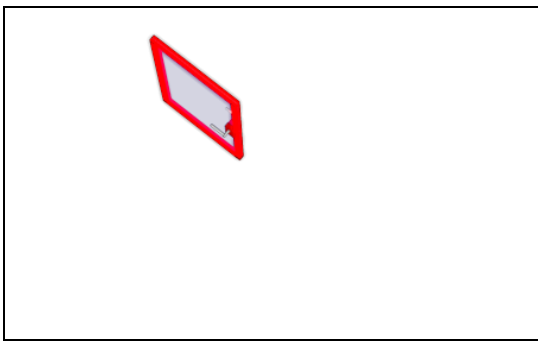
myHMI_RobotTrue.png



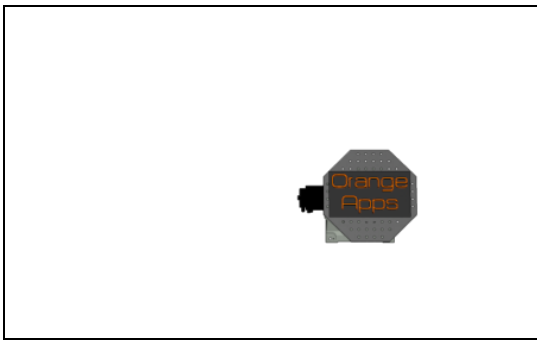
myHMI_RobotFalse.png



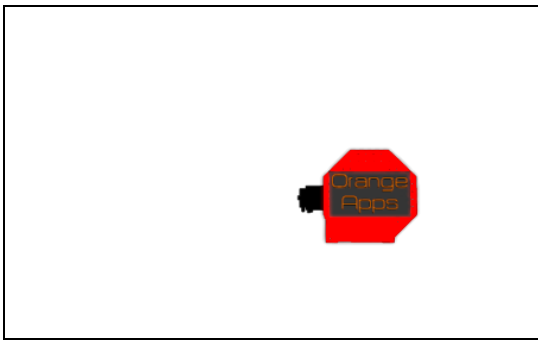
myHMI_DoorTrue.png



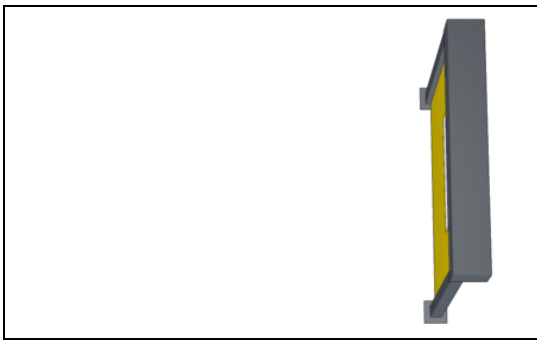
myHMI_DoorFalse.png



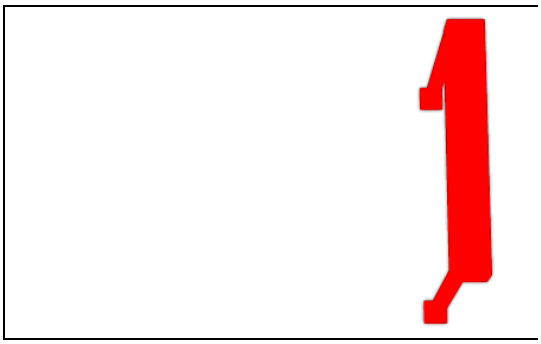
myHMI_TurnTableTrue.png



myHMI_TurnTableFalse.png



myHMI_GateTrue.png



myHMI_GateFalse.png

Beispiel

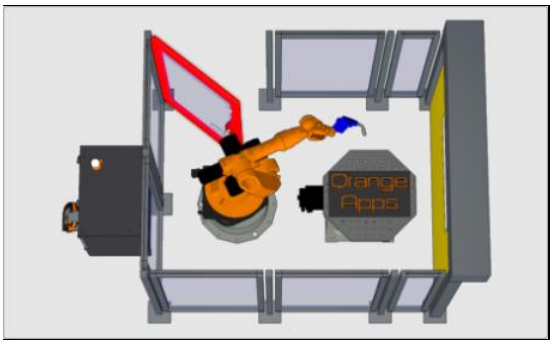
Bei folgenden Werten

```
DECL GLOBAL strPicDemo myHMIPicDemo={Gate TRUE,BackGround TRUE,Door
FALSE,TurnTable TRUE,Robot TRUE}
```

werden folgende Grafiken angezeigt:

Element	Wert	Zu ladende Bilddatei
Gate	TRUE	D:\myHMI\Pics\myHMI_GateTrue.png
Background	TRUE	D:\myHMI\Pics\myHMI_BackgroundTrue.png
Door	FALSE	D:\myHMI\Pics\myHMI_DoorFalse.png
TurnTable	TRUE	D:\myHMI\Pics\myHMI_TurnTableTrue.png
Robot	TRUE	D:\myHMI\Pics\myHMI_RobotTrue.png

Resultat:



Sollen weitere Grafiken angezeigt werden, muss nur die Struktur entsprechend erweitert und die jeweiligen Grafiken dem Ordner D:\myHMI\Pics hinzugefügt werden.

Beispiel

mit Überlagerung, die Strukturvariable heißt „myCellView“ mit den Elementen Gate, Background, Door, Turntable und Robot (alle vom Typ Integer)

Eintrag in der XML-Datei (Annahme: Die Grafiken sind im Ordner C:\KRC\User\myHMI\Pics gespeichert)

```
<Control Type="Picture" Path="Pics\{0}.png" KrlVar="myCellView"/>
```

Bei folgenden Werten

```
DECL GLOBAL strCellView myCellView={Gate 0,BackGround 1,Door 2,TurnTable
1,Robot 3}
```

(die Werte der Variablen wurden exemplarisch angenommen)

werden folgende Grafiken angezeigt:

Element	Wert	Zu ladende Bilddatei
Gate	0	C:\KRC\User\myHMI\Pics\Gate0.png
Background	1	C:\KRC\User\myHMI\Pics\Background1.png
Door	2	C:\KRC\User\myHMI\Pics\Door2.png
TurnTable	1	C:\KRC\User\myHMI\Pics\TurnTable1.png
Robot	3	C:\KRC\User\myHMI\Pics\Robot3.png

Gerne unterstützen wir Sie bei der Erstellung ansprechender Grafiken für Ihre HMI. Kontaktieren Sie uns einfach unter info@orangeapps.de.

5.3.17 Gesamtübersicht aller verfügbarer Argumente

Für jedes Steuerelement stehen weitere Argumente zur Verfügung. Über diese kann das Aussehen und das Verhalten jedes Steuerelements beeinflusst werden.

Liste aller möglichen Argumente

Argument	Beschreibung	Optional	Standardwert
Alignment	Bei Steuerelement Label: Richtet den Beschriftungstext aus (links, mittig, rechts) Bei Steuerelement Picture: Richtet die Grafik aus (links, mittig, rechts)	Ja	Bei Label: links Bei Picture: rechts
AreYouSure	True = Sicherheitsabfrage bei Wert-Änderung einblenden (Dialog)	Ja	False
Border	Steuert ob ein Bild mit oder ohne Rahmen dargestellt wird (mit=TRUE)	Ja	TRUE
Color0	Farbe der LED des Steuerelements bei Zustand False Mögliche Werte: Grey, Green, Red, Yellow	Ja	Gray
Color1	Farbe der LED des Steuerelements „Led“ bei Zustand TRUE Mögliche Werte: Grey, Green, Red, Yellow	Ja	Green
ColSpan	Anzahl Spalten über die sich das Element erstreckt	Ja	1
Description	Wird auf die Beschriftung geklickt, so wird dieser Text als Beschreibung angezeigt	Ja	
Format	Formatierung von INT und REAL Werten	Ja	
KrlVar	Verknüpfte KRL Variable	Nein	
Max	Größter zulässiger Wert	Ja	
Min	Kleinster zulässiger Wert	Ja	
ModeOP	Betriebsart ab der das Element editierbar ist	Ja	31
Module	Modul / KXR-Datei für multilinguale Inhalte	Ja	Wert aus *Gruppe*
NeedDrivesReady	Editierbarkeit des Elements ist vom Zustand der Antriebe abhängig	Ja	False
NeedSafetySwitch	Editierbarkeit des Elements ist vom Zustand des Zustimmschalter abhängig	Ja	False
Negate	Invertiert eine boolesche Variable	Ja	False

Path	Legt den Pfad zu einer Bilddatei fest	Nein	
ProState0	Editierbarkeit des Elements ist vom Zustand des Submit-Interpreters abhängig	Ja	63
ProState1	Editierbarkeit des Elements ist vom Zustand des Programm-Interpreters abhängig	ja	63
Step	Schrittweite für Auf-/Abtaste	Ja	
Text	Beschriftungstext oder Key für das Element	Nein	
Text0	Beschriftung der Schaltfläche des Steuerelements "Checkbox" bei Zustand False	Ja	Aus
Text1	Beschriftung der Schaltfläche des Steuerelements "Checkbox" bei Zustand True	Ja	Ein
TextButton	Beschriftung der Schaltfläche des Steuerelements "Button"	Ja	
UserLevelEdit	Benutzergruppe ab der das Element editierbar ist	Ja	0
UserLevelVisible	Benutzergruppe ab der das Element sichtbar ist	Ja	0
Value	Wert eines Eintrags im Steuerelement "DropDown". Wird bei Auswahl des Eintrags an die Variable des Arguments "KrlVar" übergeben.	Nein	
Width	Legt die Breite einer Grafik fest (Pixel). Die Höhe wird entsprechend proportional skaliert.	Ja	Breite des Bildes

Argumente/Elemente-Matrix

Argument	<Configuration>	<Group>	Number	Led	Switch	SwitchIO	Checkbox	Button	Slider	Progressbar	Dropdown	Text	Label	Headline	Picture
Alignment	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	O	O
AreYouSure	n/a	n/a	O	n/a	O	O	O	n/a	O	n/a	O	O	n/a	n/a	n/a
Border	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	O
Color 0/1	n/a	n/a	n/a	O	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Columns	n/a	O	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
ColSpan	n/a	n/a	O	O	O	O	O	O	O	O	O	O	O	O	O
Description	n/a	n/a	O	O	O	O	O	O	O	O	O	O	O	n/a	n/a
Format	n/a	n/a	O	n/a	n/a	n/a	n/a	n/a	O	O	n/a	n/a	n/a	n/a	n/a
KrlVar	n/a	n/a	X	X	X	X	X	X	X	X	X	X	X	n/a	O
Max	n/a	n/a	O	n/a	n/a	n/a	n/a	n/a	O	O	n/a	n/a	n/a	n/a	n/a
Min	n/a	n/a	O	n/a	n/a	n/a	n/a	n/a	O	O	n/a	n/a	n/a	n/a	n/a
ModeOP	n/a	n/a	O	n/a	O	O	O	O	O	n/a	O	O	n/a	n/a	n/a
Module	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
NeedDrivesReady	n/a	n/a	O	n/a	O	O	O	O	O	n/a	O	O	n/a	n/a	n/a
NeedSafetySwitch	n/a	n/a	O	n/a	O	O	O	O	O	n/a	O	O	n/a	n/a	n/a
Negate	n/a	n/a	n/a	O	O	O	O	O	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Path	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	X
ProState0	n/a	n/a	O	n/a	O	O	O	O	O	n/a	O	O	n/a	n/a	n/a
ProState1	n/a	n/a	O	n/a	O	O	O	O	O	n/a	O	O	n/a	n/a	n/a
Step	n/a	n/a	O	n/a	n/a	n/a	n/a	n/a	O	n/a	n/a	n/a	n/a	n/a	n/a
Text	X	X	X	X	X	X	X	X	X	X	X	X	X	X	O
Text0/1	n/a	n/a	n/a	n/a	n/a	n/a	O	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
TextButton	n/a	n/a	n/a	n/a	n/a	n/a	n/a	O	n/a	n/a	n/a	n/a	n/a	n/a	n/a
UserLevelEdit	n/a	n/a	O	n/a	O	O	O	O	O	n/a	O	O	n/a	n/a	n/a
UserLevelVisible	n/a	n/a	O	O	O	O	O	O	O	O	O	O	O	O	O

Value	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	X	n/a	n/a	n/a	n/a
Width	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	O

X: Angabe zwingend erforderlich | O: Angabe optional | n/a: nicht verfügbar

5.3.17.1 Argument "Alignment" für Steuerelement Headline und Picture

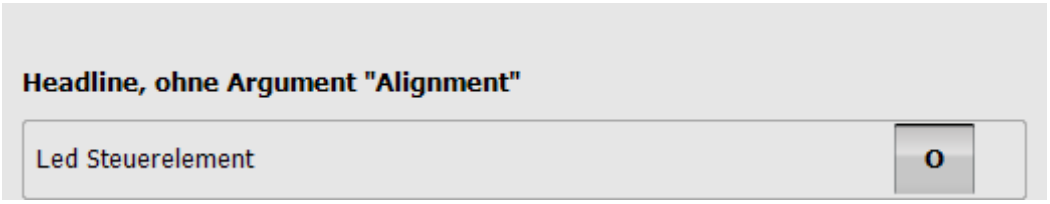
Durch Angabe des Arguments "Alignment" kann der Text innerhalb des Steuerelements "Headline" individuell ausgerichtet werden.

Format: **String**

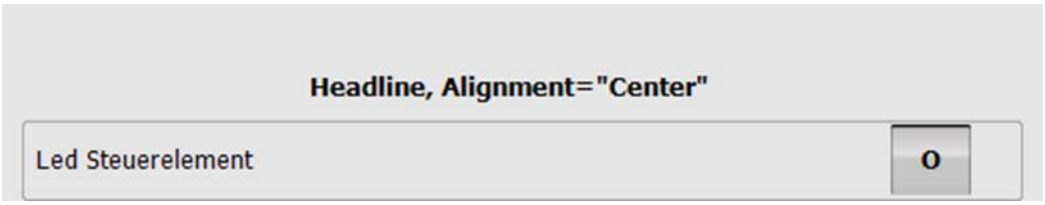
Zulässige Werte

Wert	Ausrichtung des Argument "Text"	Standard bei
Left	links	Steuerelement Headline
Center	Mittig	-
Right	rechts	Steuerelement Picture

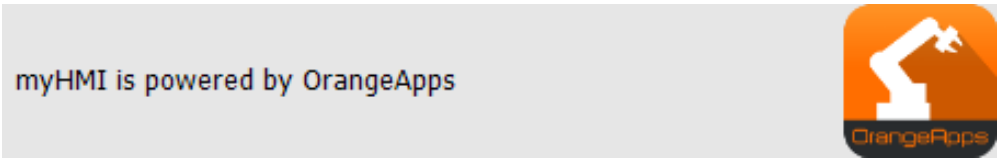
Beispiel Headline ohne Angabe von Alignment



Beispiel Headline bei Alignment="Center"



Beispiel Picture ohne Angabe von Alignment



5.3.17.2 Argument "AreYouSure”

Wird das Argument "AreYouSure” mit dem Wert TRUE angegeben, wird beim Ändern eines Wertes eine Dialogmeldung mit der Frage, ob die Änderung durchgeführt werden soll, angezeigt.

Format: **BOOL**

Zulässige Werte:

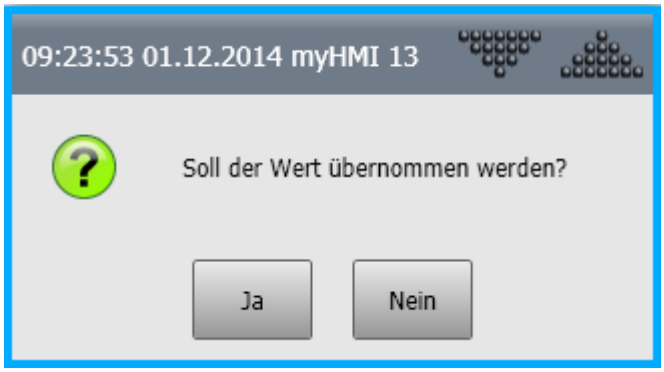
Wert	Auswirkung
TRUE	Dialogmeldung wird eingeblendet
FALSE	Dialogmeldung wird nicht eingeblendet

Standardwert: False

Mehrsprachfähig: ja

Beispiel

```
<Control Type="Switch" Text="Switch Steuerelement" KrlVar="$Flag[1]"
AreYouSure="True"/>
```



JA: Wertänderung wird übernommen
Nein: Wertänderung wird verworfen

5.3.17.3 Argument "Border”

Legt fest ob ein Bild mit oder ohne Rahmen angezeigt wird

Format: **BOOL**

Zulässige Werte:

Wert	Auswirkung
TRUE	Rahmen wird angezeigt
FALSE	Rahmen wird ausgeblendet

Standardwert: TRUE

Beispiel

```
<Control Type="Picture" Text="LasPowEnabled" Path="PicsDemo\led_{0}.png"
KrlVar="$Flag[1]" ColSpan="2" Border="FALSE"/>

<Control Type="Picture" Text="LasPowEnabled" Path="PicsDemo\IO_{0}.png"
KrlVar="$Flag[1]" Width="50" ColSpan="2" />

<Control Type="Picture" Text="LasPowEnabled" Path="PicsDemo\Smile{0}.png"
KrlVar="$Flag[1]" Width="50" ColSpan="2" Border="FALSE"/>
```

Nur beim zweiten Steuerelement wird ein Rahmen gezeichnet.

