SIEMENS

SINUMERIK

SINUMERIK 828D
SINUMERIK Integrate Run
MyScreens

Programmierhandbuch

Gültig für

CNC-Software Version 4.95

07/2021
6FC5398-4EP40-0AA0

Rechtliche Hinweise

Warnhinweiskonzept

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

⚠ **GEFAHR**
bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠ **WARNUNG**
bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠ **VORSICHT**
bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

ACHTUNG
bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung **qualifiziertem Personal** gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

Bestimmungsgemäßer Gebrauch von Siemens-Produkten

Beachten Sie Folgendes:

⚠ **WARNUNG**
Siemens-Produkte dürfen nur für die im Katalog und in der zugehörigen technischen Dokumentation vorgesehenen Einsatzfälle verwendet werden. Falls Fremdprodukte und -komponenten zum Einsatz kommen, müssen diese von Siemens empfohlen bzw. zugelassen sein. Der einwandfreie und sichere Betrieb der Produkte setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung, Montage, Installation, Inbetriebnahme, Bedienung und Instandhaltung voraus. Die zulässigen Umgebungsbedingungen müssen eingehalten werden. Hinweise in den zugehörigen Dokumentationen müssen beachtet werden.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
DEUTSCHLAND

Dokumentbestellnummer: 6FC5398-4EP40-0AA0
Ⓟ 07/2021 Änderungen vorbehalten

Inhaltsverzeichnis

Einleitung 1

1.1 Über SINUMERIK

Von einfachen CNC-Standardmaschinen über standardisierte Maschinen, bis hin zu modularen Premium-Maschinenkonzepten – die CNC-Steuerungen SINUMERIK bieten für jedes Maschinenkonzept die passende Lösung. Ob Einzelteil- oder Massenfertigung, einfache oder komplexe Werkstücke – SINUMERIK ist die hochproduktive Automatisierungslösung durchgängig für alle Fertigungsbereiche – vom Muster- und Werkzeugbau über den Formenbau bis zur Großserienfertigung.

Für weitere Informationen besuchen Sie die Internetseite zu SINUMERIK (https://www.siemens.de/sinumerik).

1.2 Über diese Dokumentation

Zielgruppe

Die vorliegende Druckschrift wendet sich an Programmierer und Projekteure.

Nutzen

Das Programmierhandbuch befähigt die Zielgruppe, Programme und Software-Oberflächen zu entwerfen, zu schreiben, zu testen und Fehler zu beheben.

Standardumfang

In der vorliegenden Dokumentation ist die Funktionalität des Standardumfangs beschrieben. Dieser kann vom Umfang der Funktionalitäten des gelieferten Systems abweichen. Die Funktionalitäten des gelieferten Systems entnehmen Sie ausschließlich den Bestellunterlagen.

Im System können weitere, in dieser Dokumentation nicht erläuterte Funktionen ablauffähig sein. Es besteht jedoch kein Anspruch auf diese Funktionen bei der Neulieferung bzw. im Servicefall.

Diese Dokumentation kann aus Gründen der Übersichtlichkeit nicht sämtliche Detailinformationen zu allen Typen des Produkts enthalten. Ferner kann diese Dokumentation nicht jeden möglichen Fall der Aufstellung, des Betriebs und der Instandhaltung berücksichtigen

Durch den Maschinenhersteller vorgenommene Ergänzungen oder Änderungen am Produkt dokumentiert der Maschinenhersteller.

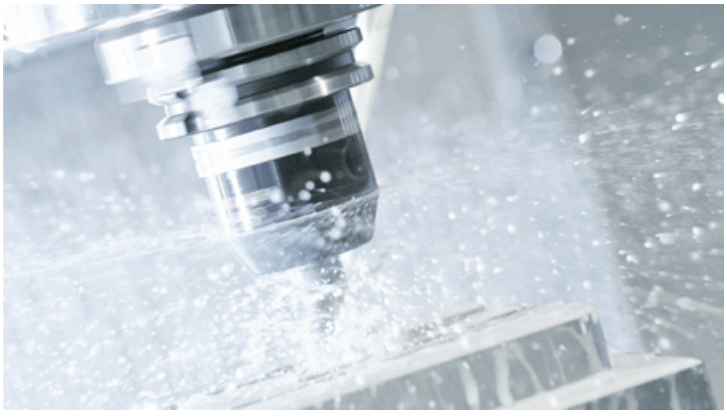
Webseiten Dritter

Dieses Dokument kann Hyperlinks auf Webseiten Dritter enthalten. Siemens übernimmt für die Inhalte dieser Webseiten weder eine Verantwortung noch macht Siemens sich diese Webseiten und ihre Inhalte zu eigen. Siemens kontrolliert nicht die Informationen auf diesen Webseiten und ist auch nicht für die dort bereitgehaltenen Inhalte und Informationen verantwortlich. Das Risiko für deren Nutzung trägt der Nutzer.

1.3 Dokumentation im Internet

1.3.1 Dokumentationsübersicht SINUMERIK 828D

Eine umfangreiche Dokumentation zu den Funktionen von SINUMERIK 828D ab der Version 4.8 SP4 finden Sie unter Dokumentationsübersicht 828D (https://support.industry.siemens.com/cs/ww/de/view/109766724).



Sie haben die Möglichkeit, die Dokumente anzuzeigen oder im PDF- und HTML5-Format herunterzuladen.

Die Dokumentation ist in folgende Kategorien unterteilt:

- Anwender: Bedienen
- Anwender: Programmieren
- Hersteller/Service: Projektieren
- Hersteller/Service: Inbetriebnahme
- Hersteller/Service: Funktionen
- Hersteller/Service: Safety Integrated
- SINUMERIK Integrate/MindApp
- Info & Training

1.3.2 Dokumentationsübersicht SINUMERIK-Bedienkomponenten

Eine umfangreiche Dokumentation zu SINUMERIK-Bedienkomponenten finden Sie unter Dokumentationsübersicht SINUMERIK-Bedienkomponenten (https://support.industry.siemens.com/cs/ww/de/view/109783841).

Sie haben die Möglichkeit, die Dokumente anzuzeigen oder im PDF- und HTML5-Format herunterzuladen.

Die Dokumentation ist in folgende Kategorien unterteilt:

- Bedientafeln
- Maschinensteuertafeln
- Machine Pushbutton Panel
- Handheld Unit/Mini-Handgeräte
- Weitere Bedienkomponenten

Einen Überblick über die wichtigsten Dokumente, Beiträge und Links zum Thema "SINUMERIK" finden Sie unter SINUMERIK Übersicht-Themenseite (https://support.industry.siemens.com/cs/document/109766201/sinumerik-ein-%C3%BCberblick-der-wichtigsten-dokumente-und-links?lc=de-ww).

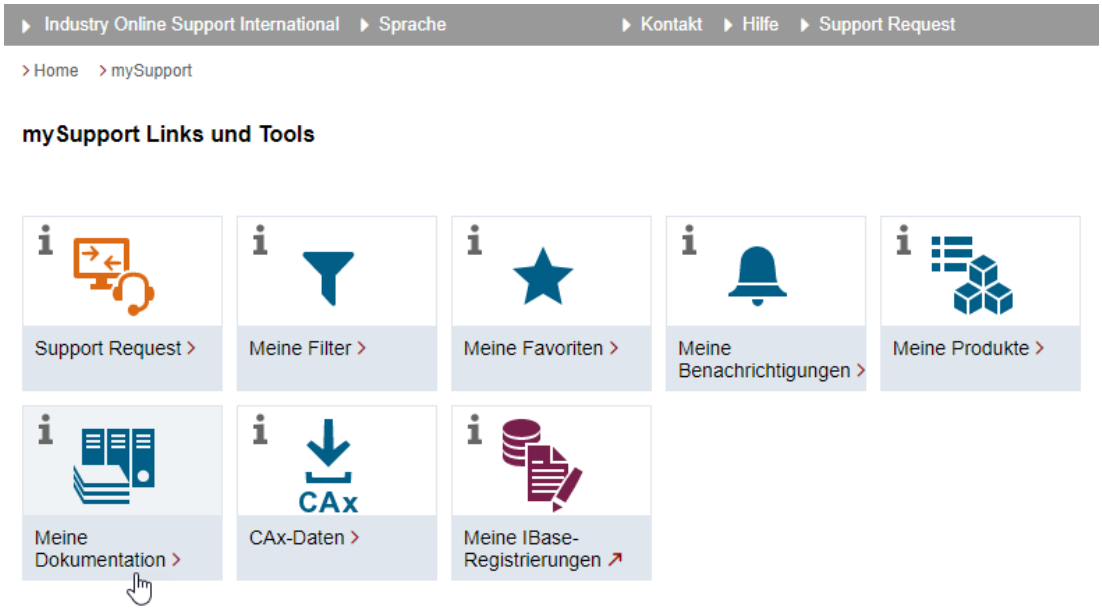
1.4 Feedback zur technischen Dokumentation

Bei Fragen, Anregungen oder Korrekturen zu der im Siemens Industry Online Support veröffentlichten technischen Dokumentation nutzen Sie den Link "Feedback senden" am Ende eines Beitrags.

1.5 mySupport-Dokumentation

Mit dem webbasierten System "mySupport-Dokumentation" können Sie Ihre Dokumentation auf Basis der Siemens-Inhalte individuell zusammenstellen und für die eigene Maschinendokumentation anpassen.

Sie starten die Anwendung über die Kachel "Meine Dokumentation" auf der mySupport-Startseite (https://support.industry.siemens.com/cs/ww/de/my):



Der Export des konfigurierten Handbuchs ist im RTF-, PDF- oder XML-Format möglich.

Hinweis

Siemens-Inhalte, die die Anwendung mySupport-Dokumentation unterstützen, erkennen Sie am Vorhandensein des Links "Konfigurieren".

1.6 Service und Support

Product Support

Weitere Informationen zum Produkt finden Sie im Internet:

Product support (https://support.industry.siemens.com/cs/ww/de/)

Unter dieser Adresse finden Sie Folgendes:

- Aktuelle Produkt-Informationen (Produktmitteilungen)
- FAQ (häufig gestellte Fragen)
- Handbücher
- Downloads
- Newsletter mit den neuesten Informationen zu Ihren Produkten
- Forum zum weltweiten Informations- und Erfahrungsaustausch für Anwender und Spezialisten
- Ansprechpartner vor Ort über unsere Ansprechpartner-Datenbank (→ "Kontakt")
- Informationen über Vor-Ort Service, Reparaturen, Ersatzteile und vieles mehr (→ "Services")

Technical Support

Landesspezifische Telefonnummern für technische Beratung finden Sie im Internet unter der Adresse (https://support.industry.siemens.com/cs/ww/de/sc/4868) im Bereich "Kontakt".

Um eine technische Frage zu stellen, nutzen Sie das Online-Formular im Bereich "Support Request".

Training

Unter folgender Adresse (https://www.siemens.de/sitrain) finden Sie Informationen zu SITRAIN. SITRAIN bietet Trainingsangebote für Siemens-Produkte, Systeme und Lösungen der Antriebs- und Automatisierungstechnik.

Siemens-Support für unterwegs

Mit der preisgekrönten App "Siemens Industry Online Support" haben Sie jederzeit und überall Zugang zu über 300.000 Dokumenten der Siemens Industry-Produkte. Die App unterstützt Sie unter anderem in folgenden Einsatzfeldern:

- Lösen von Problemen bei einer Projektumsetzung
- Fehlerbehebung bei Störungen
- Erweiterung oder Neuplanung einer Anlage

Außerdem haben Sie Zugang zum Technical Forum und weiteren Beiträgen, die von unseren Experten für Sie erstellt werden:

- FAQs
- Anwendungsbeispiele
- Handbücher
- Zertifikate
- Produktmitteilungen und viele andere

Die App "Siemens Industry Online Support" ist für Apple iOS und Android verfügbar.

Data-Matrix-Code auf dem Typenschild

Der Data-Matrix-Code auf dem Typenschild beinhaltet die spezifischen Daten des Geräts. Dieser Code kann mit jedem Smartphone eingelesen werden, über die Mobile App "Industry Online Support" können damit technische Informationen zum entsprechenden Gerät angezeigt werden.

1.7 Wichtige Produktinformationen

Verwendung von OpenSSL

Dieses Produkt kann folgende Software enthalten:

- Software, die durch das OpenSSL-Projekt für die Nutzung innerhalb des OpenSSL-Toolkits entwickelt wurde
- Von Eric Young erstellte kryptografische Software
- Von Eric Young entwickelte Software

Weitere Informationen finden Sie im Internet:

- OpenSSL (https://www.openssl.org)
- Cryptsoft (https://www.cryptsoft.com)

Einhaltung der Datenschutz-Grundverordnung

Siemens beachtet die Grundsätze des Datenschutzes, insbesondere die Gebote der Datenminimierung (privacy by design).

Für dieses Produkt bedeutet das:

Das Produkt verarbeitet oder speichert keine personenbezogenen Daten, lediglich technische Funktionsdaten (z. B. Zeitstempel). Verknüpft der Anwender diese Daten mit anderen Daten (z. B. Schichtplänen) oder speichert er personenbezogene Daten auf dem gleichen Medium (z. B. Festplatte) und stellt so einen Personenbezug her, hat er die Einhaltung der datenschutzrechtlichen Vorgaben selbst sicherzustellen.

Grundlegende Sicherheitshinweise 2

2.1 Allgemeine Sicherheitshinweise

⚠ WARNUNG

Lebensgefahr bei Nichtbeachtung von Sicherheitshinweisen und Restrisiken

Bei Nichtbeachtung der Sicherheitshinweise und Restrisiken in der zugehörigen Hardware-Dokumentation können Unfälle mit schweren Verletzungen oder Tod auftreten.

- Halten Sie die Sicherheitshinweise der Hardware-Dokumentation ein.
- Berücksichtigen Sie bei der Risikobeurteilung die Restrisiken.

⚠ WARNUNG

Fehlfunktionen der Maschine infolge fehlerhafter oder veränderter Parametrierung

Durch fehlerhafte oder veränderte Parametrierung können Fehlfunktionen an Maschinen auftreten, die zu Körperverletzungen oder Tod führen können.

- Schützen Sie die Parametrierung vor unbefugtem Zugriff.
- Beherrschen Sie mögliche Fehlfunktionen durch geeignete Maßnahmen, z. B. NOT-HALT oder NOT-AUS.

2.2 Gewährleistung und Haftung für Applikationsbeispiele

Applikationsbeispiele sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Applikationsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern sollen lediglich Hilfestellung bieten bei typischen Aufgabenstellungen.

Als Anwender sind Sie für den sachgemäßen Betrieb der beschriebenen Produkte selbst verantwortlich. Applikationsbeispiele entheben Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung.

2.3 Security-Hinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.

Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen einen Bestandteil eines solchen Konzepts.

Die Kunden sind dafür verantwortlich, unbefugten Zugriff auf ihre Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Diese Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und nur wenn entsprechende Schutzmaßnahmen (z.B. Firewalls und/oder Netzwerksegmentierung) ergriffen wurden.

Weiterführende Informationen zu möglichen Schutzmaßnahmen im Bereich Industrial Security finden Sie unter:
https://www.siemens.com/industrialsecurity (https://www.siemens.com/industrialsecurity)

Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Produkt-Updates anzuwenden, sobald sie zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter:
https://www.siemens.com/industrialsecurity (https://new.siemens.com/global/en/products/services/cert.html#Subscriptions)

Weitere Informationen finden Sie im Internet:

Projektierungshandbuch Industrial Security (https://support.industry.siemens.com/cs/ww/de/view/108862708)

⚠ WARNUNG

Unsichere Betriebszustände durch Manipulation der Software

Manipulationen der Software, z. B. Viren, Trojaner oder Würmer, können unsichere Betriebszustände in Ihrer Anlage verursachen, die zu Tod, schwerer Körperverletzung und zu Sachschäden führen können.

- Halten Sie die Software aktuell.
- Integrieren Sie die Automatisierungs- und Antriebskomponenten in ein ganzheitliches Industrial Security-Konzept der Anlage oder Maschine nach dem aktuellen Stand der Technik.
- Berücksichtigen Sie bei Ihrem ganzheitlichen Industrial Security-Konzept alle eingesetzten Produkte.
- Schützen Sie die Dateien in Wechselspeichermedien vor Schadsoftware durch entsprechende Schutzmaßnahmen, z. B. Virenscanner.
- Prüfen Sie beim Abschluss der Inbetriebnahme alle security-relevanten Einstellungen.

Funktionsumfang

3

3.1 Funktionsumfang von "Run MyScreens"

Übersicht

Mit "Run MyScreens" ist es möglich, Bedienoberflächen zu gestalten, die maschinenhersteller- oder anwenderspezifische Funktionserweiterungen darstellen oder ein anwenderspezifisches Layout realisieren.

Mit selbst erstellten Bedienoberflächen können unter anderem Zyklenaufrufe generiert werden. Die Gestaltung von Dialogen kann direkt an der Steuerung erfolgen.

"Run MyScreens" ist realisiert durch einen Interpreter und Projektierungsdateien, die die Beschreibung der Bedienoberflächen enthalten.

"Run MyScreens" wird durch ASCII-Dateien konfiguriert: Diese Projektierungsdateien enthalten die Beschreibung der Bedienoberfläche. Die für die Erstellung der Dateien notwendige Syntax wird in den folgenden Kapiteln beschrieben.

Grundumfang

Damit der Maschinenhersteller eigene Dialoge projektieren kann, steht die Funktion "Run MyScreens" zur Verfügung. Schon im Grundumfang können 5 Dialoge im Menübaum der Bedienung oder für kundenspezifische Zyklen-Dialoge projektiert werden.