5.3.17.4 Argument "Color0 / Color1"

Das Argument "Color0 / Color1" setzt die Farbe des Steuerelements "LED" in den Zuständen False und True.

Color0 setzt die Farbe beim Zustand False. Color1 setzt die Farbe beim Zustand True.

Zulässige Werte:

Wert	Auswirkung
Gray	Steuerelement wird grau dargestellt
Green	Steuerelement wird grün dargestellt
Red	Steuerelement wird rot dargestellt

Yellow	Steuerelement wird gelb dargestellt

Standardwerte: Color0 = gray, Color 1= green

Beispiel Color0="yellow", Color1=”red”

```
<Control Type="Led" Text="Led Control" KrlVar="$FLAG[1]" Color0="Yellow"
Color1="Red"/>
```

Zustand der Variable = False

Zustand der Variable = TRUE

5.3.17.5 Argument "ColSpan”

Das Argument "ColSpan” legt die Anzahl der Spalten fest über die sich ein Steuerelement erstreckt. Es interagiert mit dem Argument "Columns". "Columns" legt die Anzahl der Spalten einer Registerkarte fest. Die Angabe von "ColSpan" wird nur ausgewertet, wenn "Colums" einen Wert größer 1 hat. Wird ColSpan nicht angegeben, erstreckt sich ein Steuerelement über eine Spalte.

Format: **INT**

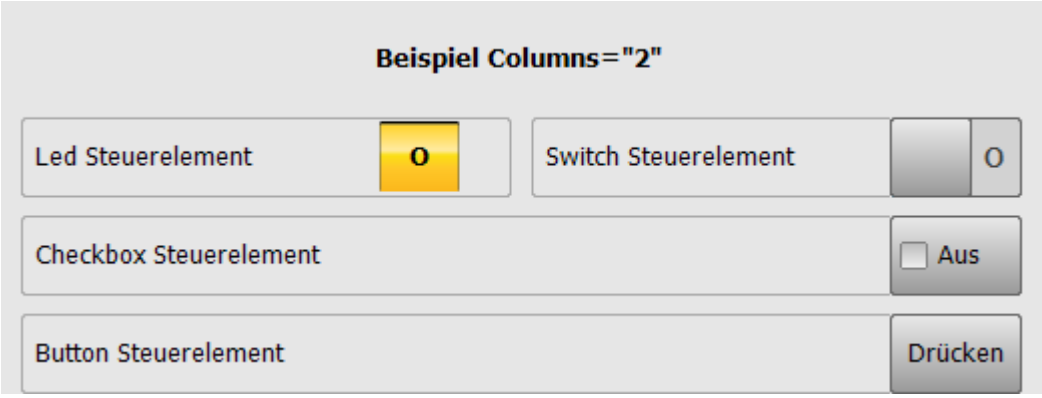
Zulässige Werte

Wert	Auswirkung
1	Steuerelement spannt sich über eine Spalte
2	Steuerelement spannt sich über zwei Spalten
3	Steuerelement spannt sich über drei Spalten

Standardwert: 1

Beispiel Columns="2"

```
<Group text="Steuerelemente" Columns="2">
      <Control Type="Headline" Text='Beispiel Columns="2"'
Alignment="MiddleCenter" ColSpan="2"/>
      <Control Type="Led" Text="Led Steuerelement" KrlVar="$FLAG[1]"
Color0="Yellow" Color1="Red"/>
      <Control Type="Switch" Text="Switch Steuerelement" KrlVar="$Flag[1]"/>
      <Control Type="Checkbox" Text="Checkbox Steuerelement"
KrlVar="$Flag[1]" ColSpan="2"/>
      <Control Type="Button" Text="Button Steuerelement"
TextButton="Drücken" KrlVar="$Flag[1]" ColSpan="2"/>
```



Erklärung

<GROUP>: Columns="2" legt die Spaltenanzahl der **gesamten** Registerkarte auf 2 fest.

HEADLINE: ColSpan="2" spannt das Steuerelement über 2 Spalten

LED: ColSpan nicht angegeben, das Steuerelement ist eine Spalte breit

SWITCH: ColSpan nicht angegeben, das Steuerelement ist eine Spalte breit

CHECKBOX: ColSpan ="2" spannt das Steuerelement über 2 Spalten

BUTTON: ColSpan ="2" spannt das Steuerelement über 2 Spalten

5.3.17.6 Argument "Description"

Legt einen zusätzlichen Beschreibungstext für ein Steuerelement fest. Tippen auf den Text des Steuerelements öffnet das Steuerelement und der Beschreibungstext wird sichtbar. Steuerelemente, für die ein Beschreibungstext vorhanden ist, werden mit einem Pfeil-Nach-Unten Symbol gekennzeichnet.

Format: **string**

Zulässige Zeichen: **String oder Key einer Sprachdatenbank**

Mehrsprachfähig: **ja**

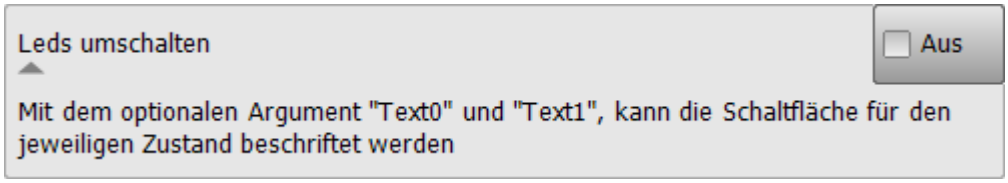
Für jeden String des Arguments "Description" wird nach einem Eintrag in einer Sprachdatenbank gesucht. Wird kein Eintrag gefunden, wird der angegebene String dargestellt.

Beispiel 1 ohne Datenbankeintrag

```
<Control Type="Checkbox" Text="Leds umschalten" Description="Mit dem
optionalen Argument "Text0" und "Text1" kann die Schaltfläche für den
jeweiligen Zustand beschriftet werden" KrlVar="$Flag[1]" Text0="Aus"
Text1="Aktiv"/>
```

Steuerelement zugeklappt:

Steuerelement aufgeklappt:



Beispiel 2 mit Datenbankeintrag

XML:

```
<Control Type="Checkbox" Text="Leds umschalten" Description="DescCheckbox"
KrlVar="$Flag[1]" Text0="Aus" Text1="Aktiv"/>
```

Kxr-Datei:

```
<uiText key="DescCheckBox">
   <text xml:lang="de-DEV"> Mit dem optionalen Argument "Text0" und "Text1"
kann die Schaltfläche für den jeweiligen Zustand beschriftet werden </text>
   <text xml:lang="en-DEV">By using the optional argument "Text0" and
"Text1" it's possible to parametrize the caption of the button </text>
   <text xml:lang="it-DEV">…
   </text>
</uiText>
```

➔ Für die HMI Sprache "Englisch", "Deutsch" und "Italienisch" wird der jeweilige Übersetzungstext aus der kxr-Datei verwendet, für alle anderen wird der englische Text verwendet.

5.3.17.7 Argument "Format"

Durch die Angabe einer Zahlenformatzeichenfolge wird die Wertanzeige verschiedener Steuerelemente unter Berücksichtigung des jeweiligen Gebietsschemas formatiert.

Verwendbar bei Steuerelement: **Label, Slider, Progressbar, Number (mit Einschränkungen)**

Zahlenformatzeichenfolgen

Format-bezeichner	Name	Beschreibung	Beispiele
"0"	0-Platzhalter	Ersetzt die Ziffer 0 ggf. durch eine entsprechende vorhandene Ziffer; andernfalls wird die Ziffer 0 in der Ergebniszeichenfolge angezeigt.	1234.5678 ("00000") -> 01235 0.45678 ("0.00", en-US) -> 0.46 0.45678 ("0.00", de-DE) -> 0,46
"#"	Ziffernplatzhalter	Ersetzt das "#"-Symbol ggf. durch eine entsprechende vorhandene Ziffer; andernfalls wird keine Ziffer in der Ergebniszeichenfolge angezeigt.	1234.5678 ("#####") -> 1235 0.45678 ("#.##", en-US) -> .46 0.45678 ("#.##", de-DE) -> ,46

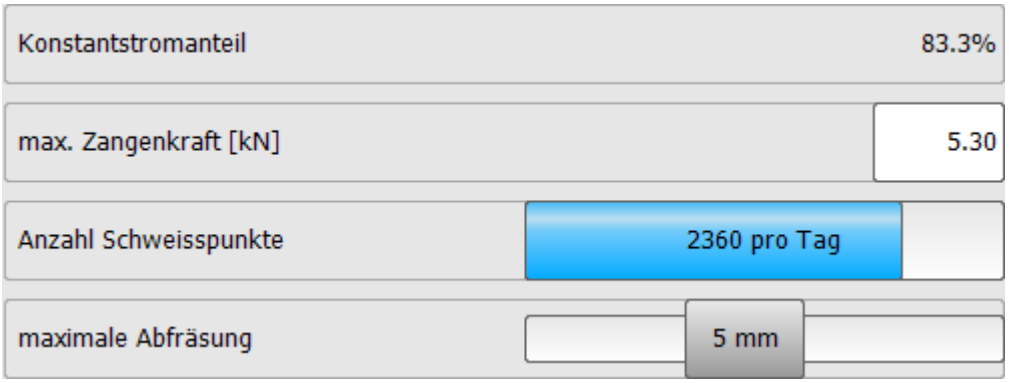
"."	Dezimaltrennzeichen	Bestimmt die Position des Dezimaltrennzeichens in der Ergebniszeichenfolge.	0.45678 ("0.00", en-US) -> 0.46 0.45678 ("0.00", de-DE) -> 0,46
","	Gruppentrenn-zeichen und Zahlenskalierung	Das Zeichen wird sowohl als Bezeichner für Gruppentrennzeichen als auch als Bezeichner für Zahlenskalierung verwendet. Bei einer Verwendung als Bezeichner für Gruppentrennzeichen wird ein lokalisiertes Trennzeichen zwischen die einzelnen Gruppen eingefügt. Bei einer Verwendung als Bezeichner für Zahlenskalierung wird eine Zahl für jedes angegebene Zeichen durch 1000 geteilt.	Bezeichner für Gruppentrennzeichen: 2147483647 ("##,#", en-US) -> 2,147,483,647 2147483647 ("##,#", es-ES) -> 2.147.483.647 Bezeichner für Zahlenskalierung: 2147483647 ("#,#,,", en-US) -> 2,147 2147483647 ("#,#,,", es-ES) -> 2.147
"%"	Prozentplatzhalter	Multipliziert eine Zahl mit 100 und fügt ein lokalisiertes Prozentsymbol in die Ergebnisze0.00ichenfolge ein.	0.3697 ("%#0.00", en-US) -> %36.97 0.3697 ("%#0.00", el-GR) -> %36,97 0.3697 ("##.0 %", en-US) -> 37.0 % 0.3697 ("##.0 %", el-GR) -> 37,0 %
"‰"	Promilleplatzhalter	Multipliziert eine Zahl mit 1000 und fügt ein lokalisiertes Promillesymbol in die Ergebniszeichenfolge ein.	0.03697 ("#0.00‰", en-US) -> 36.97‰ 0.03697 ("#0.00‰", ru-RU) -> 36,97‰
"E0" "E+0" "E-0" "e0" "e+0" "e-0"	Exponential-Schreibweise	Formatiert das Ergebnis mit der Exponentialschreibweise, wenn mindestens einmal 0 (null) darauf folgt. Die Groß- oder Kleinschreibung ("E" oder "e") gibt die Schreibweise des Symbols für den Exponenten in der Ergebniszeichenfolge an. Die Anzahl der Nullen, die auf das Zeichen "E" oder auf das Zeichen "e" folgen, bestimmt die Mindestanzahl der Ziffern im Exponenten. Ein Pluszeichen (+) gibt an, dass dem Exponenten immer ein Vorzeichen vorausgeht. Ein Minuszeichen (-) gibt an, dass nur negativen Exponenten ein Vorzeichen vorsteht.	987654 ("#0.0e0") -> 98.8e4 1503.92311 ("0.0##e+00") -> 1.504e+03 1.8901385E-16 ("0.0e+00") -> 1.9e-16
\	Escape Zeichen	Das Zeichen, das auf das Escape Zeichen folgt, wird als Literal und nicht als benutzerdefinierter Formatbezeichner interpretiert.	987654 ("\###00\#") -> #987654#
'Zeichenfolge' "Zeichenfolge"	Zeichenfolgen-literal Trennzeichen	Gibt an, dass die eingeschlossenen Zeichen unverändert in die	68 ("# ' Grad'") -> 68 Grad 68 ("#' Grad'") -> 68 Grad

		Ergebniszeichenfolge kopiert werden sollen.	
;	Abschnittstrenn-zeichen	Definiert Abschnitte mit separaten Formatzeichenfolgen für positive und negative Zahlen sowie Nullen.	12.345 ("#0.0#;(#0.0#);-\0-") -> 12.35 0 ("#0.0#;(#0.0#);-\0-") -> -0- -12.345 ("#0.0#;(#0.0#);-\0-") -> (12.35) 12.345 ("#0.0#;(#0.0#)") -> 12.35 0 ("#0.0#;(#0.0#)") -> 0.0 -12.345 ("#0.0#;(#0.0#)") -> (12.35)

Beispiele

```
<Control Type="Label" Text="Label1" KrlVar="rPercentCurrent"
Format="#.##%"/>
<Control Type="Number" Text="maxForce" KrlVar="rMaxForce" Format="#.00"
Module="Gunkxr"/>
<Control Type="Progressbar" Text="nbrPoints" KrlVar="iNumPoints" Format="#
'pro Tag'" Min="0" Max="3000"/>
<Control Type="Slider" Text="prgBarGunParam" KrlVar="iMaxCutOff" Format="#
'mm'" Min="1" Max="10"/>
```

Darstellung



Einschränkungen der Verwendung beim Steuerelement "Number”

Beim Steuerelement "Number” dürfen nur die Formatierungszeichenfolgen "#" und "0" verwendet werden.

5.3.17.8 Argument "KrlVar"

KRL-Variable dessen Wert im Steuerelement dargestellt werden soll. Erlaubt sind lokale und globale Variablen, verschachtelte Werte und die Angabe von Platzhaltern.

5.3.17.8.1 Angabe von globalen und lokalen Variablen

Die Steuerelemente können an globale oder lokale Variablen gebunden werden.

Globale Variablen

Globale Variablen werden **ohne** Benennung des Moduls, in welchem sie deklariert, angegeben.

Beispiel

```
KrlVar="$OV_PRO"
```

Lokale Variablen

Lokale Variablen werden **mit** Benennung des Moduls, in welchem sie deklariert, angegeben.

Beispiel, Variable "myVariable" deklariert in R1\Program\User\MyDatFile.dat

```
KrlVar="/R1/MyDatFile.dat/MyVariable"
```

→ Der Ordner \Program\User ist nicht anzugeben.

5.3.17.8.2 Verschachtelte Variablen

Variablen können ineinander verschachtelt werden. Bei Aufruf des Plugins werden die Ausdrücke komplett von innen aufgelöst und der Wert aller Variablen überwacht. Bei einer Wertänderung einer inneren Variable wird der komplette Ausdruck neu ermittelt.

Beispiel

```
<Control Text="Base Daten" Type="Text"
KrlVar="Base_Data[myArray[/R1/MyDatFile.dat/MyVariable]].X"/>
```

- Zuerst wird der Wert der lokalen Variable "/R1/MyDatFile.dat/MyVariable" ermittelt → z.B. "5"
- Nun wird mit diesem Wert der Wert des Arrays "myArray[5]" ermittelt → z.B. "2"
- Zuletzt wird der Wert von Base_Data[2].X ermittelt und angezeigt

5.3.17.8.3 Interne Variable als Platzhalter für Zähler

Um noch flexiblere HMI's zu erstellen, ist es möglich, KRL-Arrays mit der Hilfe von zwei oder mehr Steuerelementen darzustellen. Dabei übernimmt ein Steuerelement die Funktion eines Zählers. Dieser dient zur Auswahl des Array-Index und jedes weitere Steuerelement dient der Anzeige des eigentlichen Werts. Pro Seite stehen 32 interne Variablen zur Verfügung.

Zähler werden mit Hilfe eines Platzhalters "%1", "%2", "%3" … - %32" im Argument "KrlVar" angegeben.

Ändert sich der Wert des Zählers, werden alle damit verbunden Steuerelemente automatisch aktualisiert.

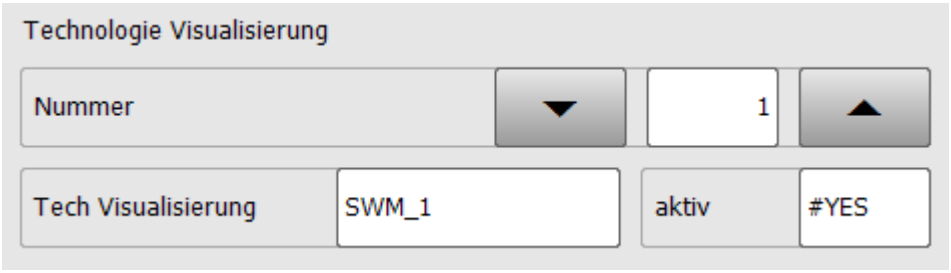
Beispiel

```
<Control Type="Headline" Text="Technologie Visualisierung"/>
<Control Type="Number" Text="Nummer" KrlVar="%1" Step="1" Min="1" Max="10"/>
<Control Type="Text" Text="Tech Visualisierung"
KrlVar="Tech_Visu[%1].Techshortcut[]"/>
<Control Type="Text" Text="aktiv" KrlVar="Tech_Visu[%1].IsActive"/>
```

Deklaration der im Beispiel verwendeten Variable:

```
DECL Technologie Tech_Visu[10]
TechVisu[1]={TechShortcut[] "SWM_1", IsActive #Yes}
```

Anzeige in HMI:



5.3.17.9 Die Argumente "Min" und "Max"

Die Angabe der Argumente "Min" und "Max" hat je nach Steuerelement unterschiedliche Bedeutungen.

Format: **INT**

Zulässige Werte: **10^{-31}** bis **10^{+31}**

Folgende Steuerelemente unterstützen die Angabe dieser Argumente:

- Steuerelement "Number"
- Steuerelement "Progressbar"
- Steuerelement "Slider"

Die Angabe der Min- und Max-Werte kann über KRL-Variablen erfolgen.

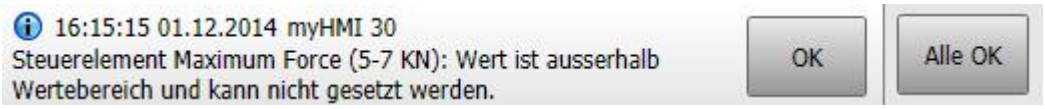
5.3.17.9.1 Verwendung bei Steuerelement "Number"

Das Steuerelement "Number" unterstützt eine Überprüfung der Werteingabe des Bedieners. Dazu können mittels der Argumente "Min" und "Max" die Unter- und Obergrenze, innerhalb der Eingabewert liegen muss, angegeben werden. Ist der Eingabewert außerhalb der Grenzen, wird das Zahlenfeld rot dargestellt und die Übernahme des Eingabewertes ist nicht möglich.

Beispiel

```
<Control Type="Number" Text="Maximum Force (5-7kN)" KrlVar="max_Force"
Min="5" Max="7" ColSpan="2"/>
```

Im Meldungsfenster erscheint eine Hinweismeldung:



5.3.17.9.2 Verwendung bei den Steuerelementen "Slider" und "Progressbar"

Bei beiden Steuerelementen werden die Argumente "Min" und "Max" zur korrekten Darstellung des Füllbalkens bzw. des Schiebereglers verwendet.

Beim Steuerelement "Slider" wird der Hintergrund rot dargestellt, falls der Anzeigewert außerhalb der Grenzen ist.

Beispiel

```
<Control Type="Slider" Text="Maximum Force (5-7kN)" KrlVar="max_Force"
Min="5" Max="7" ColSpan="2"/>
```

5.3.17.10 Argument "Module"

Legt den Namen der Sprachdatenbank für die automatische Sprachumschaltung fest.

Format: string

Zulässige Werte: Namen der Sprachdatenbank (*.kxr)

Das Plugin unterstützt die Verwendung von kxr-Dateien für Mehrsprachigkeit. Bei allen Argumenten vom Typ "string" kann ein Key einer Sprachdatenbank angegeben werden. Ist für die eingestellte HMI-Sprache für den jeweiligen Key eine Übersetzung vorhanden, wird der übersetzte Text in der HMI angezeigt. Wird kein Key in der Datenbank gefunden, wird der angegebene Text angezeigt.

Beispiel:

```
<Control Type="Number" Text="maxForce" KrlVar="xHome1.a1" Module="Servogun"
ColSpan="2"/>
```

Eintrag in der Datei Servogun.kxr

```
    <uiText key="maxForce">
      <text xml:lang="de-DEV">maximale Zangenkraft (5-7 kN)</text>
      <text xml:lang="en-DEV">maximum gun force (5-7 kN)</text>
    </uiText>
```

Anzeige bei HMI-Sprache Deutsch:

Anzeige bei HMI-Sprache Englisch

Die Umschaltung erfolgt zwischen der jeweiligen Sprache erfolgt automatisch.

Wird ein Key in der Datenbank gefunden, aber kein Eintrag für die aktuelle HMI-Sprache, wird der Text der Sprache Englisch dargestellt (falls vorhanden).

Das Argument "Module" kann verwendet werden im Element:

- <Configuration>
- <Group>
- <Control>

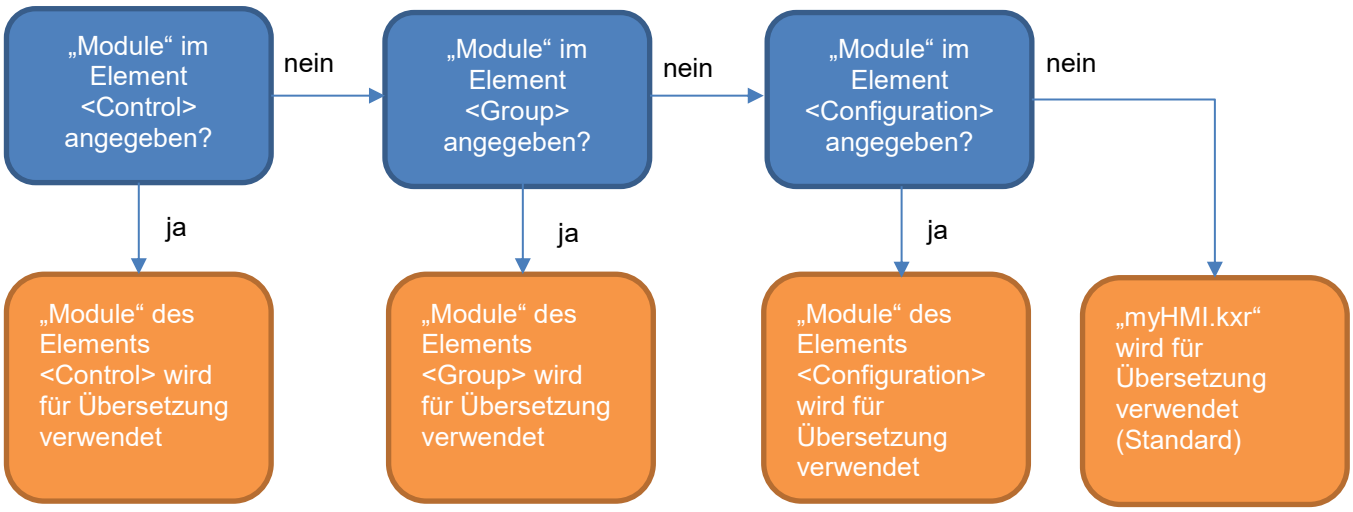
Verwendung und Gültigkeit

Verwendung bei	Gültigkeit
<Control>	Gilt für das jeweilige Steuerelement
<Group>	Gilt für alle Elemente innerhalb dieser Gruppe
<Configuration>	Gilt für alle Elemente aller Gruppen

Erklärung der Verwendung

Wird für ein Steuerelement eine Datenbank angegeben wird diese verwendet. Ist keine Datenbank angegeben, wird die Datenbank des Elements <Group> verwendet. Ist dort ebenfalls keine Datenbank angegeben, wird die Datenbank des Elements <Configuration> verwendet. Ist auch dort das Argument "Module" nicht angegeben, wird standardmäßig die Datenbank "myHMI.kxr" verwendet.

Schematische Darstellung



5.3.17.11 Argument "Mode_OP" → Betriebsart

Definiert die Betriebsart ab welcher ein Steuerelement editierbar ist.

Format: **INT**

Zulässige Werte:

Bezeichnung	Wert	Beispiele
Invalid	16	Standardwert = 31 • Steuerelement ist immer aktiv Beispiel-Wert = 3 • Steuerelement in der Betriebsart "T1" oder "T2" aktiv Beispiel-Wert = 12 • Steuerelement in der Betriebsart "Aut" und "Ext" aktiv
Ext	8	
Aut	4	
T1	2	
T2	1	

5.3.17.12 Argument "NeedDrivesOk" → Antriebe

Definiert welchen Zustand die Antriebe haben müssen, damit ein Steuerelement editierbar ist.

Wert des Argument	Auswirkung
True	Steuerelement ist nur aktiv, wenn die Antriebe eingeschalten sind
False oder nicht angegeben	Steuerelement ist aktiv

5.3.17.13 Argument "Negate"

Negiert den Zustand einer booleschen Variable.

Format: **BOOL**

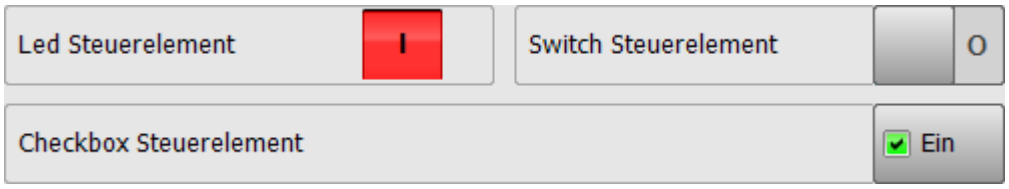
Zulässige Werte

Wert	Auswirkung
True	Der Zustande der Variable wird negiert
False	Der Zustande der Variable wird nicht negiert

Beispiel

```
<Control Type="Led" Text="Led Control" KrlVar="$FLAG[1]" Negate="TRUE"
Color0="Yellow" Color1="Red" Mode_OP="#EX"/>
<Control Type="Switch" Text="Switch Control" KrlVar="$Flag[1]"/>
<Control Type="Checkbox" Text="Checkbox Control" KrlVar="$Flag[1]"
Negate="TRUE" ColSpan="2"/>
```

Anzeige bei $Flag[1]=False



5.3.17.14 Argument "Path"

Legt den Dateinamen und den Pfad zur ladenden Bilddatei beim Steuerelement „Picture“ fest.

Im Dateinamen kann der Platzhalter {0} angegeben werden. Dieser wird dann durch den Wert einer anzugebenden KRL-Variable ersetzt.

Beispiel

```
<Control Type="Picture" Text="Power Laser enabled" Path="D:\IO_{0}.png"
KrlVar="$IN[1]"/>
```

➔ Der Wert von $IN[1] wird als String anstelle des Platzhalter {0} in den Dateinamen eingesetzt.

5.3.17.15 Argument "ProState0" → Submit-Interpreter

Definiert den Zustand des Submit-Interpreters ab welcher ein Steuerelement editierbar ist.

Bezeichnung	Wert	Beispiele
Active	32	Standardwert = 63 • Steuerelement ist immer aktiv Beispiel-Wert = 32 • Steuerelement ist nur aktiv, wenn der Submit läuft Beispiel-Wert = 2 • Steuerelement ist nur aktiv, wenn der Submit abgewählt ist
End	16	
Reset	8	
Stop	4	
Free	2	
Unknown	1	

Beispiel

```
<Control Type="Switch" Text="Switch Control" KrlVar="$FLAG[1]"
Color0="Yellow" Color1="Red" ProState="6"/>
```

→ Steuerelement nur bedienbar, wenn der Submit-Interpreter gestoppt oder abgewählt ist.

5.3.17.16 Argument "ProState1" → Programm-Interpreter

Definiert den Zustand des Programm-Interpreters ab welcher ein Steuerelement editierbar ist.

Bezeichnung	Wert	Beispiele
Active	32	Standardwert = 63 • Steuerelement ist immer aktiv Beispiel-Wert = 30 • Steuerelement ist nur aktiv, wenn der Programmzeiger am Ende steht oder wenn das angewählte Programm zurückgesetzt wurde oder wenn das angewählte Programm gestoppt wurde oder kein Programm angewählt ist • → das Steuerelement ist inaktiv, wenn ein Programm läuft
End	16	
Reset	8	
Stop	4	
Free	2	
Unknown	1	

5.3.17.17 Argument "Step"

Dient zur Festlegung der Schrittweite der Auf/Ab-Tasten des Steuerelements "Number". Nur wenn das Argument "Step" angegeben wird, werden die Auf/Ab-Tasten in der HMI dargestellt.

Format: **REAL**

Zulässige Werte: 10^{-31} bis 10^{+31}

Beispiel

```
<Control Type="Number" Text="Maximum Force (5-7 KN)" KrlVar=" max_Force"
Min="5" Max="5" Step="0.1" ColSpan="2"/>
```

Darstellung

Bei jedem Drücken der Auf/Ab-Taste erhöht/verringert sich der Werte der Variable "max_Force" um den Wert 0.1

Bei Integer-Variablen können keine Real-Werte für Step verwendet werden!

5.3.17.18 Argument "Text"

Dient zur Darstellung des Beschriftungstextes eines Steuerelements.

Format: **string**

Zulässige Zeichen: **String oder Key einer Sprachdatenbank**

Mehrsprachfähig: **ja**

Für jeden String des Arguments "Text" wird nach einem Eintrag in einer Sprachdatenbank gesucht. Wird kein Eintrag gefunden, wird der angegebene String dargestellt. Welche Datenbank verwendet wird, kann mit dem Argument "Module" festgelegt werden. Siehe dazu Kapitel 5.3.17.10.

5.3.17.19 Argument "Text0" und "Text1"

Diese Argumente dienen zur Beschriftung der Schaltfläche des Steuerelements "Checkbox" bei Zustand FALSE und TRUE.

Format: **string**

Zulässige Zeichen: **String oder Key einer Sprachdatenbank**

Mehrsprachfähig: **ja**

Für jeden String des Arguments "Text0" und "Text1" wird nach einem Eintrag in einer Sprachdatenbank gesucht. Wird kein Eintrag gefunden, wird der angegebene String dargestellt. Welche Datenbank verwendet wird, kann mit dem Argument "Module" festgelegt werden. Siehe dazu Kapitel 5.3.17.10.

Verwendung

Argument	Auswirkung	Standardwert wenn Argument nicht angegeben
Text0	Legt die Beschriftung der Schaltfläche beim Zustand FALSE der verknüpften Variable fest	Aus
Text1	Legt die Beschriftung der Schaltfläche beim Zustand TRUE der verknüpften Variable fest	Ein

5.3.17.20 Argument "TextButton"

Dient zur Beschriftung der Schaltfläche des Steuerelements "Button".

Format: **string**

Zulässige Zeichen: **String oder Key einer Sprachdatenbank**

Mehrsprachfähig: **ja**

Standardwert: **-**

Für jeden String des Arguments "TextButton" wird nach einem Eintrag in einer Sprachdatenbank gesucht. Wird kein Eintrag gefunden, wird der angegebene String dargestellt. Welche Datenbank verwendet wird, kann mit dem Argument "Module" festgelegt werden. Siehe dazu Kapitel 5.3.17.10.

Beispiel

```
<Control Type="Button" Text="Button Steuerelement" TextButton="Push"
KrlVar="$Flag[1]" ColSpan="2"/>
```

5.3.17.21 Argument "UserLevelEdit"

Legt fest, **ab** welchem Benutzerlevel eine Registerkarte oder ein Steuerelement editier- bzw. bedienbar ist.

Format: **INT**

Standardwert: **10**

Verwendung und Gültigkeit

Verwendung bei	Gültigkeit
<Control>	Gilt für das jeweilige Steuerelement
<Group>	Gilt für alle Elemente innerhalb dieser Gruppe

Beispiel

```
<Control Type="Switch" Text="Checkbox" KrlVar="$Flag[1]" UserLevelEdit="20"
AreYouSure="True" />
```

Darstellung bei Standardbenutzer

Darstellung bei Benutzer "Experte”

5.3.17.22 Argument "UserLevelVisible”

Legt fest, ab welchem Benutzerlevel eine Registerkarte oder ein Steuerelement sichtbar ist.

Format: **INT**

Standardwert: **10**

5.3.17.23 Benutzerlevel KRC4

Auf der KRC4 werden standardmäßig folgende Benutzerlevel verwendet:

Benutzer	Level
Bediener	10
Experte	20
Sicherheitsinstandhalter	27
Sicherheitsinbetriebnehmer	29
Administrator	30

5.3.17.24 Element "DropDownItem" des Steuerelements "DropDown"

Um einem Dropdown Steuerelement Einträge hinzuzufügen, wird das Element <DropDownItem> verwendet. Innerhalb diesem Element kommen die Argumente "Value" und "Text" zum Einsatz.