Software-Option

Um die Anzahl der Dialoge zu erweitern benötigen Sie eine der folgenden Software-Optionen:

- SINUMERIK 828D/840D sl, SINUMERIK Integrate Run MyScreens (6FC5800-0AP64-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI / 3GL (6FC5800-0AP60-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI / WinCC (6FC5800-0AP61-0YB0)

Randbedingungen

Es sind folgende Bedingungen zu beachten:

- Dialogwechsel sind nur innerhalb eines Bedienbereichs möglich.
- Anwendervariablen dürfen nicht den gleichen Namen wie System- oder PLC-Variablen haben (siehe auch Listenhandbuch Systemvariablen /PGAsl/)
- Die von der PLC aktivierten Dialoge bilden einen eigenen Bedienbereich (ähnlich Messzyklen-Bilder).

Tools

- UTF8-fähiger Editor (z. B. der integrierte Editor der Bedienoberfläche oder Notepad)
- Zur Erstellung von Grafiken/Bildern ist ein Grafikprogramm erforderlich.

Dateinamen und Kodierung

Hinweis

Beachten Sie bei der Verwendung von HMI Operate in der NCU, dass auf der CF-Card alle Dateinamen in Kleinbuchstaben geschrieben werden (com, png, txt). Dies ist wegen Linux erforderlich.

Auf der PCU können Sie Dateinamen beliebig in Groß- und Kleinbuchstaben schreiben. Es wird jedoch empfohlen, z. B. für eine eventuelle spätere Übertragung auf Linux, auch hier nur Kleinbuchstaben zu verwenden.

Hinweis

Achten Sie beim Speichern der Projektierungs- und Sprachdateien darauf, dass in dem von Ihnen verwendeten Editor die Kodierung auf UTF-8 eingestellt ist.

Ablagepfade

Beachten Sie bei der Ablage der Projektierungsdateien, Hilfedateien usw. folgende Konvention:

[System siemens-Verzeichnis]		
	Linux:	/card/siemens/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\siemens
[System oem-Verzeichnis]		
	Linux:	/card/oem/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\oem
[System user-Verzeichnis]		
	Linux:	/card/user/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\user
[System addon-Verzeichnis]		
	Linux:	/card/addon/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\addon
Ablageorte		
	"Run MyScreens"-Projektierung:	[System oem-Verzeichnis]/proj
	Konfigurationsdatei:	[System oem-Verzeichnis]/cfg
	Sprachdateien:	[System oem-Verzeichnis]/lng

	Bilddateien:	[System oem-Verzeichnis]/Ico/ico[Auflösung] Beispiel: [System oem-Verzeichnis]/Ico/ico640
	Online-Hilfe:	[System oem-Verzeichnis]/hlp/[Sprache] Beispiel: [System oem-Verzeichnis]/hlp/eng

Verwendung

Folgende Funktionen können realisiert werden:

- Dialoge einblenden und bereitstellen von:
  - Softkeys
  - Variablen
  - Text und Hilfetext
  - Grafiken und Hilfebildern
- Dialoge aufrufen durch:
  - Betätigen der (Einstiegs-) Softkeys
  - Anwahl von PLC/NC
- Dialoge dynamisch umstrukturieren:
  - Softkeys ändern, löschen
  - Variablenfelder definieren und gestalten
  - Anzeigetexte (sprachabhängig oder -unabhängig) einblenden, austauschen, löschen
  - Grafiken einblenden, austauschen, löschen
- Aktionen in Gang setzen bei:
  - Dialoge einblenden
  - Werte (Variablen) eingeben
  - Softkeys betätigen
  - Dialoge verlassen
  - Wertänderung einer System- oder Anwendervariable
- Datenaustausch zwischen Dialogen
- Variablen:
  - Lesen (NC-, PLC-, Anwendervariablen)
  - Schreiben (NC-, PLC-, Anwendervariablen)
  - Verknüpfen mit mathematischen, vergleichenden oder logischen Operatoren

- Funktionen ausführen:
  - Unterprogramme
  - Datei-Funktionen
  - PI-Dienste
- Berücksichtigung von Schutzstufen nach Benutzergruppen

Erste Schritte

4

4.1 Einleitung

Anhand von folgendem Beispiel lernen Sie die notwendigen Arbeitsschritte kennen, um mit Run MySreens eigene Dialoge in die SINUMERIK Operate Bedienoberfläche einzufügen. Sie lernen unter anderem wie Sie eigene Dialoge erstellen, kontextsensitive Hilfebilder und Hilfeaufrufe einfügen, Softkeys definieren und wie Sie zwischen den Dialogen navigieren können.

4.2 Beispielprojekt

4.2.1 Aufgabenbeschreibung

Im Beispielprojekt erstellen Sie die im Folgenden beschriebenen Dialoge inklusive einer kontextsensitiven Hilfe.

Dialog 1

Im ersten Dialog werden beschreibbare R-Parameter (0 und 1) und die Geometrieachsnamen angezeigt (Eingabefelder). Für die beiden R-Parameter werden entsprechende Hilfebilder eingebunden. Für die Geometrieachsen wird eine kontextsensitive Hilfe eingebunden. Darüber hinaus enthält der Dialog Beispiele für Trennlinien (horizontal und vertikal), Toggle-Felder, Ein-/Ausgabefelder mit integrierter Einheiten-Selektion und für Fortschrittsbalken (mit und ohne Farbumschlag).