5.3.17.24.1 Argument "Value"

Legt den Wert fest, welcher der verbundenen Variable aus dem Argument "KrlVar" übergeben wird.

Format: **abhängig vom Typ der Variable des Elements "KrlVar"**

Standardwert: **-**

Optional: **nein**

5.3.17.24.2 Argument "Text"

Beschreibt den Eintrag im Dropdown-Steuerelement. Falls "Text" nicht angegeben ist, wird als Eintrag der Wert von "Value" festgelegt.

Format: **string oder Key einer Sprachdatenbank**

Standardwert: **-**

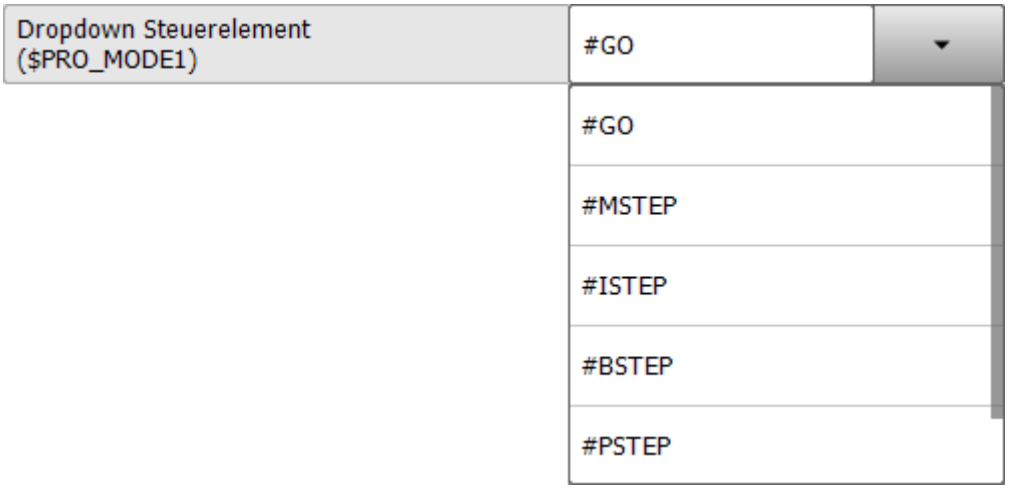
Optional: **ja**

Mehrsprachfähig: **ja**

Beispiel 1, "Text” nicht angegeben

```
<Control Type="Dropdown" Text="Dropdown Steuerelement ($PRO_MODE1)"
KrlVar="$PRO_MODE1">
      <DropDownItem Value="#GO"/>
      <DropDownItem Value="#MSTEP"/>
      <DropDownItem Value="#ISTEP"/>
```

```
      <DropDownItem Value="#BSTEP"/>
      <DropDownItem Value="#PSTEP"/>
      <DropDownItem Value="#CSTEP"/>
</Control>
```



Beispiel 2, "Text" angegeben

```
<Control Type="Dropdown" Text="Dropdown Steuerelement ($PRO_MODE1)"
KrlVar="$PRO_MODE1">
         <DropDownItem Value="#GO" Text="GO"/>
         <DropDownItem Value="#MSTEP" Text="Bewegung"/>
         <DropDownItem Value="#ISTEP" Text="Einzelschritt"/>
         <DropDownItem Value="#BSTEP" Text="Rückwärts"/>
         <DropDownItem Value="#PSTEP" Text="Program Step" />
         <DropDownItem Value="#CSTEP" Text="Continuous Step"/>
</Control>
```

Alle Texteinträge sind als Schlüssel in einem kxr-Modul definiert und werden deshalb übersetzt.

5.3.17.25 Argument "Width”

Legt beim Steuerelement „Picture“ die Breite des Bildes fest. Die Höhe des Bildes wird entsprechend der Proportionen des Originalbildes skaliert.

Format: **INT (Pixel)**

Standardwert: **-**

Beispiel:

```
<Control Type="Picture" Text="Power Laser enabled" Path="D:\IO_{0}.png"
KrlVar="$IN[1]" Width="80"/>
```

➔ Die Grafik wird auf die Breite von 80 Pixel skaliert.

5.4 Layout

Die Steuerelemente werden entsprechend Ihrer Reihenfolge innerhalb der XML-Datei in Spalten geordnet auf Registerkarten dargestellt.

Dabei gelten folgende Einschränkungen:

- Maximal fünf Registerkarten je HMI
- Maximal 32 Steuerelemente, **an die Variablen gebunden sind,** je Registerkarte

Zur Verbesserung der Übersichtlichkeit und der Bedienbarkeit kann das Layout jeder Registerkarte in bis zu drei Spalten unterteilt werden. Dazu wird beim Element <Group> das Argument "Columns" verwendet. Je nach Wert von "Columns" werden die Steuerelemente nebeneinander in Spalten angeordnet. Durch das Argument "ColSpan" können Steuerelemente über mehrere Spalten "gespannt" werden.

Beispiel für drei Spalten

```
<Group Text="3 Spalten" Columns="3">
  <Control Type="Headline" Text="Eingaenge" ColSpan="3"/>
  <Control Type="Led" Text="Eingang 1" KrlVar="$IN[1]"/>
  <Control Type="Led" Text="Eingang 2" KrlVar="$IN[2]"/>
  <Control Type="Led" Text="Eingang 3" KrlVar="$IN[3]"/>
  <Control Type="Led" Text="Eingang 4" KrlVar="$IN[4]"/>
  <Control Type="Led" Text="Eingang 5" KrlVar="$IN[5]"/>
  <Control Type="Led" Text="Eingang 6" KrlVar="$IN[6]"/>
  <Control Type="Label" Text="Ausgaenge" ColSpan="3"/>
  <Control Type="Switch" Text="Ausgang 1" KrlVar="$OUT[1]"
NeedSafetySwitch="TRUE"/>
  <Control Type="Switch" Text="Ausgang 2" KrlVar="$OUT[2]"
NeedSafetySwitch="TRUE"/>
  <Control Type="Switch" Text="Ausgang 3" KrlVar="$OUT[3]"
NeedSafetySwitch="TRUE"/>
  <Control Type="Switch" Text="Ausgang 4" KrlVar="$OUT[4]"
NeedSafetySwitch="TRUE"/>
  <Control Type="Switch" Text="Ausgang 5" KrlVar="$OUT[5]"
NeedSafetySwitch="TRUE"/>
  <Control Type="Switch" Text="Ausgang 6" KrlVar="$OUT[6]"
NeedSafetySwitch="TRUE"/>
  <Control Type="Headline" Text="Flags" ColSpan="3"/>
  <Control Type="Led" Text="Wert von $Flag[1]" KrlVar="$FLAG[1]"
ColSpan="2"/>
  <Control Type="Switch" Text="" KrlVar="$Flag[1]"/>
  <Control Type="Headline" Text="Zeitgeber" ColSpan="3"/>
  <Control Type="Switch" Text="Zeitgeber 1 Stop" KrlVar="$Timer_Stop[1]"
ColSpan="2"/>
  <Control Type="Number" Text="Wert Zeitgeber 1" KrlVar="$Timer[1]"/>
```

```
  <Control Type="Headline" Text="Geschwindigkeiten" ColSpan="3"/>
  <Control Type="Number" Text="Programm Override" KrlVar="$OV_PRO" Min="0"
Max="100"   Step="10" ColSpan="3"/><Control Type="Number" Text="Hand
Override" KrlVar="$OV_JOG" Min="0" Max="100" Step="10" ColSpan="3"/>
</Group>
```

HMI mit 3 Spalten

5.5 Mehrsprachigkeit

Zur mehrsprachigen Darstellung der Texte ist eine Übersetzungsdatei (kxr) notwendig. Die Angabe, welche Übersetzungsdatei verwendet werden soll, erfolgt über das Argument „Module“.

Falls der Name der Übersetzungsdatei gleich dem Namen der xml-Datei ist, kann auf die Angabe durch das Argument „Module“ verzichtet werden. myHMI versucht standardmäßig auf die gleichnamige kxr-Datei zuzugreifen.

Über die Angabe des Arguments "Module” besteht die Möglichkeit Texte mehrsprachig darzustellen. Um Text in die eingestellte HMI Sprache zu übersetzen; sind folgende Voraussetzungen notwendig:

- Es muss eine Übersetzungsdatei (*.kxr) mit entsprechenden "Keys" im Ordner C:\KRC\Data vorhanden sein. Die Übersetzungsdatei muss die Kodierung UTF-8 haben.
- In der XML-Datei für die HMI muss das Argument "Module" auf diese kxr-Datei verweisen (außer die Dateinamen von kxr- und xml-Datei sind identisch)
- Der zu übersetzende Text muss als "Key" in der XML-Datei angegeben werden.
- Es besteht die Möglichkeit verschiedene kxr-Dateien im Argument "Module" anzugeben.

Beispiel 1: Alle Texte des Steuerelements "Button" sollen in die Sprachen Deutsch und Englisch übersetzt werden.

Vorgehensweise:

1. Eine KXR-Datei neu anlegen oder eine bestehende kopieren. Der Name ist frei wählbar.

   Hier: Demo.kxr
2. Beim Steuerelement "Button" im Argument "Module" den Namen der KXR-Datei angeben (nur den Namen ohne die Endung ".kxr")
3. Für das Argument "Text" einen "Key" angeben (string)
4. Für das Argument "TextButton" und, falls verwendet für das Argument "Description" einen Key angeben (string)
5. Einträge in der kxr-Datei vornehmen
6. SmartHMI oder Roboter neu starten um die Einträge in der kxr-Datei dem System bekannt zu machen.

XML-Datei:

```
<Control Type="Button" Text="txtTextBtn" TextButton="txtPushBtn"
Description="descrBtn" Module="Demo" KrlVar="$Flag[1]" ColSpan="2"/>
```

KXR-Datei:

```
<?XML version="1.0" encoding="utf-8"?>
<resources xmlns="http://www.kuka.com/schemas/kxr/2009">
  <module name="Demo">
    <uiText key="txtTextBtn">
      <text xml:lang="de-DEV">Greifer schliessen</text>
      <text xml:lang="en-DEV">close gripper</text>
    </uiText>
    <uiText key="txtPushBtn">
```

```
      <text xml:lang="de-DEV">Schliessen</text>
      <text xml:lang="en-DEV">Close</text>
    </uiText>
    <uiText key="descrBtn">
      <text xml:lang="de-DEV">Beispiel Mehrsprachigkeit</text>
      <text xml:lang="en-DEV">Example multi lingualismn</text>
    </uiText>
  </module>
</resources>
```

Beispiel 2: Alle Texte der HMI sollen in die Sprachen Deutsch und Englisch übersetzt werden.

Vorgehensweise:

1. Eine KXR-Datei neu anlegen oder eine bestehende kopieren. Der Name ist frei wählbar.

   Hier: Demo.kxr
2. Im Element „Configuration“ beim Argument "Module" den Namen der KXR-Datei angeben (nur den Namen ohne die Endung ".kxr")
3. Bei den Steuerelementen im Argument "Text" einen "Key" angeben (string)

XML-Datei:

```
<Configuration Text="Demotext" Module="Demo">
```

KXR-Datei:

```
<?XML version="1.0" encoding="utf-8"?>
<resources xmlns="http://www.kuka.com/schemas/kxr/2009">
  <module name="Demo">
    <uiText key="Demotext">
      <text xml:lang="de-DEV">Text für Demo in deutsch</text>
      <text xml:lang="en-DEV">text for demo in english</text>
    </uiText>
    <uiText key="txtTextBtn">
      <text xml:lang="de-DEV">Greifer schliessen</text>
      <text xml:lang="en-DEV">close gripper</text>
    </uiText>
    <uiText key="txtPushBtn">
      <text xml:lang="de-DEV">Schliessen</text>
      <text xml:lang="en-DEV">Close</text>
    </uiText>
    <uiText key="descrBtn">
      <text xml:lang="de-DEV">Beispiel Mehrsprachigkeit</text>
      <text xml:lang="en-DEV">Example multi lingualismn</text>
    </uiText>
  </module>
</resources>
```

Erklärung der Einträge in der kxr-Datei:

Element / Argument	Beschreibung
<module name>	Hier den Namen der KXR-Datei eintragen. Dieser Eintrag muss identisch mit dem Eintrag im Argument "Module" im Steuerelement Button sein

Key	Korrespondierend mit den Einträgen der Argumente "Text", "Description" und "TextButton" im Steuerelement
<text xml:lang="**de**-DEV">	Deutsche Übersetzung
<text xml:lang="**en**-DEV">	Englische Übersetzung

Verfügbare Sprachen

Elemente	Sprache	Elemente	Sprache
cs	Tschechisch	pl	Polnisch
da	Dänisch	pt	Portugiesisch
de	Deutsch	ro	Rumänisch
en	Englisch	sk	Slowakisch
es	Spanisch	sl	Slowenisch
el	Griechisch	sv	Schwedisch
fi	Finnisch	tr	Türkisch
fr	Französisch	ru	Russisch
it	Italienisch	ko	Koreanisch
hu	Ungarisch	zh	chinesisch
nl	Holländisch	ja	japanisch

Wird ein Key in der Datenbank gefunden, aber kein Eintrag für die aktuelle HMI-Sprache, wird der Text der Sprache Englisch dargestellt (falls vorhanden).

Wird kein Key in der Datenbank gefunden, wird der Eintrag in den Argumenten dargestellt.

Zur Verwendung des Arguments „Module“ Kapitel 5.3.17.10 beachten!

Die KXR-Datei muss die Kodierung „UTF-8“ haben. Wir empfehlen die Verwendung des Editors „Notepad++“

Änderungen in einer KXR-Sprachdatei erfordern einen Neustart des SmartHMI, oder alternativ einen Kaltstart der Roboter-Steuerung.

6 Demo HMI

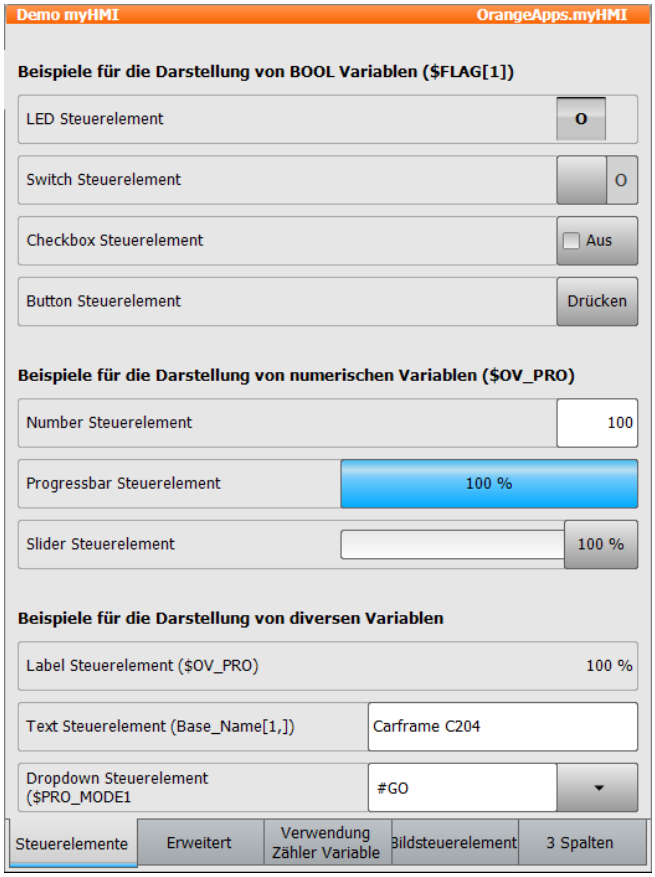
Dem Setup liegt eine HMI bei, welche die Funktionsweise von myHMI darstellt und als Vorlage für neue HMI's dienen kann.