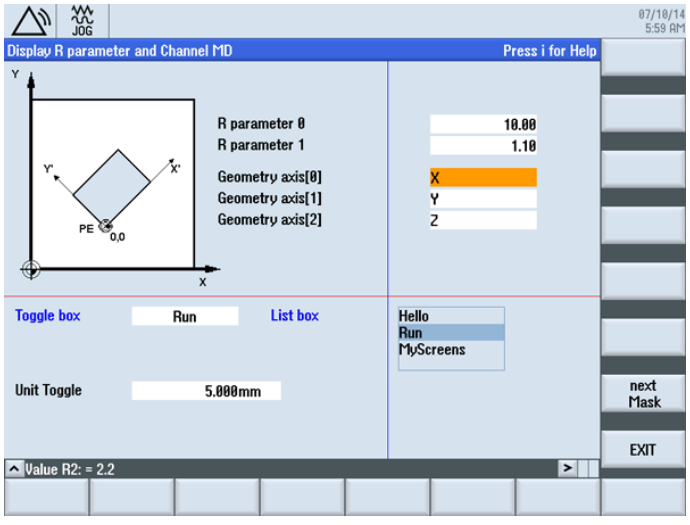


Bild 4-1 R-Parameter mit Hilfebildern

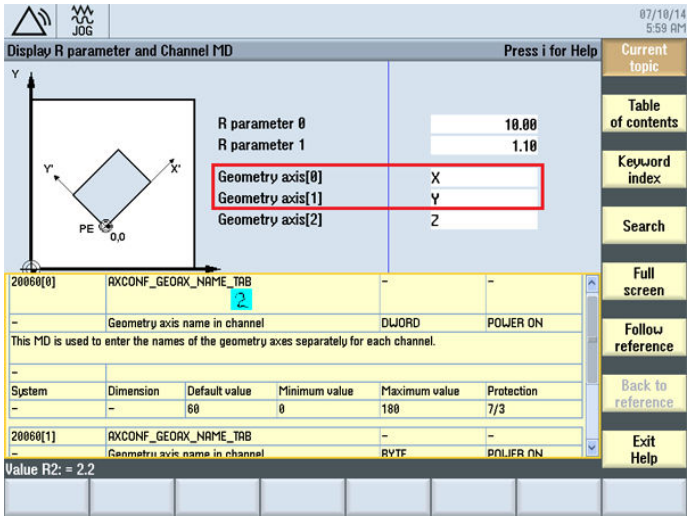


Bild 4-2 Geometrieachsen mit kontextsensitiver Hilfe

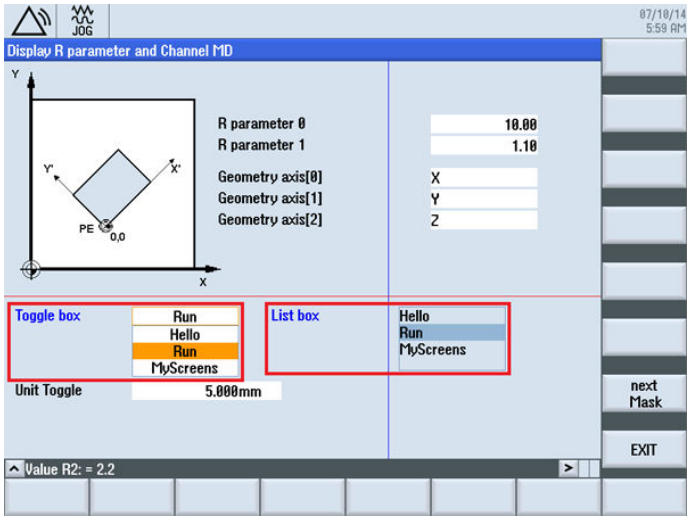


Bild 4-3 Toggle-Felder: Liste und Box

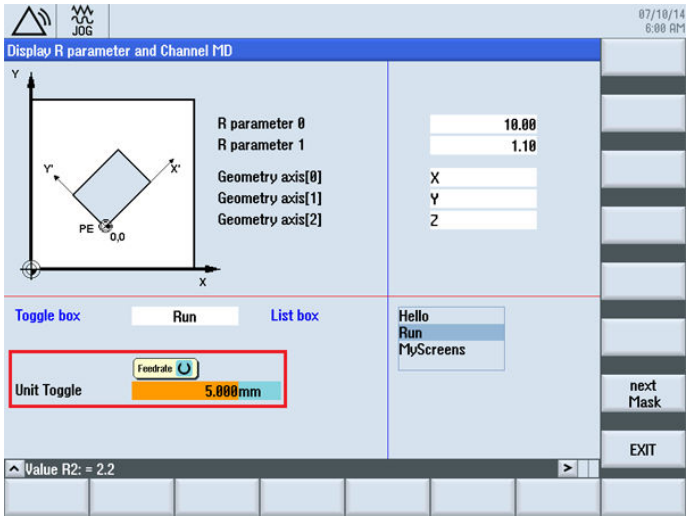


Bild 4-4 Ein-/Ausgabefeld mit integrierter Einheiten-Selektion

Dialog 2

Im zweiten Dialog werden MKS- und WKS-Werte angezeigt. Darüber hinaus enthält der Dialog Beispiele für eine Fortschrittsanzeige mit und ohne Farbumschlag.

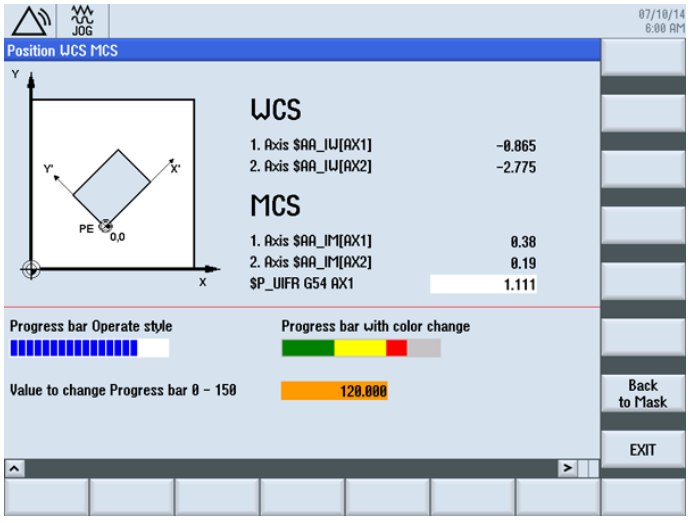


Bild 4-5 Fortschrittsbalken mit (rechts) und ohne (links) Farbumschlag

Navigation

Der Aufruf des ersten Dialogs erfolgt über den Einstiegssoftkey „START“ im Bedienbereich Diagnose. Sie verwenden den horizontalen Softkey SK7.

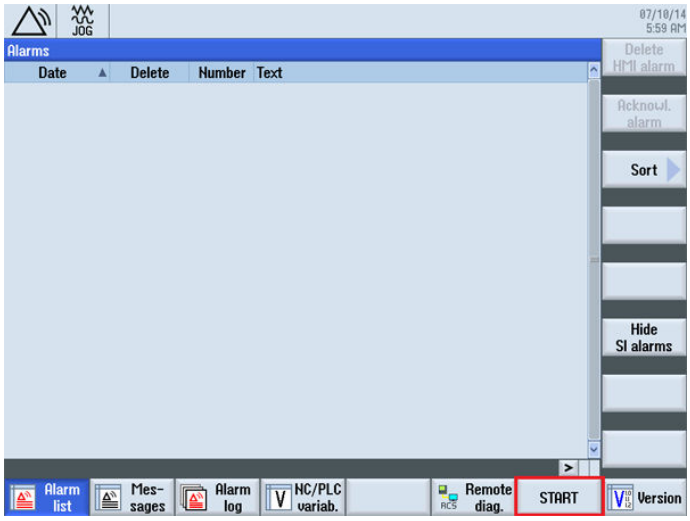


Bild 4-6 Einstiegssoftkey „START“ im Bedienbereich Maschine, Betriebsart AUTO

Aus dem ersten Dialog heraus können Sie mit dem Softkey „next Mask“ den zweiten Dialog aufrufen. Mit dem Softkey „EXIT“ kehren Sie in das Grundbild des Bedienbereichs zurück (siehe Bild oben).

Vom zweiten Dialog aus können Sie ebenfalls über den Softkey „EXIT“ zum Grundbild des Bedienbereichs zurückkehren (siehe Bild oben). Über den Softkey "Back to Mask" können Sie zum ersten Dialog zurückkehren.

Vorgehensweise

In den folgenden Kapiteln werden die notwendigen Arbeitsschritte erläutert:

1. Erstellen der Projektierungsdatei (COM-Datei) (Seite 25)
2. Ablegen der Projektierungsdatei im OEM Verzeichnis (Seite 28)
3. Erstellen der Online-Hilfe (Seite 29)
4. Einbinden der Online-Hilfe und Ablegen der Dateien im OEM Verzeichnis (Seite 32)
5. Kopieren der "easyscreen.ini" in das OEM Verzeichnis (Seite 33)
6. Bekanntmachen der COM-Datei in der "easyscreen.ini" (Seite 33)
7. Testen des Projektes (Seite 33)