Um die Demo auf dem Roboter anzuzeigen, müssen folgende Schritte durchgeführt werden:

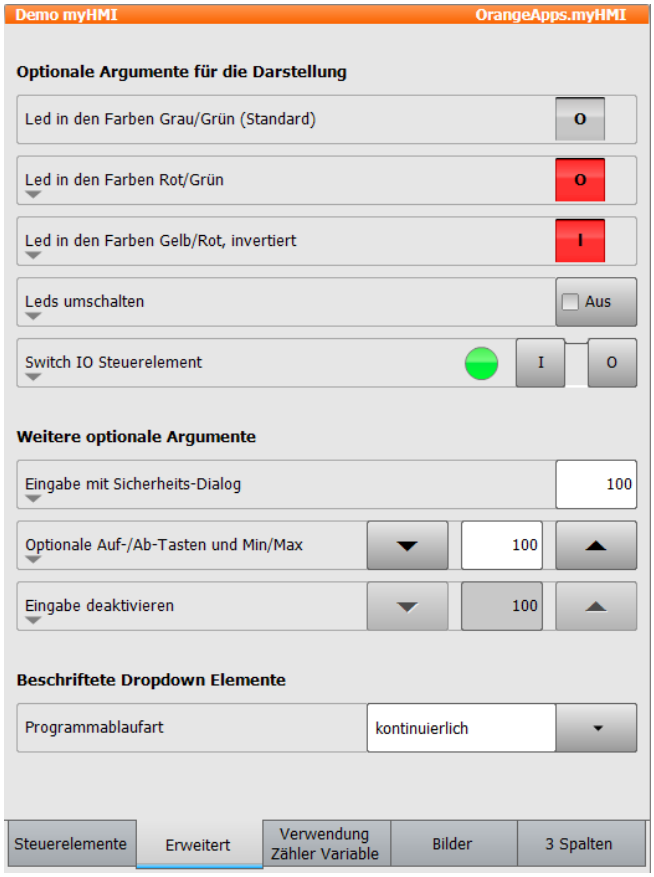
- Menüeintrag mit Menüassistent erstellen. Die HMI ist in der Datei „C:\KRC\USER\DemoMyHMI.xml" definiert.
- Das KRL-Modul „myHMIDemo" installieren (z.B. unter R1 \ Programme). Das Modul kann aus dem Ordner „C:\KRC\USER\myHMI" kopiert werden

6.1 Screenshots

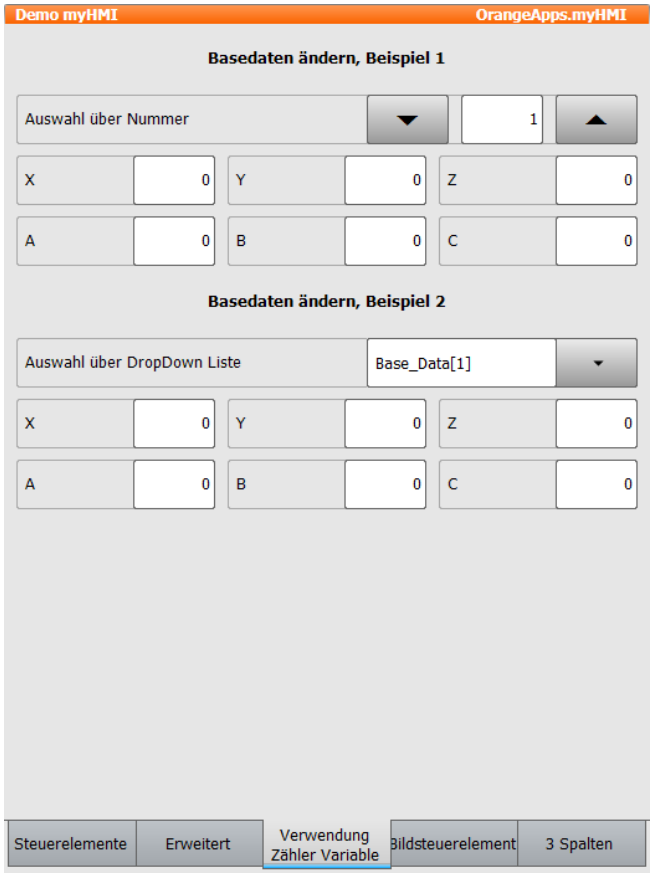
Registerkarte „Steuerelemente"



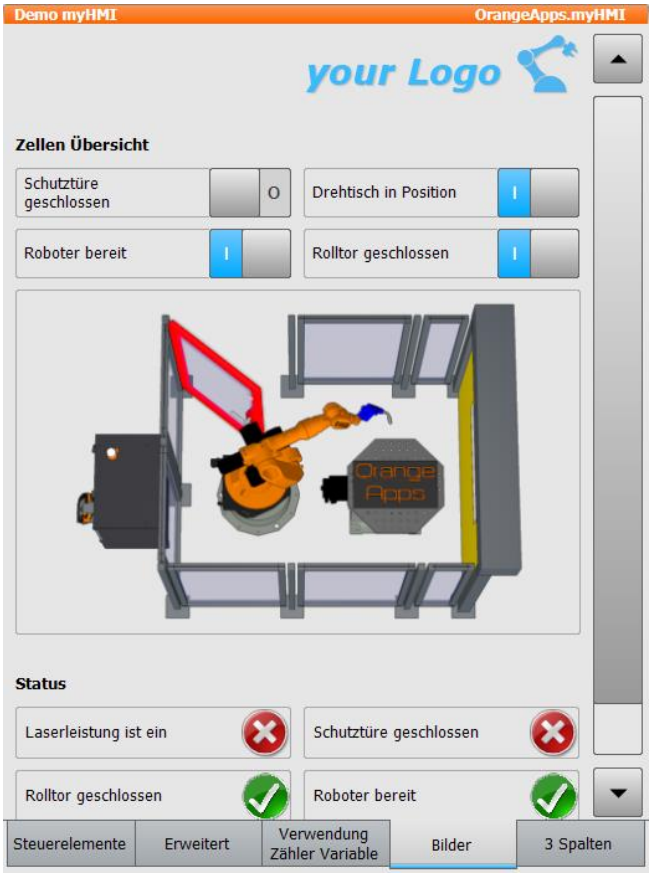
Registerkarte „Erweitert“



Registerkarte „Verwendung Zähler Variable“



Registerkarte „Bilder“



Diese Registerkarte demonstriert die dynamische Darstellung von Grafiken und die überlagerte Darstellung von Grafiken zu einer Gesamtgrafik in Abhängigkeit vom Zustand einer KRL-Variable. Dadurch lassen sich variablengesteuert unterschiedliche Zustände bildlich darstellen.

Mit den Schaltern lassen sich unterschiedliche Anlagenzustände darstellen.

Im Ordner „Program“ befindet sich ein ausführbares Demoprogram „myHMIDemo“ welche diese dynamische Grafikdarstellung aus einem Roboterprogramm veranschaulicht.

Registerkarte „3 Spalten“

7 Eigene HMI erstellen

Folgende Dateien werden benötigt:

- *Name_der_HMI*.xml → Definition der HMI
- (optional) *Name_der_KXR*.kxr → Sprachdatenbank für Übersetzungen

7.1 XML erstellen

Um eine eigene XML Datei zu erstellen kann empfiehlt es sich eine bereits bestehende Datei zu kopieren, z.B. Demo.xml. Die erstellte xml muss im Verzeichnis **C:\KRC\User\myHMI** gespeichert werden.

Zur Bearbeitung der xml-Datei sollte ein geeigneter Editor wie z.B. Notepad ++ verwendet werden.

7.1.1 Grundgerüst erstellen

Das Grundgerüst der XML-Datei bildet das Element <Configuration>. Es beschreibt den Hauptknoten. In ihm befinden sich alle Unterknoten. Über Argumente kann ein Text für die Überschrift der HMI und ein Modul für Übersetzungen mittels KXR-Datei angegeben werden. Wird eine Übersetzungsdatei angegeben, ist diese standardmäßig für alle Elemente gültig.

```
<Configuration Text="myFirstHMI" Module="myFirstHMI">
. . .
</Configuration>
```

Argumente

Argument	Beschreibung	Optional	Standardwert
Text	Beschriftungstext oder Key für die Überschrift	X	
Module	KXR-Datei für multilinguale Inhalte	X	

7.1.2 Gruppen/Registerkarten definieren

Die Gruppen dienen zur thematischen Trennung von Inhalten und werden auf der HMI als Registerkarten angezeigt. Es muss mindestens eine Gruppe angegeben werden, und es können bis zu fünf Gruppen angegeben werden.

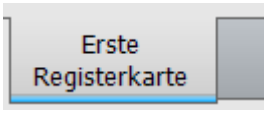
Argumente

Argument	Beschreibung	Optional	Standardwert
Text	Beschriftungstext oder Key für die Registerkarte		
Module	KXR-Datei für multilinguale Inhalte	X	Wert aus <Configuration>
Columns	Anzahl der Spalten 1-3	X	1

Beispiel XML

```
<Configuration Text="myFirstHMI" Module="myFirstHMI">
<Group Text="Erste Registerkarte" Columns=”2” >
  . . .
</Group>
</Configuration>
```

Beispiel Darstellung

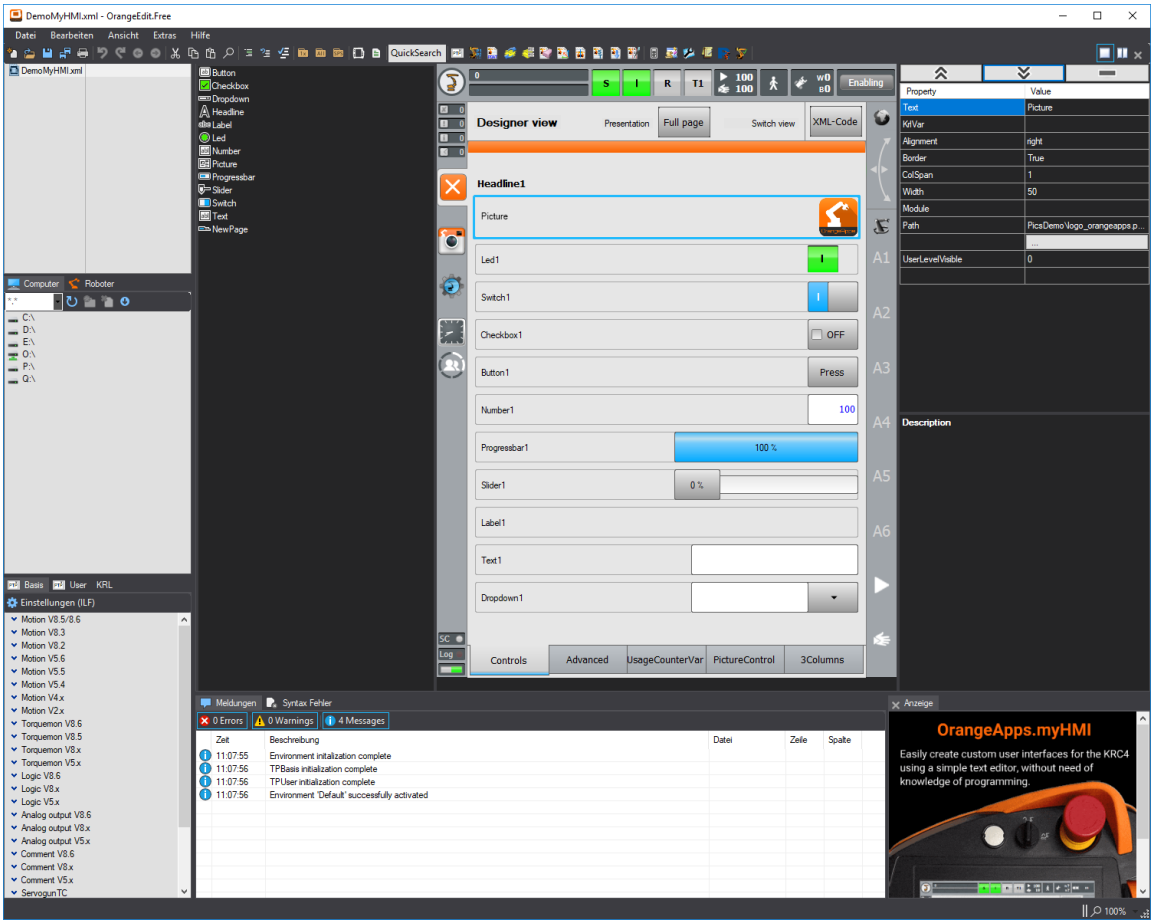


7.1.3 Steuerelemente definieren

In der xml-Datei werden die Steuerelemente entsprechend Kapitel 5.3 definiert.

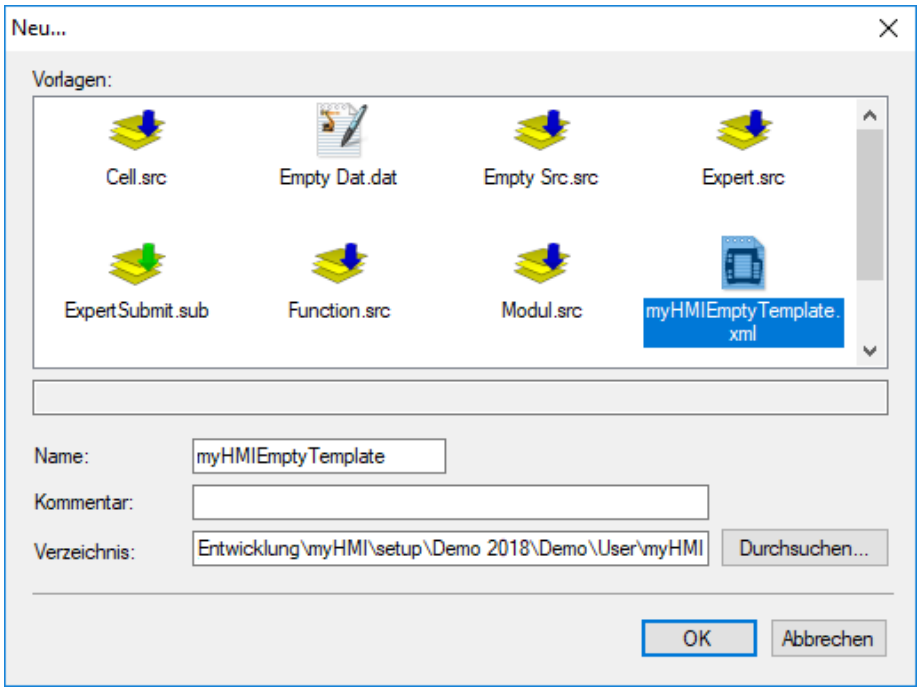
7.2 myHMI-Designer im OrangeEdit

Die Software „OrangeEdit“ (http://orangeapps.de/?lng=de&page=apps%2Forangeedit) verfügt über einen Designer zum einfachen Erstellen bzw. Bearbeiten einer HMI.



7.2.1 HMI neu erstellen

Über die Schaltfläche ***Datei – Neu*** öffnet sich ein Auswahldialog. Dort kann ein leeres Template für eine neue HMI geöffnet werden.

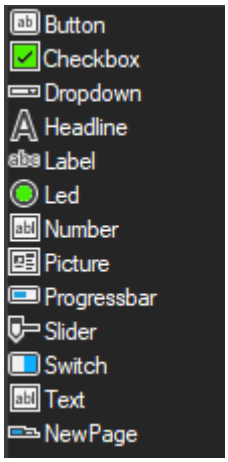


7.2.2 Bestehende HMI öffnen

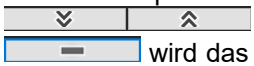
Eine bestehende HMI wird über die Schaltfläche ***Datei – Öffnen*** geöffnet.

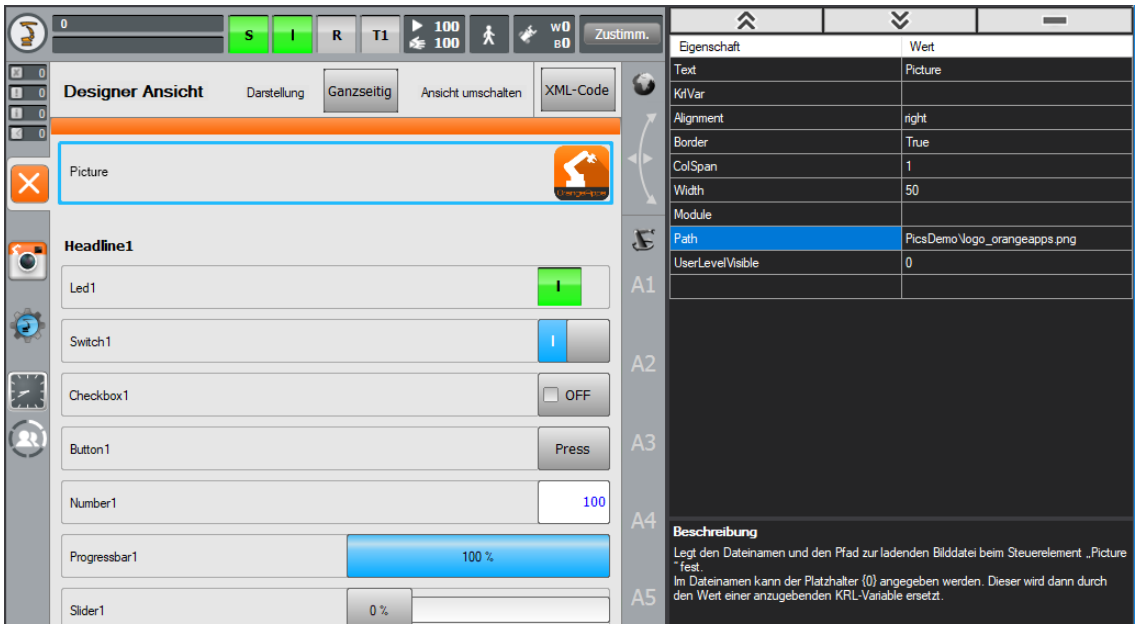
7.2.3 Werkzeugleiste

In der Werkzeugleiste stehen alle derzeit verfügbaren Steuerelemente zur Verfügung. Über Drag & Drop wird ein Steuerelement auf der HMI platziert.