4.2.2 Erstellen der Projektierungsdatei (Beispiel)

Inhalt der Projektierungsdatei

Erstellen Sie in einem UTF8-fähigen Editor die Projektierungsdatei "diag.com" für die beiden Dialoge.

```
; Anfangskennung Einstiegssoftkey
//S(START)
   ; Einstiegssoftkey nur Text
   HS7=("START")
   ; Einstiegssoftkey mit sprachabhängigem Text und png
   ;HS7=([$80792,"\\sk_ok.png"])

   ; Press-Methode
   PRESS(HS7)
      ; Funktion LM oder LS
      LM("MASK1")
      ; LM mit Angabe einer com-Datei
      LM("MASK1","TEST.COM")
   END_PRESS
; Endekennung Einstiegssoftkey
//END

; Definition Dialog 1 mit Überschrift und Bild
//M(MASK1/"Display R parameter and Channel MD"/"mz961_01.png")
   ; Definition der Variablen
   DEF VAR1 = (R2///,"R parameter 0"///"$R[0]"/200,50,150/400,50,100,)
   ; mit Hilfebild
   DEF VAR2 = (R2///,"R parameter 1"//"mz961_02.png"/"$R[1]"/200,70,150/400,70,100)
   ; mit Online-Hilfe
   DEF ACHS_NAM1 = (S///"Press i for Help","Geometry axis[0]"////200,100,150/400,100,100//"sinume-
rik_md_1.html","20060[0]")
   ; mit Online-Hilfe
   DEF ACHS_NAM2 = (S///"Press i for Help","Geometry axis[1]"////200,120,150/400,120,100//"sinume-
rik_md_1.html","20060[1]")
   DEF ACHS_NAM3 = (S///,"Geometry axis[2]"////200,140,150/400,140,100)
   ; Definition Toggle-Felder und Einheiten Toggler
   DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run"/,"Toggle box"////10,230,100/120,230,100/,,6)

   DEF VAR_TGB = (S/* "Hello", "Run", "MyScreens"/"Run"/,"List box"/
dt4///250,230,100/370,230,100,60/,,"#0602ee")
   ; BC, FC, BC_ST, FC_ST, BC_GT, FC_GT, BC_UT, FC_UT, SC1, SC2
   DEF VarEdit = (R///,"Unit Toggle",,,"Feedrate"///"$R[11]"/5,300,100/120,300,100///"VarTgl"),
VarTgl = (S/*0="mm",1="inch"/0//WR2////220,300,40)
```

```
; Definition Softkeys im Dialog
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("next Mask")
VS8=("EXIT")

; Definition LOAD-Block
LOAD
   ; Lesen Wert mit RNP
   ACHS_NAM1 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]")
   ACHS_NAM2 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[1]")
   ACHS_NAM3 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[2]")
   ; Ausgabe einer Dialogzeile
   DLGL("Value R2: = " << RNP("$R[2]"))
   ; Schreiben eines Wertes mit WNP
   WNP("$R[3]",VAR0)

  ; vertikale Trennlinie
  V_SEPARATOR(360,1,6,1)
  ; horizontale Trennlinie
  H_SEPARATOR(220,1,7,1)
END_LOAD

; Press-Methode
PRESS(VS7)
   ; Laden eines weiteren Dialogs
   LM("MASK2")
; Endekennung Press-Methode
END_PRESS

; Press-Methode
PRESS(VS8)
```

```
      ; Verlassen des Dialogs
      EXIT
   ; Endekennung Press-Methode
   END_PRESS
; Endekennung Dialog 1
//END

; Definition Dialog 2 mit Überschrift und Bild
//M(MASK2/"Position WCS MCS"/"mz961_01.png")
   ; Definition der Variablen
   DEF TEXT1 = (I///,"WCS"/WR0,fs2///230,30,120/,,1)
   DEF VAR1 = (R3///,"1. Axis $AA_IW[AX1]"/WR1//"$AA_IW[AX1]"/230,70,150/400,70,100)
   DEF VAR2 = (R3///,"2. Axis $AA_IW[AX2]"/WR1//"$AA_IW[AX2]"/230,90,150/400,90,100)

   DEF TEXT2 = (I///,"MCS"/WR0,fs2///230,120,120/,,1)
   DEF VAR3 = (R2///,"1. Axis $AA_IM[AX1]"/WR1//"$AA_IM[AX1]"/230,160,150/400,160,100)
   DEF VAR4 = (R2///,"2. Axis $AA_IM[AX2]"/WR1//"$AA_IM[AX2]"/230,180,150/400,180,100)
   DEF VAR5 = (R3///,"$P_UIFR G54 AX1"///"$P_UIFR[1,AX1,TR]"/230,200,150/400,200,100)

   ; Definition der Fortschrittsbalken
   DEF PROGGY0 = (R/0,150//,"Progress bar Operate style"/DT2,DO0//"$R[10]"/5,240,190/5,260,150/6,10)
   DEF PROGGY4 = (R/0,150,50,100/120/,"Progress bar with color change"/DT1,DO0//"$R[10]"/
260,240,190/260,260,150/3,4,,,9,7)

   ; Definition der Variablen
   DEF PROGVAL = (R/0,150//,"Value to change Progress bar 0 - 150"///"$R[10]"/5,300,230/260,300,100)

  ; Definition Softkeys im Dialog
  HS1=("")
  HS2=("")
  HS3=("")
  HS4=("")
  HS5=("")
  HS6=("")
  HS7=("")
  HS8=("")
  VS1=("")
  VS2=("")
  VS3=("")
  VS4=("")
  VS5=("")
  VS6=("")
  VS7=("Back to Mask")
  VS8=("EXIT")
```

```
   ; Load-Methode
   LOAD
      H_SEPARATOR(230,1,7,1)
   END_LOAD

   ; Change-Methode
   CHANGE(VAR5)
      ; Funktion PI_START
      PI_START("/NC,201,_N_SETUFR")
   ; Endekennung Change-Methode
   END_CHANGE

   ; Press-Methode
   PRESS(RECALL)
      ; zurück in aufrufende Maske
      LM("MASK1")
   ; Endekennung Press-Methode
   END_PRESS

   ; Press-Methode
   PRESS(VS7)
      ; zurück in aufrufende Maske
      LM("MASK1")
   ; Endekennung Press-Methode
   END_PRESS

   ; Press-Methode
   PRESS(VS8)
      ; Verlassen des Dialog zur Standardapplikation
      EXIT
   ; Endekennung Press-Methode
   END_PRESS
; Endekennung Dialog
//END
```

4.2.3 Ablegen der Projektierungsdatei im OEM Verzeichnis (Beispiel)

Ablagepfad

Legen Sie die Projektierungsdatei "diag.com" unter folgendem Pfad ab:

[System oem-Verzeichnis]/proj

4.2.4 Erstellen der Online-Hilfe (Beispiel)

Erstellen Sie die HTML-Datei "sinumerik_md_1.html". Das Beispiel unter "Inhalt der Online-Hilfe" zeigt den kontextsensitiven Hilfeaufruf über **name="20060[0]"** und **name="20060[1]"**.

Hinweise

Hinweise zur Erstellung von HTML-Dateien finden Sie im Kapitel HTML-Dateien erzeugen (Seite 73).

Hinweise zur Einbindung der Online-Hilfe finden Sie im Kapitel Einbinden der Online-Hilfe und Ablegen der Dateien im OEM Verzeichnis (Beispiel) (Seite 32).

Inhalt der Online-Hilfe

Das Beispiel ist nicht kommentiert.

```
<html><head><meta http-equiv="Content-Type" content="text/html; charset="UTF-8"/><title></
title></head>
   <body>
      <table border="1" cellspacing="0" cellpadding="2" width="100%">
         <tr>
            <td><a name="20060[0]"><b>20060[0]</b></a></td>
            <td colspan="3">AXCONF_GEOAX_NAME_TAB</td>
            <center>
               <img src="../picture/bitmap3.png">
            </center>
            <td>-</td>
            <td>-</td>
         </tr>
         <tr>
            <td>-</td>
            <td colspan="3">Geometry axis name in channel</td>
            <td>DWORD</td>
            <td>POWER ON</td>
         </tr>
         <tr>
            <td colspan="6">This MD is used to enter the names of the geometry axes
separately for each channel.<br />
         <tr>
            <td>-</td>
            <td colspan="5"> </td>
         </tr>
         <tr>
          <td width="16*">System</td><td width="16*">Dimension</td><td width="16*">Default
value</td>
            <td width="16*">Minimum value</td><td width="16*">Maximum value</td>
            <td width="16*">Protection</td>
         </tr>
         <tr>
            <td>-</td>
            <td>-</td>
            <td>60</td>
            <td>0</td>
            <td>180</td>
            <td>7/3</td>
         </tr>
      </table>
      <p></p>
      <table border="1" cellspacing="0" cellpadding="2" width="100%">
         <tr>
            <td><a name="20060[1]"><b>20060[1]</b></a></td>
            <td colspan="3">AXCONF_GEOAX_NAME_TAB</td>
            <td>-</td>
            <td>-</td>
         </tr>
         <tr>
            <td>-</td>
            <td colspan="3">Geometry axis name in channel</td><td>BYTE</td>
            <td>POWER ON</td>
         </tr>
         <tr>
```

```
            <td colspan="6">This MD is used to enter the names of the geometry axes
separately for each channel.<br />
         <tr>
            <td>-</td>
            <td colspan="5"> </td>
         </tr>
         <tr>
            <td width="16*">System</td>
            <td width="16*">Dimension</td>
            <td width="16*">Default value</td>
            <td width="16*">Minimum value</td>
            <td width="16*">Maximum value</td>
            <td width="16*">Protection</td>
         </tr>
         <tr>
            <td>-</td>
            <td>-</td>
            <td>0</td>
            <td>0</td>
            <td>2</td>
            <td>7/3</td>
         </tr>
      </table>
      <p></p>
   </body>
</html>
```

4.2.5 Einbinden der Online-Hilfe und Ablegen der Dateien im OEM Verzeichnis (Beispiel)

Einbinden der Online-Hilfe

Die Syntax zur Einbindung der Online-Hilfe ist analog zu SINUMERIK Operate:

DEF RFP=(R//1/,"RFP","RFP"///////"sinumerik_md_1.html","9006")

Hinweis

Der Name der HTML-Datei muss bedingt durch LINUX in Kleinbuchstaben geschrieben werden!

Ablagepfad

Legen Sie die HTML-Datei "sinumerik_md_1.html" für die deutschsprachige Hilfe unter folgendem Pfad ab:

[System oem-Verzeichnis]/hlp/deu

Für weitere Sprachen müssen Sie die Ordner bei Bedarf anlegen (z. B: chs, eng, esp, fra, ita ...).

Eine Liste der Sprachkennzeichen finden Sie im Anhang "Referenz-Listen".

Weitere Informationen

Detaillierte Informationen zur Erstellung von OEM-spezifischen Online-Hilfen finden Sie im Kapitel Online-Hilfe projektieren (Seite 72).

4.2.6 Kopieren der "easyscreen.ini" in das OEM Verzeichnis (Beispiel)

Ablagepfad

Kopieren Sie die Datei "easyscreen.ini" aus dem Verzeichnis

[System siemens-Verzeichnis]/cfg

in das Verzeichnis

[System oem-Verzeichnis]/cfg.

4.2.7 Bekanntmachen der COM-Datei in der "easyscreen.ini" (Beispiel)

Anpassung in der "easyscreen.ini"

Nehmen Sie folgende Änderung in der "easyscreen.ini" im OEM-Verzeichnis vor. Damit ist die Projektierungsdatei "diag.com" bekannt gemacht.

```
;#########################
;# AREA Diagnosis       #
;#########################
;<=============================================================>
;< OEM Softkey on first horizontal Main Menu                    >
;< SOFTKEY position="7"                                         >
;<=============================================================>
StartFile28 = area := AreaDiagnosis, dialog:= SlDgDialog, menu := DgGlobalHu, startfile := diag.com
```

4.2.8 Testen des Projektes (Beispiel)

Dialogaufruf testen

Wechseln Sie in den Bedienbereich Diagnose. Klicken Sie auf den horizontalen Softkey "START".