7.2.4 Eigenschaften eines Steuerelements bearbeiten

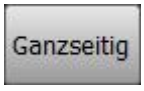
Durch Mausklick wird ein Steuerelement zum Bearbeiten ausgewählt. Im Eigenschaftsfenster werden alle verfügbaren Eigenschaften zum aktiven Steuerelement angezeigt. Dort können diese entsprechend bearbeitet werden. Mit den Auf- und Ab-Schaltflächen  wird das Steuerelement auf der HMI verschoben. Mit der Schaltfläche  wird das Steuerelement entfernt. Das Verschieben und Löschen eines Elements ist auch über das Kontextmenü möglich.



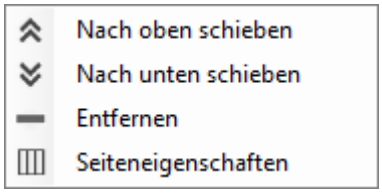
7.2.5 Designerfenster

Im Designerfenster stehen verschiedene Schaltflächen sowie über die rechte Maustaste ein Kontextmenü zur Verfügung.

Schaltflächen

Schaltfläche	Beschreibung
	Schaltet die Anzeige zwischen halb- und ganzseitigem Fenster um
	Blendet zur aktuellen HMI den xml-Code ein. Das aktuell selektierte Steuerelement wird farblich hinterlegt.
	Erstellt einen Screenshot der aktuellen HMI und öffnet einen Speichern-Dialog
	Öffnet ein Menü zur Simulation des aktuell gültigen Benutzerlevels. Nach Auswahl eines Benutzerlevels wird die Darstellung der Steuerelement auf der HMI aktualisiert. 

Kontextmenü (rechte Maustaste)



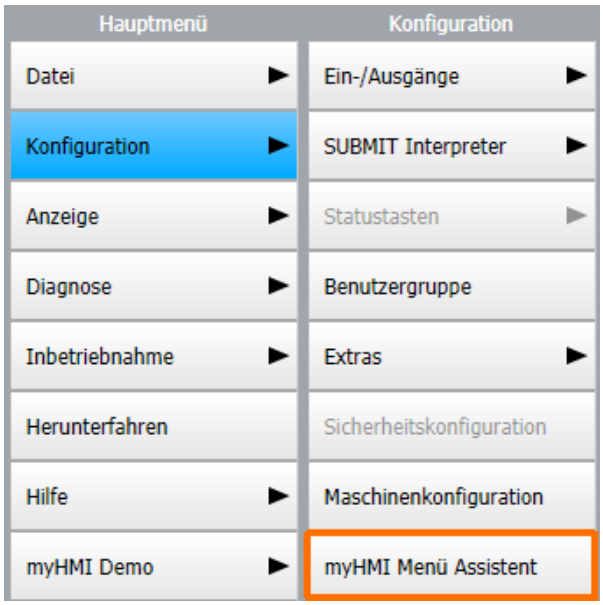
7.2.6 Speichern einer HMI

Nach dem Speichern einer HMI wird eine xml-Datei erzeugt. Diese kann im Roboter gespeichert und zur Anzeige gebracht werden.

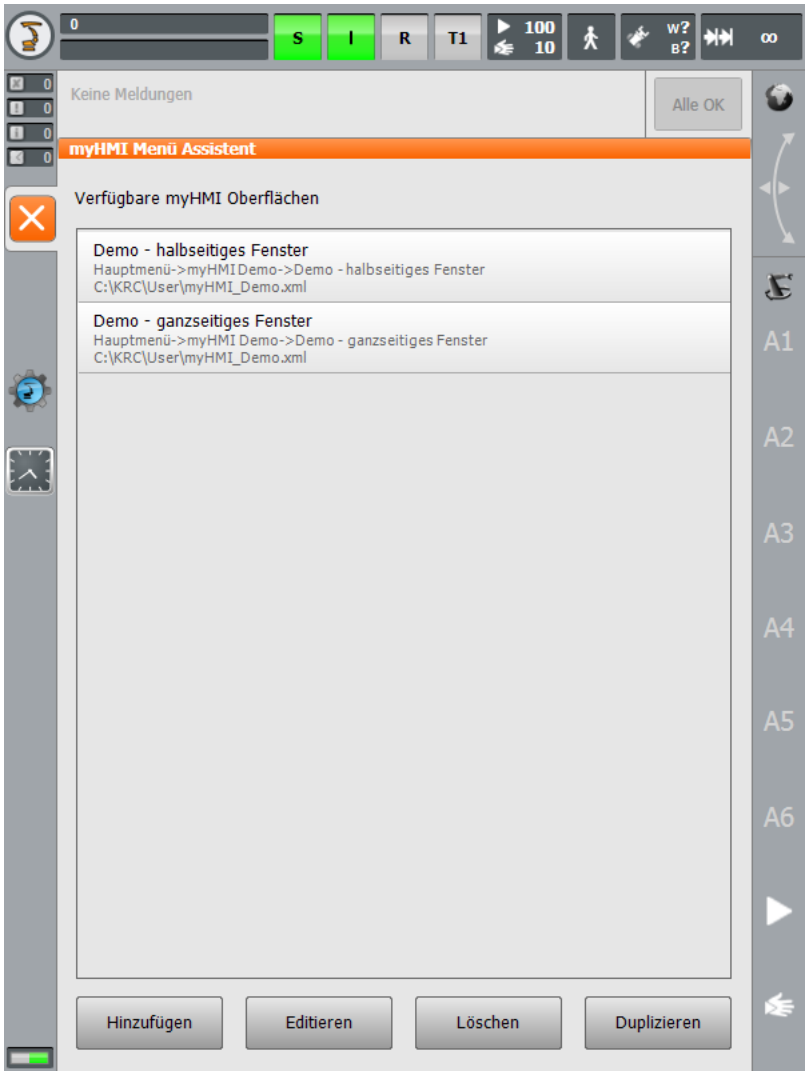
7.3 Menü Assistent - Menüeintrag im KUKA Hauptmenü erzeugen

Um die erstellte HMI aus dem KUKA Hauptmenü aufzurufen, ist ein Eintrag in der Datei SmartHMI.User.config erforderlich. Um die Eintragung zu erleichtern steht ein Assistent zur Verfügung. Dieser befindet sich im Hauptmenü des Roboters im Ordner ***Konfiguration → myHMI Menü Assistent.*** Der benötigte Benutzerlevel zum Starten des Assistenten ist ***Administrator.***

Aufruf des Assistenten



Im Hauptfenster des Assistenten werden die derzeit verfügbaren HMI's tabellarisch angezeigt. Hier können neue Einträge hinzugefügt oder bestehende Einträge editiert, gelöscht oder dupliziert werden.

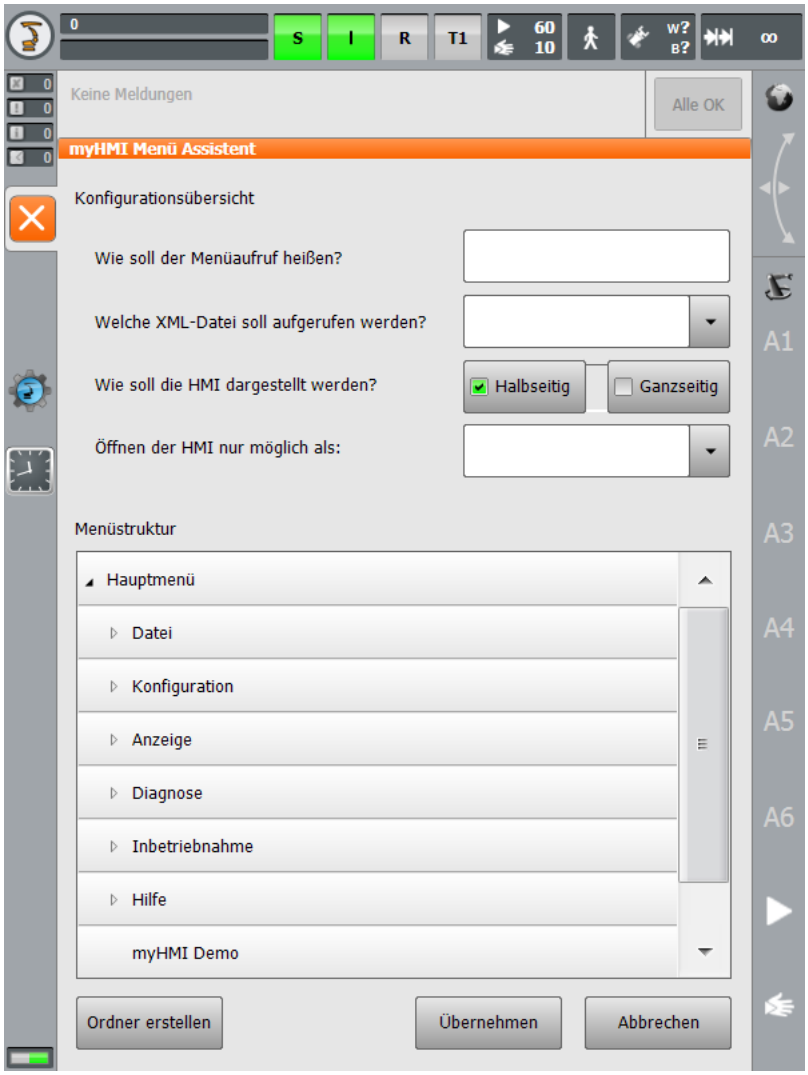


Beschreibung der Schaltflächen

Schaltfläche	Beschreibung
Hinzufügen	Ein neuer Eintrag wird hinzugefügt
Editieren	Der markierte Eintrag kann bearbeitet werden
Löschen	Der markierte Eintrag kann gelöscht werden. Es erfolgt eine Sicherheitsabfrage.
Duplizieren	Der markierte Eintrag wird dupliziert

7.3.1 Eintrag hinzufügen oder editieren

Im folgenden Fenster wird ein Menüeintrag hinzugefügt oder bearbeitet.



In der Auflistung ***Menüstruktur*** kann ausgewählt werden an welcher Stelle im Menü der Eintrag erscheinen soll.

Beschreibung der Elemente

Element	Beschreibung
Wie soll der Menüaufruf heißen?	Name des Eintrags im Hauptmenü
Welche xml-Datei soll aufgerufen werden?	Name der xml-Datei. Im Dropdown-Feld werden alle im Verzeichnis C:\KRC\User\myHMI vorhandenen xml-Dateien aufgelistet.
Wie soll die HMI dargestellt werden?	Die HMI kann als ganz- oder halbseitiges Fenster dargestellt werden.
Öffnen der HMI nur möglich als:	Das Öffnen der HMI kann an den Benutzerlevel gekoppelt werden. Als Level stehen im Dropdown-Feld zur Verfügung: Anwender, Experte,

	Sicherheitsinstandhalter, Sicherheitsinbetriebnehmer, Administrator Wird kein Benutzerlevel ausgewählt, kann die HMI von jedem Benutzer geöffnet werden.

Beschreibung der Schaltflächen

Schaltfläche	Beschreibung
Ordner erstellen	An beliebiger Stelle im Hauptmenü wird ein Unterordner erzeugt
Übernehmen	Die getätigten Einstellungen werden gespeichert
Abbrechen	Die getätigten Einstellungen werden verworfen

7.3.1.1 Mehrsprachigkeit des Namens des Menüeintrages

Analog zu den Übersetzungen der HMI-Elemente, kann über eine Sprachdatenbank der Menüeintrag mehrsprachig gestaltet werden.

Dazu muss der Name des Menüeintrages als Key und der Name der Sprachdatenbank angegeben werden. Die Schreibweise lautet:

Name der Sprachdatenbank **#** *Key*

Beispiel Eintrag: FirstHMI#myFirstHMI

➔ Der Name der KXR Datei in C:\KRC\Data lautet: FirstHMI.kxr
➔ Der Key für die Übersetzung in dieser Datei ist: myFirstHMI

Wenn die kxr-Datei denselben Namen wie die xml-Datei hat (z.B. Demo.kxr und Demo.xml), kann auf die Angabe der kxr-Datei verzichtet werden. Der Menü-Assistent erzeugt automatisch den key Eintrag in der kxr-Datei

Beim Speichern des Eintrags wird die Existenz der kxr-Datei ermittelt und folgende Schritte ausgeführt:

1. Falls die Datei nicht vorhanden ist wird diese erstellt und die Keys mit vordefinierter Übersetzung in der Sprache der HMI und einer zweiten Sprache (entweder Englisch oder Deutsch) eingefügt
2. Falls die Datei bereits vorhanden ist, wird geprüft ob die Keys existieren. Falls nicht, wird er wie unter 1. In die Datei eingefügt.

Text in firstHMI.kxr:

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns="http://www.kuka.com/schemas/kxr/2009">
  <module name="FirstHMI">
    <uiText key="myFirstHMI">
      <text xml:lang="de-DEV">meine erste HMI</text>
      <text xml:lang="en-DEV">my first HMI</text>
    </uiText>
  </module>
</resources>
```

Zur Bedeutung der einzelnen Elemente Kapitel 5.5 beachten

7.4 Beispiel – Erstellen einer HMI mit dem Namen „myFirstHmi“

Am folgenden Beispiel wird eine HMI mit folgenden Eigenschaften erstellt:

- Dateiname der xml: myFirstHMI.xml
- Aufruf der HMI im Menü: Anzeige → Ordner meine erste HMI → meine erste HMI
- Benutzerlevel für Aufruf: Experte
- Größe der HMI: ganzseitig
- Anzahl der Registerkarten: 2
- Alle Texte und Menüaufruf sollen mehrsprachig in den Sprachen englisch und deutsch sein.

Vorgehensweise:

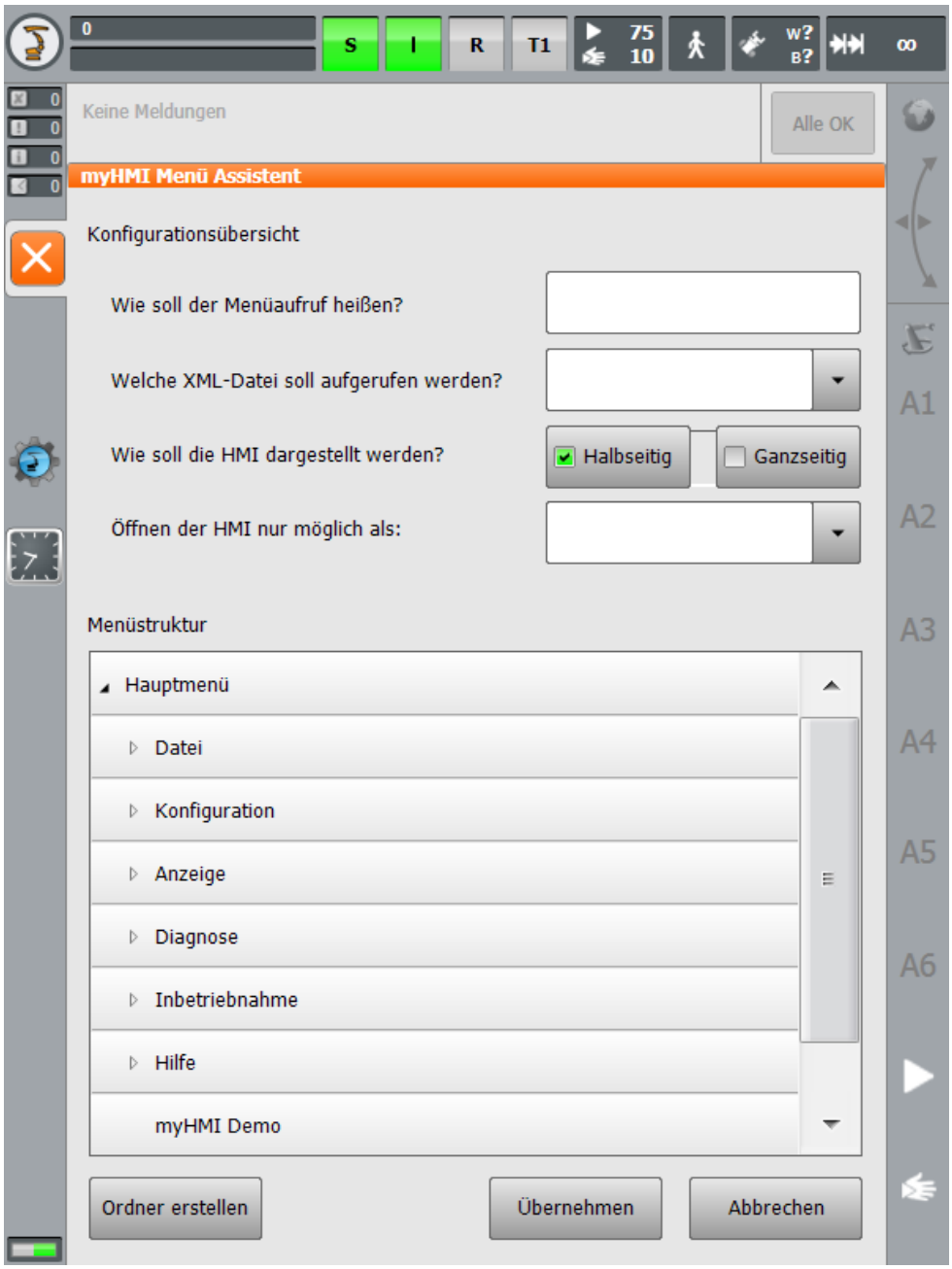
- DemoMyHMI.xml nach myFirstHMI.xml kopieren
- xml-Datei entsprechend den Anforderungen bearbeiten. Dabei Kapitel 5 beachten.
- Menü Assistent öffnen und Eintrag anlegen → die Sprachdatenbank wird, falls noch nicht vorhanden, beim Speichern automatisch im Ordner C:\KRC\Data erzeugt
- Sprachdatenbank bearbeiten
- Roboter mit Kaltstart neu starten um die Sprachdatenbank dem Robotersystem bekannt zu machen

7.4.1 Xml-Datei für HMI, "myFirstHMI.xml"

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Configuration Text="Headline">
     <Group text="Steuerelemente">
           <Control Type="Headline" Text="Headline1"/>
           <Control Type="Led" Text="Led1" KrlVar="$Flag[1]"/>
           <Control Type="Switch" Text="Switch1" KrlVar="$Flag[1]"/>
           <Control Type="Checkbox" Text="Checkbox1" KrlVar="$Flag[1]"/>
           <Control Type="Button" Text="Button1" TextButton="Press"
KrlVar="$Flag[1]"/>
           <Control Type="Headline" Text="Headline2"/>
           <Control Type="Number" Text="Number1" KrlVar="$OV_PRO"/>
           <Control Type="Progressbar" Text="Progressbar1" KrlVar="$OV_PRO"
Format="0 \%" Min="0" Max="100"/>
           <Control Type="Slider" Text="Slider1" KrlVar="$OV_PRO" Format="0
\%" Min="0" Max="100"/>
           <Control Type="Headline" Text="Headline3"/>
           <Control Type="Label" Text="Label1" KrlVar="$OV_PRO" Format="0
\%"/>
           <Control Type="Text" Text="Text1" KrlVar="BASE_NAME[1,]"/>
           <Control Type="Dropdown" Text="Dropdown1" KrlVar="$PRO_MODE1">
                 <DropDownItem Value="#GO" Text="Go"/>
                 <DropDownItem Value="#MSTEP" Text="MStep"/>
                 <DropDownItem Value="#ISTEP" Text="IStep"/>
           </Control>
     </Group>
     <Group Text="3Spalten" Columns="3">
           <Control Type="Headline" Text="Headline4" ColSpan="3"/>
           <Control Type="Led" Text="Eingang1" KrlVar="$FLAG[1]"/>
           <Control Type="Led" Text="Eingang2" KrlVar="$FLAG[2]"/>
           <Control Type="Led" Text="Eingang3" KrlVar="$FLAG[3]"/>
           <Control Type="Led" Text="Eingang4" KrlVar="$FLAG[4]"/>
           <Control Type="Led" Text="Eingang5" KrlVar="$FLAG[5]"/>
           <Control Type="Led" Text="Eingang6" KrlVar="$FLAG[6]"/>
     </Group>
</Configuration>
```

7.4.2 Menü Assistent

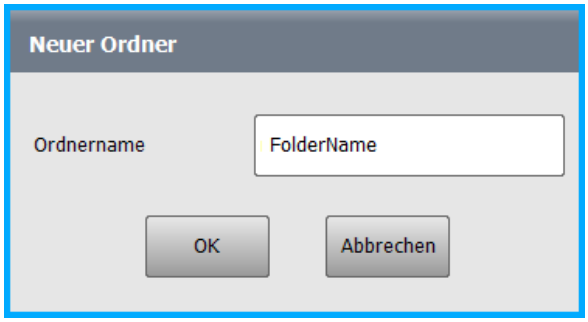
Einloggen als ***Administrator*** und im Hauptmenü unter ***Konfiguration→ myHMI Menü Assistent*** den Assistenten starten. Auf ***Hinzufügen*** klicken. Hier nun alle Einträge erstellen.



1. Ordner für den Menü Eintrag im Hauptmenü anlegen

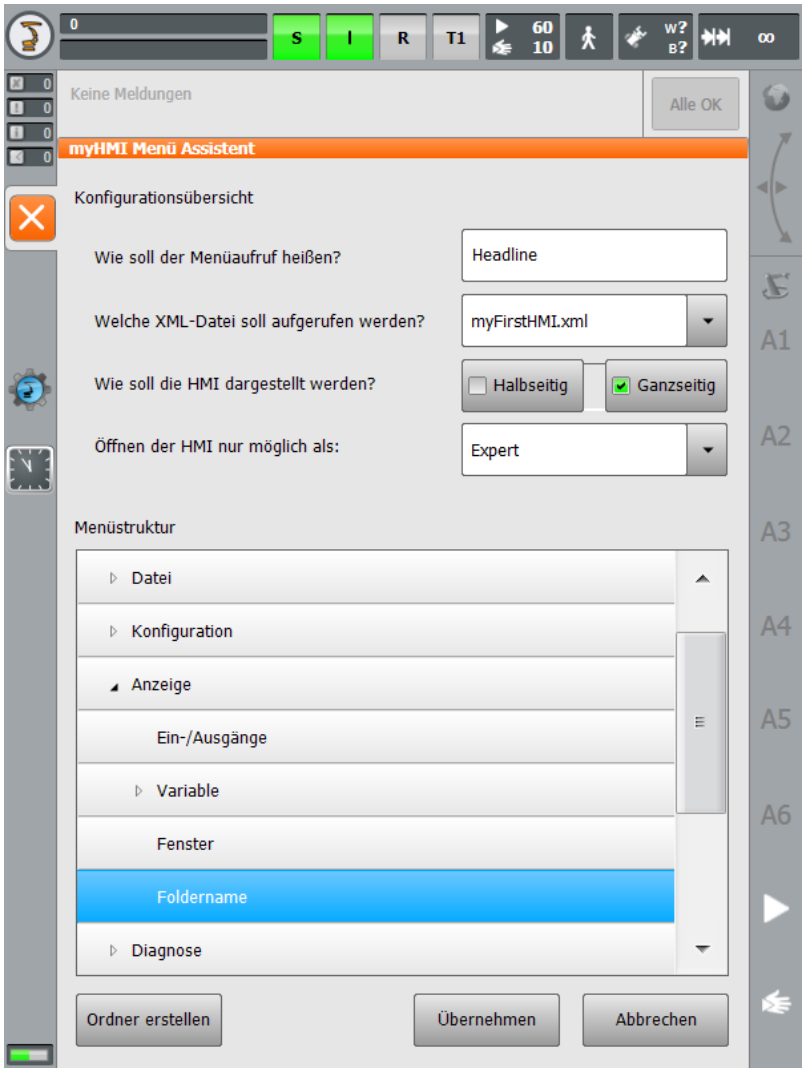
In der Menüstruktur den Ordner ***Anzeige*** öffnen und auf ***Ordner anlegen*** klicken.

Im Textfeld den Key für die Sprachdatenbank angeben (z.B. „Foldername“):



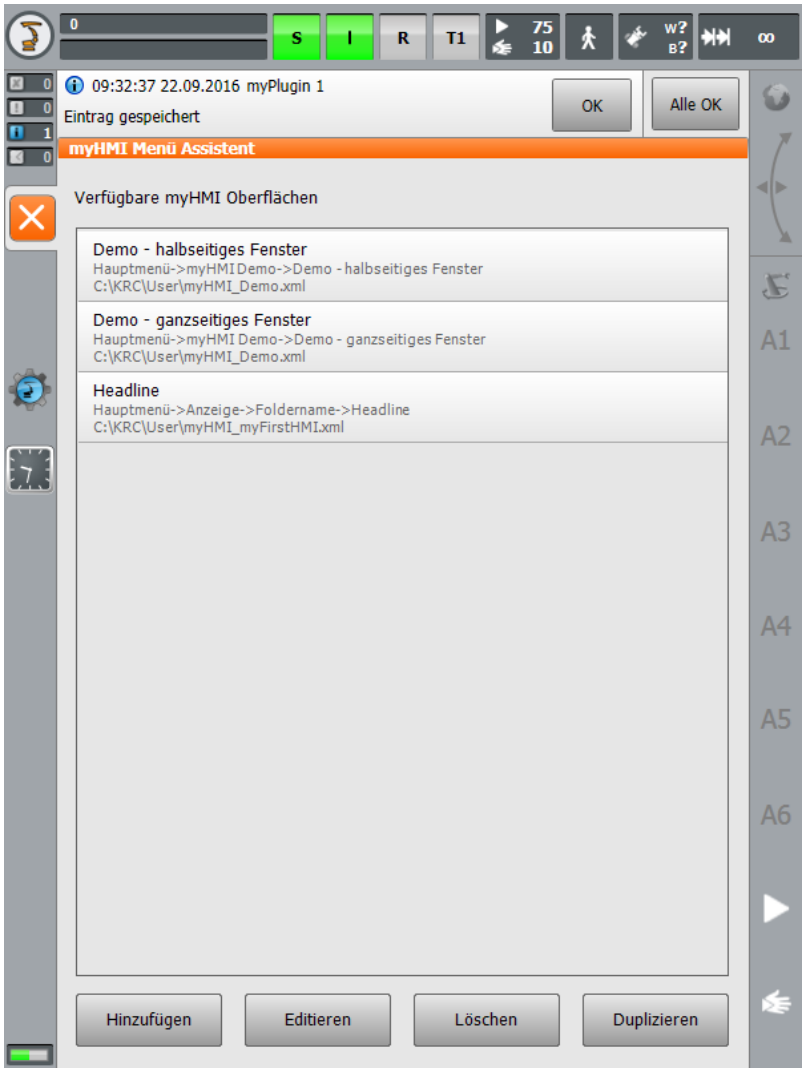
2. Einträge bearbeiten

Im Textfeld für den Menünamen den Key für die Sprachdatenbank angeben (z.B. „Headline“).



Mit ***Übernehmen*** wird der Eintrag gespeichert.

→



Die angegebene Sprachdatenbank „myFirstHMI.kxr“ wurde automatisch erstellt und die beiden Keys ***Headline*** und ***Foldername*** eingefügt. Diese müssen nun noch bearbeitet werden.

3. Sprachdatei "myFirstHMI.kxr" bearbeiten

   Vor der Bearbeitung:

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns="http://www.kuka.com/schemas/kxr/2009">
  <module name="myFirstHMI">
    <uiText key="Headline">
      <text xml:lang="de-DEV">Headline_de</text>
      <text xml:lang="en-DEV">Headline_en</text>
    </uiText>
    <uiText key="Foldername">
      <text xml:lang="de-DEV">Foldername_de</text>
      <text xml:lang="en-DEV">Foldername_de</text>
    </uiText>    </module>
</resources>
```

→ nach der Bearbeitung

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns="http://www.kuka.com/schemas/kxr/2009">
  <module name="myFirstHMI">
    <uiText key="Headline">
      <text xml:lang="de-DEV">meine erste HMI</text>
      <text xml:lang="en-DEV">my first HMI</text>
    </uiText>
    <uiText key="Foldername">
      <text xml:lang="de-DEV">Ordner meine erste HMI</text>
      <text xml:lang="en-DEV">Folder my first HMI</text>
    </uiText>    </module>
</resources>
```

Die Übersetzungen für die HMI werden auch in dieser Datei eingetragen

→

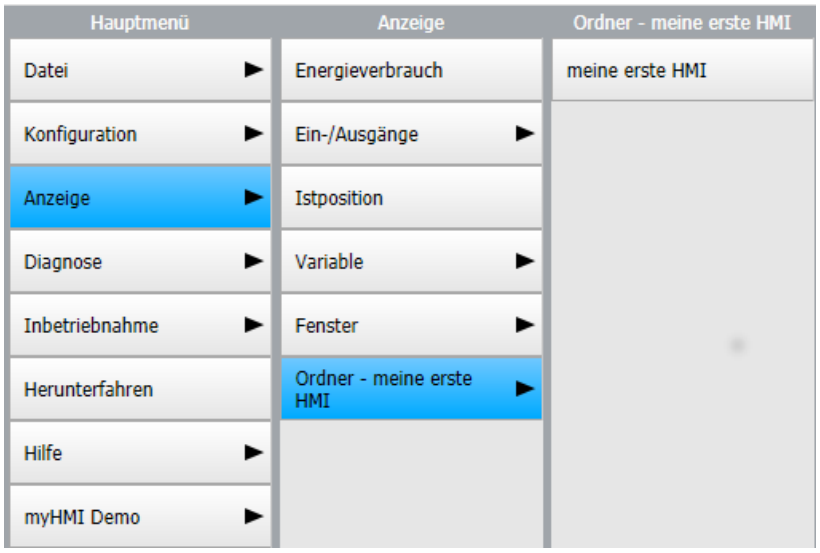
```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns="http://www.kuka.com/schemas/kxr/2009">
   <module name="myFirstHMI">
      <uiText key="Headline">
         <text xml:lang="de-DEV">meine erste HMI</text>
         <text xml:lang="en-DEV">my first HMI</text>
      </uiText>
      <uiText key="Foldername">
         <text xml:lang="de-DEV">Ordner - meine erste HMI</text>
         <text xml:lang="en-DEV">Folder - my first HMI</text>
      </uiText>
      <uiText key="Headline1">
         <text xml:lang="de-DEV">Beispiele für die Darstellung von BOOL
Variablen ($FLAG[1])</text>
         <text xml:lang="en-DEV">Examples for the representation of BOOL
variables ($Flag[1])</text>
      </uiText>
      <uiText key="Steuerelemente">
         <text xml:lang="de-DEV">Steuerelemente</text>
         <text xml:lang="en-DEV">Controls</text>
      </uiText>
      <uiText key="LED1">
         <text xml:lang="de-DEV">LED Steuerelement</text>
         <text xml:lang="en-DEV">LED control</text>
      </uiText>
      <uiText key="Switch1">
         <text xml:lang="de-DEV">Switch Steuerelement</text>
         <text xml:lang="en-DEV">Switch control</text>
      </uiText>
      <uiText key="CheckBox1">
         <text xml:lang="de-DEV">Checkbox Steuerelement</text>
         <text xml:lang="en-DEV">Checkbox control</text>
      </uiText>
      <uiText key="Button1">
         <text xml:lang="de-DEV">Button Steuerelement</text>
         <text xml:lang="en-DEV">Button control</text>
      </uiText>
      <uiText key="Press">
         <text xml:lang="de-DEV">Drücken</text>
         <text xml:lang="en-DEV">Press</text>
      </uiText>
      <uiText key="Headline2">
         <text xml:lang="de-DEV">Beispiele für die Darstellung von
numerischen Variablen ($OV_PRO)</text>
         <text xml:lang="en-DEV">Examples for the representation of Number
variables ($OV_PRO)</text>
      </uiText>
      <uiText key="Number1">
         <text xml:lang="de-DEV">Number Steuerelement</text>
         <text xml:lang="en-DEV">Number control</text>
      </uiText>
```

```
      <uiText key="Progressbar1">
         <text xml:lang="de-DEV">Progressbar Steuerelement</text>
         <text xml:lang="en-DEV">Progressbar control</text>
      </uiText>
      <uiText key="Slider1">
         <text xml:lang="de-DEV">Slider Steuerelement</text>
         <text xml:lang="en-DEV">Slider control</text>
      </uiText>
      <uiText key="Headline3">
         <text xml:lang="de-DEV">Beispiele für die Darstellung von diversen
Variablen</text>
         <text xml:lang="en-DEV">Examples for the representation of
miscellaneous variables</text>
      </uiText>
      <uiText key="Label1">
         <text xml:lang="de-DEV">Label Steuerelement ($OV_PRO)</text>
         <text xml:lang="en-DEV">Label control ($OV_PRO)</text>
      </uiText>
      <uiText key="Text1">
         <text xml:lang="de-DEV">Text Steuerelement (Base_Name[1,])</text>
         <text xml:lang="en-DEV">Text control (Base_Name[1,])</text>
      </uiText>
      <uiText key="Dropdown1">
         <text xml:lang="de-DEV">Dropdown Steuerelement ($PRO_MODE1</text>
         <text xml:lang="en-DEV">Dropdown control ($PRO_MODE1)</text>
      </uiText>
      <uiText key="Go">
         <text xml:lang="de-DEV">kontinuierlich</text>
         <text xml:lang="en-DEV">continous</text>
      </uiText>
      <uiText key="MStep">
         <text xml:lang="de-DEV">Bewegung</text>
         <text xml:lang="en-DEV">motion</text>
      </uiText>
      <uiText key="IStep">
         <text xml:lang="de-DEV">Einzelschritt</text>
         <text xml:lang="en-DEV">single step</text>
      </uiText>
      <uiText key="3Spalten">
         <text xml:lang="de-DEV">Darstellung in 3 Spalten</text>
         <text xml:lang="en-DEV">Representation in 3 columns</text>
      </uiText>
      <uiText key="Headline4">
         <text xml:lang="de-DEV">Eingänge 1-6</text>
         <text xml:lang="en-DEV">Input 1-6</text>
      </uiText>
      <uiText key="Eingang1">
         <text xml:lang="de-DEV">Eingang 1</text>
         <text xml:lang="en-DEV">Input 1</text>
      </uiText>
      <uiText key="Eingang2">
         <text xml:lang="de-DEV">Eingang 2</text>
         <text xml:lang="en-DEV">Input 2</text>
      </uiText>
      <uiText key="Eingang3">
         <text xml:lang="de-DEV">Eingang 3</text>
         <text xml:lang="en-DEV">Input 3</text>
      </uiText>
      <uiText key="Eingang4">
         <text xml:lang="de-DEV">Eingang 4</text>
         <text xml:lang="en-DEV">Input 4</text>
      </uiText>
      <uiText key="Eingang5">
         <text xml:lang="de-DEV">Eingang 5</text>
         <text xml:lang="en-DEV">Input 5</text>
      </uiText>
      <uiText key="Eingang6">
         <text xml:lang="de-DEV">Eingang 6</text>
         <text xml:lang="en-DEV">Input 6</text>
      </uiText>
```