Wenn "Run MyScreens" beim Interpretieren der Projektierungsdateien Fehler erkennt, so werden diese in der Textdatei "easyscreen_log.txt" abgelegt. Weitere Informationen finden Sie im Kapitel Fehlerbehandlung (Logbuch) (Seite 41).

Kontextsensitive Hilfebilder und Online-Hilfe testen

Testen Sie die kontextsensitiven Hilfebilder und die Online-Hilfe. Falls Fehler auftreten, prüfen Sie die Ablagepfade.

Grundlagen

5

5.1 Struktur der Projektierungsdatei

Einleitung

Die Beschreibung neuer Bedienoberflächen wird in Projektierungsdateien gespeichert. Diese Dateien werden automatisch interpretiert und das Ergebnis am Bildschirm dargestellt. Die Projektierungsdateien sind bei Lieferung nicht vorhanden und müssen erst angelegt werden.

Zur Erstellung der Projektierungs- und Sprachdateien wird ein UTF-8-fähiger Editor (z. B. der integrierte Editor der Bedienoberfläche oder Notepad) verwendet. Die Beschreibung kann zusätzlich durch Kommentare erläutert werden. Als Kommentarzeichen wird vor jede Erläuterung ein ";" eingefügt.

Hinweis

Achten Sie beim Speichern der Projektierungs- und Sprachdateien darauf, dass in dem von Ihnen verwendeten Editor die Kodierung auf UTF-8 eingestellt ist.

Die Schlüsselwörter dürfen jedoch - auch in einer UTF-8-kodierten COM-Datei - nur aus Zeichen bestehen, die im ASCII-Zeichensatz enthalten sind. Sonst kann die Interpretation und damit die Darstellung der Masken/Dialoge nicht gewährleistet werden.

Projektierung

Jeder HMI-Bedienbereich verfügt über feste Einstiegssoftkeys, über die in die neu erstellten Dialoge verzweigt werden kann.

Bei einem Aufruf "Lade Maske" (LM) oder "Lade Softkey-Leiste" (LS) in einer Projektierungsdatei kann ein neuer Dateiname angegeben werden, in dem das aufgerufene Objekt steht. Auf diese Weise kann die Projektierung gegliedert werden, z. B. alle Funktionen einer Bedienebene in einer eigenen Projektierungsdatei.

Aufbau der Projektierungsdatei

Eine Projektierungsdatei besteht aus folgenden Elementen:

1. Beschreibung der Einstiegssoftkeys
2. Definition von Dialogen
3. Definition der Variablen
4. Beschreibung der Methoden
5. Definition von Softkey-Leisten

Hinweis

Reihenfolge

Die angegebene Reihenfolge in der Projektierungsdatei ist zwingend einzuhalten.

Beispiel:

```
//S (START)                  ; Definition der Einstiegssoftkeys (optional)
....
//END
//M (.....)                  ; Definition des Dialogs
DEF .....                    ; Definition der Variablen
LOAD                         ; Beschreibung der Blöcke
...
END_LOAD
UNLOAD
...
END_UNLOAD
...
//END
//S (...)                    ; Definition einer Softkey-Leiste
//END
```

Ablage der Projektierungsdateien

Die Projektierungsdateien werden im Verzeichnis [System user-Verzeichnis]/proj und entsprechend auch in den Verzeichnissen [System addon-Verzeichnis] und [System oem-Verzeichnis] abgelegt.

Konvertierung von Texten aus anderen HMI-Applikationen

Vorgehensweise, um eine Textdatei mit Codepage-Kodierung nach Text-Kodierung UTF-8 zu konvertieren:

1. Öffnen Sie die Textdatei auf einem PG/PC in einem Text-Editor.
2. Stellen Sie beim Speichern die UTF-8 Kodierung ein

Der Einlesemechanismus über Codepage-Kodierung wird weiterhin unterstützt.

5.2 Aufbau des Bedienbaums

Prinzip des Bedienbaums

Mehrere miteinander verknüpfte Dialoge bilden einen Bedienbaum. Eine Verknüpfung besteht, wenn Sie von einem Dialog in einen anderen wechseln können. Über neu definierte horizontale oder vertikale Softkeys in diesem Dialog können Sie zum Vorgängerdialog oder in einen beliebigen anderen Dialog wechseln.

Hinter jedem Einstiegssoftkey kann ein Bedienbaum erzeugt werden:

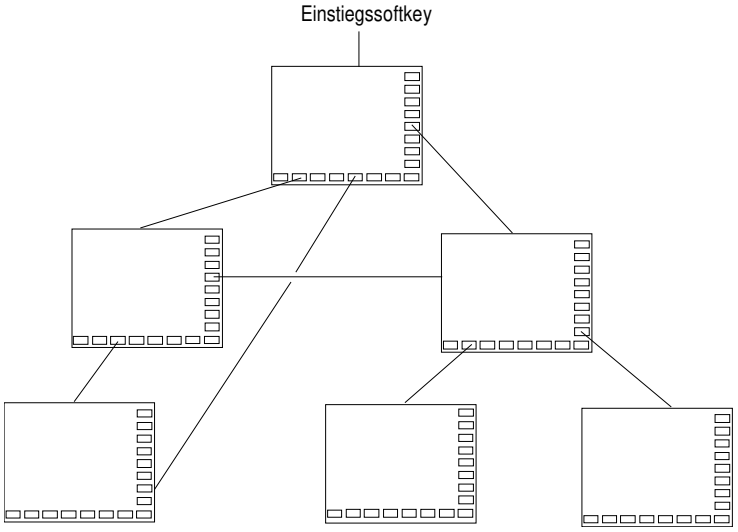


Bild 5-1 Bedienbaum

Einstiegssoftkeys

In einer in der "easyscreen.ini" angegebenen Projektierungsdatei werden ein oder mehrere Softkeys (Einstiegssoftkeys) definiert, die Ausgangspunkt für die eigenen Bedienabläufe sind.

Mit einer Softkey-Festlegung ist das Laden eines eigenen Dialogs oder einer weiteren Softkey-Leiste verbunden, mit denen dann die weiteren Aktionen durchgeführt werden.

Bei Betätigung des Einstiegssoftkeys wird der zugeordnete Dialog geladen. Dabei werden auch die zum Dialog gehörigen Softkeys aktiviert. Variablen werden auf den Standardpositionen ausgegeben, wenn keine speziellen Positionen projektiert wurden.

Zur Standardapplikation zurückkehren

Sie können den neu gestalteten Dialog verlassen und zum Standard-Bedienbereich zurückzukehren.

Mit der Taste <RECALL> können Sie die neu gestalteten Bedienoberflächen verlassen, solange nichts anderes für diese Taste projektiert wurde.

Hinweis

Dialoge im PLC-Anwenderprogramm aufrufen

Außer über Softkeys ist die Dialoganwahl auch von PLC aus möglich: Für den Signalaustausch von PLC → HMI existiert ein Nahtstellensignal im DB19.DBB10.

5.3 Definition und Funktionen für Einstiegssoftkeys

5.3.1 Einstiegssoftkey definieren

Dialogunabhängiger Softkey

Einstiegssoftkeys sind dialogunabhängige Softkeys, die nicht von einem Dialog aus aufgerufen werden, sondern die **vor** dem ersten neuen Dialog projektiert wurden. Um zum Einstiegsdialog oder zu einer Einstiegssoftkey-Leiste zu gelangen, muss der Einstiegssoftkey definiert werden.

Programmierung

Der Beschreibungsblock für einen Einstiegssoftkey ist wie folgt aufgebaut:

```
//S(Start)                ; Anfangskennung Einstiegssoftkey
HS1=(...)                 ; Einstiegssoftkey definieren: horizontaler SK 1
PRESS(HS1)                ; Methode
  LM...                   ; Funktion LM oder LS
END_PRESS                 ; Methodenende
//END                     ; Endekennung Einstiegssoftkey
```

Zulässige Positionen für Einstiegssoftkeys

In den Bedienbereichen sind folgende Positionen für Einstiegssoftkeys von "Run MyScreens" zulässig:

Bedienbereich	Position
Maschine	HSK6
Parameter	HSK7
Programm	HSK6 und HSK15 Messzyklen: HSK13 und HSK14
Programm-Manager	HSK2-8 und HSK12-16, falls nicht mit Laufwerken belegt.

Bedienbereich	Position
Diagnose	HSK7
Inbetriebnahme	HSK7

Einstiegssoftkeys werden in speziellen Dateien projektiert. Die Namen dieser Dateien werden in der Datei "easyscreen.ini" deklariert. Typischerweise haben diese einen bedienbereichspezifischen Namen (z. B. "startup.com" für den Bereich Inbetriebnahme). Eine Ausnahme stellt der Bedienbereich Maschine dar. Im Bedienbereich Maschine existieren mehrere betriebsartenspezifische Dateien ("ma_jog.com", "ma_auto.com").

Die Softkey-Leiste mit den Einstiegssoftkeys heißt "Start". Die Benutzung bestehender Einstiegssoftkey-Projektierungen ist weiterhin möglich. Die Funktionalität des Zusammenführens ("Merge") der Einstiegssoftkeys mit den Softkeys der jeweiligen HMI-Applikation (Bedienbereich) im Einstiegssoftkey-Menü wird nicht unterstützt.

Bis zum ersten Dialogaufruf, also dem Zeitpunkt, ab dem die volle Funktionalität (z. B. Ausführung von PRESS-Blöcken) zur Verfügung steht, kann ein Menü oder eine Softkey-Leiste nur komplett durch eine andere ersetzt werden.

Vorlage für Konfiguration der Einstiegssoftkeys

Eine detaillierte Beschreibung aller zulässigen Positionen für Einstiegssoftkeys und deren Projektierung befindet sich in der Datei "easyscreen.ini" in folgendem Verzeichnis:

[System siemens-Verzeichnis]/cfg

Diese Datei dient als Vorlage für die eigene Konfiguration der Einstiegssoftkeys.

Siehe auch

Listen der Einstiegssoftkeys

5.3.2 Funktionen für Einstiegssoftkeys

Funktionen für dialogunabhängige Softkeys

Mit Einstiegssoftkeys können nur bestimmte Funktionen ausgelöst werden.

Folgende Funktionen sind zulässig:

- Mit der **Funktion LM** kann ein anderer Dialog geladen werden: **LM***(*"Bezeichner"[,"Datei"])
- Mit der **Funktion LS** kann eine andere Softkey-Leiste eingeblendet werden: **LS**("Bezeichner"[, "Datei"][, Merge])
- Mit der **Funktion "EXIT"** können Sie die neu gestalteten Bedienoberflächen verlassen und zur Standardapplikation zurückkehren.
- Mit der **Funktion "EXITLS"** können Sie die aktuelle Bedienoberfläche verlassen und eine definierte Softkey-Leiste laden.

Methode PRESS

Innerhalb des Beschreibungsblocks wird der Softkey definiert und in der Methode PRESS die Funktion "LM" oder "LS" zugewiesen.

Wird die Definition des Einstiegssoftkey als Kommentar gekennzeichnet (Semikolon ; am Anfang der Zeile) oder die Projektierungsdatei entfernt, ist der Einstiegssoftkey ohne Funktion.

```
//S(Start)                        ; Anfangskennung
HS6=("1. Maske")                  ; horizontalen SK 6 mit "1. Maske" beschriften
PRESS(HS6)                        ; PRESS-Methode für horizontalen SK 6
  LM("Maske1")                    ; Funktion Maske1 laden, wobei Maske 1 innerhalb
                                  der gleichen Datei definiert sein muss.
END_PRESS                         ; Ende der PRESS-Methode
HS7=("2. Maske")                  ; horizontalen SK 7 mit "2. Maske" beschriften
PRESS(HS7)                        ; PRESS-Methode für horizontalen SK 7
  LM("Maske2")                    ; Funktion Maske2 laden, wobei Maske2 innerhalb
                                  der gleichen Datei definiert sein muss.
END_PRESS                         ; Ende der PRESS-Methode
//END                             ; Ende-Kennung des Einstiegsblocks
```

Beispiel

```
HS1 = ("neue Softkey-Leiste")
HS2 = ("keine Funktion")
PRESS (HS1)
  LS("Leiste1")                     ; neue Softkey-Leiste laden
END_PRESS
PRESS (HS2)                         ; leere PRESS-Methode
END_PRESS
```

Verschiedene Projektierungen der Einstiegssoftkeys

Verschiedene Projektierungen der Einstiegssoftkeys werden zusammengeführt. Dabei wird zunächst aus der "easyscreen.ini" der Name der zu interpretierenden Datei ausgelesen. Es wird in folgenden Verzeichnissen nach *.com Dateien gesucht:

- [System user-Verzeichnis]/proj
- [System oem-Verzeichnis]/proj
- [System addon-Verzeichnis]/proj
- [System siemens-Verzeichnis]/proj

Die enthaltenen Projektierungen für die Einstiegssoftkeys werden nun zu einer Projektierung zusammengeführt, d.h. die einzelnen Softkeys werden verglichen. Gibt es zwei oder mehr Projektierungen für einen Softkey, wird stets der höherwertige in die Merge-Version übernommen.

Eventuell enthaltene Softkey-Leisten oder Dialoge werden ignoriert. Enthält ein Softkey einen Befehl ohne Dateiangabe z. B. LM("test"), da die gewünschte Softkey-Leiste oder Dialog in derselben Datei enthalten ist, wird der entsprechende Dateiname in der internen Merge-Version ergänzt, so dass hierbei keine Anpassungen nötig sind. Die erhaltene Merge-Projektierung kommt im Anschluss zur Anzeige.

5.4 Fehlerbehandlung (Logbuch)

Übersicht

Wenn "Run MyScreens" beim Interpretieren der Projektierungsdateien Fehler erkennt, so werden diese in der Textdatei "easyscreen_log.txt" abgelegt. Es werden nur Fehler des aktuell angewählten Dialogs eingetragen. Fehlereinträge aus zuvor angewählten Dialogen werden gelöscht.

Die Datei enthält folgende Informationen:

- Bei welcher Aktion ein Fehler aufgetreten ist.
- Die Zeilen- und Spaltennummer des ersten fehlerhaften Zeichens.
- Die gesamte fehlerhafte Zeile der Projektierungsdatei.
- Die durch die Funktion DEBUG gemachten Einträge.

Hinweis

Jedem Eintrag wird ein aktueller Zeitstempel in eckigen Klammern vorangestellt. Dies kann z. B. hilfreich sein, um zeitkritische Projektierungen zu analysieren.

Ablage der "easyscreen-log.txt"

Die Datei "easyscreen_log.txt" ist in folgendem Verzeichnis abgelegt:

[System user-Verzeichnis]/log

Syntax

Mit der Interpretation der Syntax wird erst begonnen, wenn der Einstiegssoftkey definiert und ein Dialog mit Anfangs- und Endekennung und einer Definitionszeile projektiert ist.

```
//S(Start)
HS6=("1. Maske")
PRESS(HS6)
  LM("Maske1")
END_PRESS
//END
```

```
//M(Maske1)
  DEF Var1=(R)
  DEF VAR2 = (R)
  LOAD
  VAR1 = VAR2 + 1              ; Fehlermeldung im Logbuch, da VAR2 keinen Wert hat
...
//END

; richtig wäre z.B.:
//M(Maske1)
   DEF Var1=(R)
   DEF VAR2 = (R)

   LOAD
      VAR2 = 7
      VAR1 = VAR2 + 1 ;
...
```

5.5 Hinweise zur "easyscreen.ini"

Ab SINUMERIK Operate V4.7 ist die "easycreen.ini" um die in diesem Kapitel beschriebenen Einträge erweitert. Die "easyscreen.ini" finden Sie im Verzeichnis [System siemens-Verzeichnis]/cfg.

Hilfebild-Startposition

Eintrag in "easyscreen.ini":

```
[GENERAL]
HlpPicFixPos=true
```

Hinweis:

Die Startposition von Hilfebildern wird auflösungsunabhängig an die projektierte Pixelposition positioniert (Standard=true).

Streckungsverhalten, Standard-Zeilenhöhe und -Zeilenabstand

Eintrag in easyscreen.ini:

```
[GENERAL]
SymmetricalAspectRatio=false
```

```
DefaultLineHeight=18
DefaultLineSpacing=3
```

Hinweise:

- Der Eintrag „SymmetricalAspectRatio" bestimmt, ob bei der Anpassung einer Projektierung an eine bestimmte Bildschirmauflösung in X- und in Y-Richtung die gleichen Streckungsfaktoren verwendet werden sollen.
  - „false" (Standard): Felder und Grafiken werden bei Widescreen-Auflösungen in Y-Richtung gestaucht (unsymmetrische Streckungen bezogen auf 640x480). Zum Beispiel wird ein in 640x480 projektiertes Quadrat dadurch bei einem Widescreen-Panel vertikal gestaucht und so zu einem Rechteck.
  - „true": In X- und Y-Richtung wird mit dem gleichen Streckungsfaktor gestreckt, wodurch Felder und Grafiken ihre ursprünglich projektierten Proportionen bezogen auf 640x480 beibehalten. Zum Beispiel wird ein in 640x480 projektiertes Quadrat dadurch bei einem Widescreen-Panel weiterhin als Quadrat dargestellt.
- Mit den Einträgen „DefaultLineHeight" und „DefaultLineSpacing" können die Standard-Zeilenhöhe (Standard: 18 Pixel) und der Standard-Zeilenabstand (Standard: 3 Pixel) bezogen auf 640x480 festgelegt werden. Diese werden immer nur dann wirksam, wenn in der Projektierung für die Position des Kurztexts oder des Ein-/Ausgabefeldes keine Y-Position oder Höhe angegeben ist.