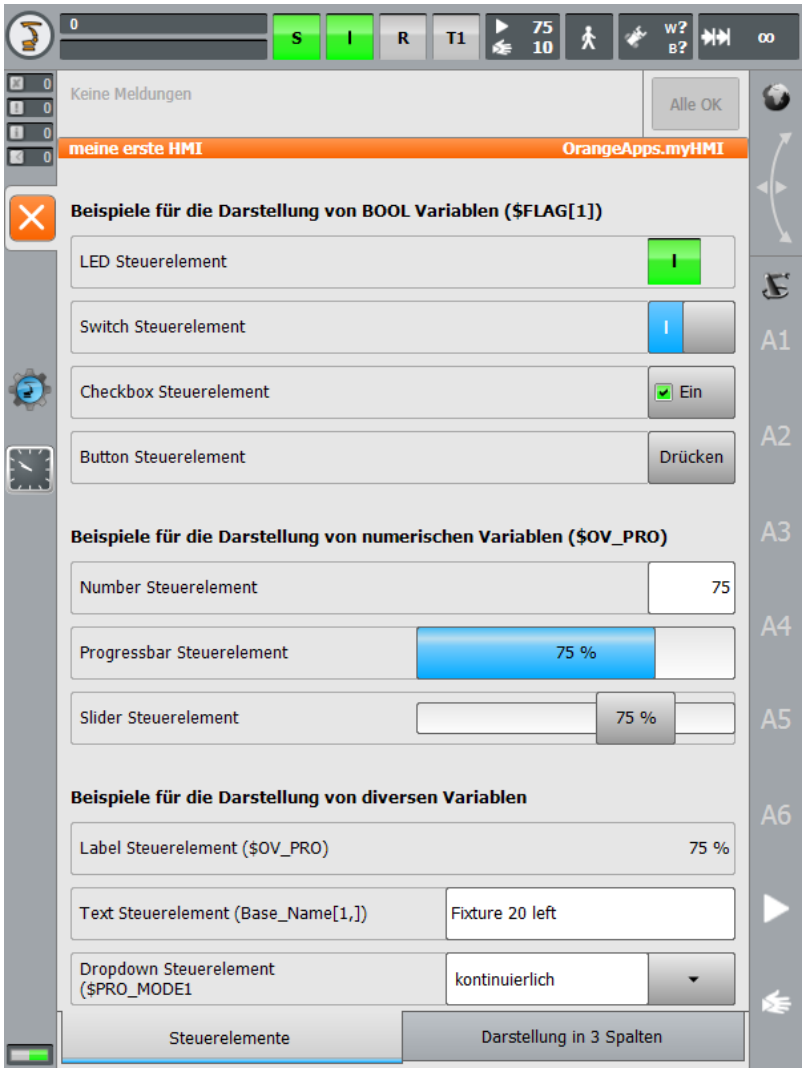
```
   </module>
</resources>
```

Damit die Eintragungen in der Sprachdatenbank wirksam werden, muss der Roboter neu gestartet werden.

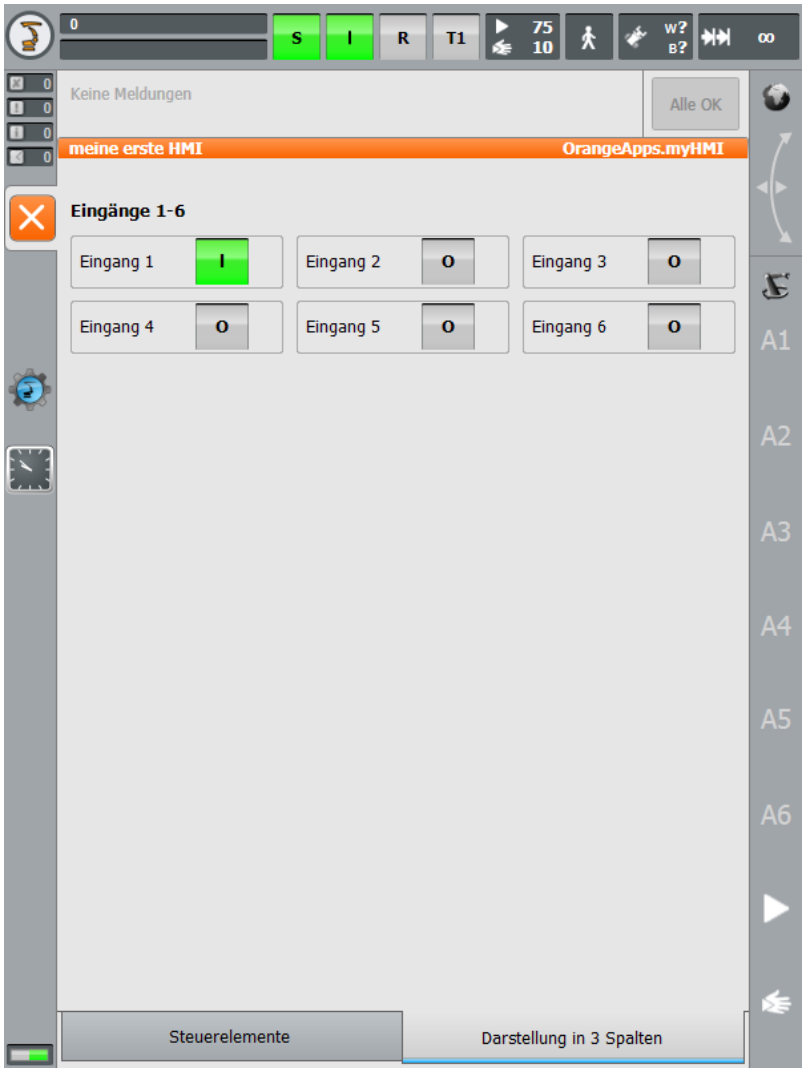
Dargestellter Menüeintrag im Hauptmenü



HMI – Registerkarte 1



HMI – Registerkarte 2



8 HMI’s automatisiert öffnen und schließen

HMI’s können aus KRL Modulen gezielt geöffnet und geschlossen werden. Zudem können HMI’s nach dem Hochlauf der Steuerung und beim Wechsel der Betriebsart oder des Benutzers angezeigt werden.

8.1 KRL Funktionen zum Öffnen und Schließen von HMI’s

8.1.1 Funktion MyHmiOpen

Öffnet eine HMI aus KRL heraus.

Funktionsaufruf

MyHMIOpen(char „**Name der HMI“**, enum **ViewMode**, int **Tabnummer**)

Parameter

- View [char], Pfad zur MyHMI.xml Datei z.B. „MyHmiDemo.xml“
- ViewMode [enum], optional, „#Half“ für halbseitiges Fenster, „#Full“ für ganze Seite (Default=#Full)
- Tab [int], optional, Reiter der HMI, welcher geöffnet werden soll (Default=1)

Beispiel

```
MyHmiOpen("myHmiDemo.xml", #Full, 2)
```

Öffnet die HMI ganzseitig auf Reiter 2

8.1.2 Funktion MyHmiClose

Schließt eine HMI aus KRL heraus.

Funktionsaufruf

myHMIClose(char „**Name der HMI“**)

Parameter

- View [char], optional, Pfad zur MyHMI.xml Datei z.B. „MyHmiDemo.xml“ (Default= *)

Beispiel

```
MyHmiClose("myHmiDemo.xml")
```

Schließt die HMI „myHmiDemo.xml“

Beispiel

```
MyHmiClose()
```

Schließt alle geöffneten HMI‘s

8.1.3 Funktion MyHmiIsOpen

Gibt zurück, ob eine HMI geöffnet ist

Funktionsaufruf

MyHmiIsOpen(char „**Name der HMI“**)

Parameter

- View [char], Pfad zur MyHMI.xml Datei z.B. „MyHmiDemo.xml“ (optional: Default *)

Rückgabewert

TRUE oder FALSE, TRUE=angegeben HMI ist geöffnet

Beispiel

```
IF MyHmiIsOpen("myHmiDemo.xml") THEN
  ;HMI ist offen
ELSE
  ;HMI ist nicht offen
ENDIF
```

Gibt zurück ob die HMI „myHmiDemo.xml“ geöffnet ist

8.2 KRL Variablen für bedingtes Öffnen von HMI's

Mit den folgenden Variablen können HMI's zu bestimmten Bedingungen geöffnet werden. Die Variablen befinden sich in der Datei „R1\TP\myHMI_User.dat“.

8.2.1 HMI nach Hochlauf öffnen (Autostart)

Durch Beschreiben der folgende Variable, kann eine HMI nach dem Hochlauf der Steuerung geöffnet werden (Autostart)

Variable

OpenOnStartup(char **Name der HMI,** enum **ViewMode**,int **Tabnummer**)

Parameter

- View [char], Pfad zur MyHMI.xml Datei z.B. „MyHmiDemo.xml“
- ViewMode [enum], optional, „#Half“ für halbseitiges Fenster, „#Full“ für ganze Seite (Default=#Full)
- Tab [int], optional, Reiter der HMI, welcher geöffnet werden soll (Default=1)

Beispiel

Autostart deaktiviert, da keine HMI (XML-Datei) angegeben wurde

```
DECL myHmi_S OpenOnStartup={View[] " ",ViewMode #Full,Tab 0}
```

Beispiel

Autostart aktiv, nach Hochlauf wird die HMI „myHmiDemo.xml" ganzseitig mit Reiter 3 geöffnet

```
DECL myHmi_S OpenOnStartup={View[] "myHmiDemo.xml",ViewMode #Full,Tab 3}
```

8.2.2 HMI bei Betriebsarten-Wechsel und/oder Benutzer-Wechsel öffnen

Durch Beschreiben der Variablen **OpenOnCondition** kann eine HMI nach einem Benutzer- oder Betriebsarten-Wechsel geöffnet werden.

Die Angabe der Betriebsarten und der Benutzerlevel ist optional und erfolgt als Charakter Variablen. Die Werte werden dabei kommagetrennt angegeben.

OpenOnCondition ist eine Variable mit 10 Feldern (**OpenOnCondition[10]).** Somit können bis zu 10 verschiedene Bedingungen konfiguriert werden. Die erste zutreffende Bedingung wird erkannt und die entsprechende HMI geöffnet.

Variable

OpenOnCondition[Nummer des Array] = (char **Name der HMI,** enum **ViewMode**, int **Tabnummer,** char **ModeOps**,char **UserModes**)

Parameter

- View [char], Pfad zur MyHMI.xml Datei z.B. „MyHmiDemo.xml"
- ViewMode [enum], optional, „#Half" für halbseitiges Fenster, „#Full" für ganze Seite (Default=#Full)
- Tab [int], optional, Reiter der HMI, welcher geöffnet werden soll (Default=1)
- ModeOps [char], optional, Betriebsarten T1, T2, AUT, EXT (Default= *)
- UserModes [char], optional, Benutzerlevel 5,10,15,20,29,30 (Default= *)

Beispiel

Bei jedem Benutzer- oder Betriebsarten-Wechsel wird „myHmiDemo" geöffnet

```
OpenOnCondition[1]={View[] "myHmiDemo.xml",ViewMode #Full,Tab 0,ModeOps[] "
",UserModes[] " "}
```

Beispiel

Nur wenn Betriebsart „EXT", oder „AUT" und Benutzer-Level 5 oder 10 aktiv ist wird „myHmiDemo" ganzseitig mit Reiter 2 geöffnet

```
OpenOnCondition[1]={View[] "myHmiDemo.xml",ViewMode #Full,Tab 2,ModeOps[]
"EXT,AUT",UserModes[] "5,10"}
```

Betriebsarten

- T1 – Test 1
- T2 – Test 2
- AUT – Automatik
- EXT – Extern

Benutzer-Level

- 5 – Standard
- 10 – Bediener
- 20 – Expert
- 27 – Sicherheits-Instandhalter
- 29 – Sicherheits-Inbetriebnehmer
- 30 – Administrator

9 Anhang

9.1 Meldungen von myHMI

Meldung	Beschreibung
Fehler in XML-Datei	Ein allgemeiner Fehler beim Lesen der Datei ist aufgetreten
Keine Gruppen in XML-Datei definiert	Es sind keine Gruppen in der XML Datei konfiguriert
Fehler in XML-Datei, Gruppe x	In der Gruppe x ist ein allgemeiner Fehler in der XML Beschreibung
Fehler in XML-Datei, Steuerelement x	Beim Steuerelement x ist ein allgemeiner Fehler in der XML Beschreibung
Fehler XML-Datei, ein Steuerelement hat keinen Titel	Ein Steuerelement in der XML-Datei hat kein "Title" Attribut
Fehler XML-Datei, für Steuerelement x ist keine KRL Variable definiert	Das Steuerelement x in der XML-Datei hat kein "KrlVar" Attribut
Fehler XML-Datei, für Steuerelement x ist kein Typ (int, real) definiert	Das Steuerelement x in der XML-Datei hat kein gültiges "Type" Attribut
Steuerelement x, Variable y nicht vorhanden	Die KRL Variable y ist nicht vorhanden
Steuerelement x, allgemeiner Fehler	Das Steuerelement x verursachte eine allgemeine Ausnahme
Steuerelement x, Wert konnte nicht gesetzt werden	Der eingegebene Wert konnte nicht auf die KRL Variable geschrieben werden
Fehler beim Laden der Bilddatei. Bildformat wird nicht unterstützt.	Die Bilddatei hat ein falsches Format.
Fehler beim Laden der Bilddatei.	Beim Laden der Bilddatei ist ein unbekannter Fehler aufgetreten.

9.2 Lizenzmeldungen

Meldung	Beschreibung
Dialog-Meldung: Lizenz für das Produkt myHMI ungültig oder abgelaufen. Kontaktieren sie ihren Systemintegrator.	Die Lizenz-Datei ist ungültig z.B. falsche Seriennummer, oder eine laufzeitbegrenzte Lizenz ist abgelaufen. Eine neue Lizenz-Datei behebt das Problem.
Dialog-Meldung: Keine Lizenz für das Produkt myHMI vorhanden. Kontaktieren sie ihren Systemintegrator.	Es ist keine Lizenz-Datei auf dem System. Eine neue Lizenz-Datei behebt das Problem.

Status-Meldung: X Tage verbleibend bis Lizenz abläuft	Wird bei laufzeitbegrenzten Lizenzen angezeigt, wenn verbleibender Zeitraum < 15 Tage ist.
Status-Meldung: Keine Lizenz-Datei für Roboter X vorhanden	Keine Lizenz-Datei für den Roboter mit der Serien-Nummer X (X=Seriennummer). Eine neue Lizenz-Datei behebt das Problem.
Status-Meldung: Lizenz für Roboter X ungültig oder abgelaufen	Keine gültige Lizenz für den Roboter mit der Serien-Nummer X (X=Seriennummer). Eine neue Lizenz-Datei behebt das Problem.
Info-Meldung: Datum wurde manipuliert, Lizenz wurde zurückgesetzt!	Bei Verwendung einer laufzeitbegrenzten Lizenz wurde erkannt, dass das Datum des Roboter-Systems verändert wurde. Die Lizenz wird ungültig. Eine neue Lizenz-Datei behebt das Problem.