Zyklen im Grundbild Maschine

- Einstieg in Maschine Betriebsart JOG über HS6 mit der Möglichkeit der Generierung von Zyklenaufruf ins Grundbild Maschine mit Sektion [JOBSHOPINTEGRATION]:
  ```
  [JOBSHOPINTEGRATION]
  Integration = true
  ```
  oder über der Startblock in der Projektierung
  ```
  LM("Maske1",,1)
  PRESS(VS8)
     GC("MOVE_RIDE") ; Zyklenaufruf wird generiert
     EXIT
  END_PRESS
  ```
- Beibehaltung der Operate Menüs mit Sektion [Integration]:
  Enthält der Dialog nur vertikale Softkeys, bleibt beim Aufblenden des Dialogs die horizontale Standard-Softkeyleiste inklusive Einstiegssoftkey erhalten. Enthält der Dialog horizontale und vertikale Softkeys, ersetzt die horizontale Leiste aus dem Dialog die horizontale Leiste mit dem Einstiegssoftkey.
  ```
  [Integration]
  OperateMenusEnabled = true
  ```

Auflösungsabhängige Maskenposition: Form Panels

Eintrag in "easyscreen.ini":

```
[640x480]
MyPanel = x:=0, y:=220, width:=340, height:=174
[800x480]
MyPanel = x:=0, y:=220, width:=420, height:=174
...
```

Hinweise:

Die Maskenposition kann wie folgt definiert werden:

- Die Maskenposition kann in „Pixeln bezogen auf 640x480“, angegeben werden.
  Beispiel:
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**0,219,335,174**`)`
- Die Maskenposition kann an die auflösungsabhängige Position eines FormPanels aus einem Operate Standard-Screen-Layout, z.B. „FormPanel4“ („Hilfsfunktionen“) aus dem Screen-Layout "slmastandardscreenlayout.SlMaStandardScreenLayout“ von Area „Maschine“
  `//M(MyMask/"MyCaption"/"\`
  `\myhelp.png"/`**"slmastandardscreenlayout.SlMaStandardScreenLayout.FormPanel4"**
  `)`
  gekoppelt werden.
  Damit ist es komfortabel möglich, Standard-Formen von Operate zu überblenden bzw. Easyscreen-Masken genau an deren Position zu platzieren.
  Beispiel:
  `//M(Mask/"Mask"//`**"slstandardscreenlayout.SlStandardScreenLayout.LowerForm"**`)`
  Es kann aber auch ein beliebig anderes Screen-Layout verwendet werden.
- Die Maskenposition kann an eine selbst definierte, auflösungsabhängige Definitionen von FormPanels in der Datei "easyscreen.ini" gekoppelt werden.
  Bespiel:
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**"MyPanel"**`)`

5.6 Hinweise für Umsteiger auf "Run MyScreens"

Hinweis

Beachten Sie bei der Verwendung HMI Operate in der NCU, dass auf der CF-Card alle Dateinamen in Kleinbuchstaben abgelegt werden (com, png, txt).

Hinweis

Achten Sie beim Speichern der Projektierungs- und Sprachdateien darauf, dass in dem von Ihnen verwendete Editor die Kodierung auf UTF-8 eingestellt ist.

Bilddateien

Speichern Sie Bilddateien immer im PNG-Format "*.png".

Die Daten müssen z. B. für OEM-Anpassungen unter

[System oem-Verzeichnis]/ico/[Auflösung]

abgelegt werden.

Weitere Informationen finden Sie im Kapitel Bilder/Grafik verwenden (Seite 63).

Anpassen der Projektierungsdatei

Prüfen Sie Ihre Projektierungsdateien nach folgenden Punkten:

- Einstiegssoftkeys mit den aktuell zulässigen Softkeys vergleichen und ggf. anpassen.
- Umbenennen der eingebundenen Bilddateien entsprechend Punkt "Bilddateien" oben.

Die Daten werden z. B. für OEM-Anpassungen in folgendem Verzeichnis abgelegt:

[System oem-Verzeichnis]/proj

[System user-Verzeichnis]/proj A

[System addon-Verzeichnis]/proj

Anpassen der Hilfe-Dateien

Alle Hilfe-Dateien müssen in UTF-8 gespeichert werden. Prüfen Sie Ihre vorhandenen Dateien und speichern sie diese entsprechend mit einem geeigneten Editor neu ab.

Die HTML-Dateien werden in folgendem Verzeichnis abgelegt, z. B. für Deutsch:

[System oem-Verzeichnis]/hlp/deu

[System user-Verzeichnis]/hlp/deu

[System addon-Verzeichnis]/hlp/deu

Die Verzeichnisse für weitere Sprachen müssen Sie entsprechend der Sprachkennungen anlegen.

Überprüfen der "Run MyScreens" Lizenz

Überprüfen Sie, ob die Anzahl Ihrer eingefügten Dialoge den Grundumfang von maximal 5 Dialogen übersteigt.

Um die Anzahl der Dialoge zu erweitern benötigen Sie eine der folgende Software-Optionen:

- SINUMERIK 828D/840D sl, SINUMERIK Integrate Run MyScreens (6FC5800-0AP64-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyScreens + Run MyHMI (6FC5800-0AP65-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI / 3GL (6FC5800-0AP60-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI / WinCC (6FC5800-0AP61-0YB0)

5.7 Erweiterte Projektierungssyntax

Ab SINUMERIK Operate V4.7 steht Ihnen für die Definition von Masken, Variablen, Softkeys und für Tabellen-Spalten eine vereinfachte Syntax zur Verfügung. Mit dieser alternativen Syntax wird die Lesbarkeit und Wartbarkeit verbessert. Die Eigenschaften und Attribute können in beliebiger Reihenfolge angegeben werden, Leereinträge entfallen. Gegenüber der bisherigen Syntax ist die Auflistung der Eigenschaften und Attribute statt wie bisher mit runden Klammern "**(**" und "**)**" nun durch geschweifte Klammern "**{**" und "**}**" umschlossen.

Die Eigenschaften und Attribute werden wie folgt angegeben:
`{<Name> = <Wert>, <Name> = <Wert>, …}`

Die bisherige Syntax bleibt kompatibel.

Erweiterte Syntax für die Definition von Masken

//M {<Maskenname> [,**HD**=<Überschrift>] [,**HLP**=<Grafik>] [,**X**=<X-Position>] [,**Y**=<Y-Position>] [,**W**=<Breite>] [,**H**=<Höhe>] [,**VAR**=<>System- oder Anwendervariable] [,**HLP_X**=<X-Position Hilfebild>] [,**HLP_Y**=<Y-Position Hilfebild>] [,**CM**=<Spaltenausrichtung>] [,**CB**=Verhalten beim Öffnen des Dialogs] [,**XG**=<Hilfebild als X3d-Grafik interpretieren>] [,**PANEL**=<Name des verknüpften FormPanels>][,**MC**=<Maskenhintergrundfarbe>][,**HD_AL**=<Ausrichtung des Maskenheaders>] [,**LANGFILELIST**=<Liste der maskenspezifischen Sprachdateien>]}

Beispiel:
```
//M{VariantTest, HD="My Mask"}
```

Erweiterte Syntax für die Definition von Variablen

DEF <Variablenname> = {[**TYP**=<Typ>] [,**MIN**=<Minimalwert>] [,**MAX**=<Maximalwert>] [,**TGL**=<Togglewerte>] [,**VAL**=<Vorbelegung>] [,**LT**=<Langtext>] [,**ST**=<Kurztext>] [,**GT**=<Grafiktext>] [,**UT**=<Einheitentext>] [,**TT**=<ToolTip-Text>] [,**TG**=<Toogle-Option>] [,**WR**=<Eingabemodus>] [,**AC**=<Zugriffsstufe>] [,**AL**=<Textausrichtung>] [,**FS**=<Schriftgröße>] [,**LI**=<Grenzwertbehandlung>] [,**UR**=<Aktualisierungsrate>] [,**CB**=<Verhalten beim Öffnen des Dialogs>] [,**HLP**=<Hilfebild>] [,**VAR**=<System- oder Anwendervariable>] >] [,**TXT_X**=<X-Position Kurztext>] [,**TXT_Y**=<Y-Position Kurztext>] [,**TXT_W**=<Breite Kurztext>] [,**TXT_H**=<Höhe Kurztext>] [,**X**=<X-Position Ein-/Ausgabefeld>] [,**Y**=<Y-Position Ein-/Ausgabefeld>] [,**W**=<Breite Ein-/Ausgabefeld>] [,**H**=<Höhe Ein-/Ausgabefeld>] [,**UT_DX**=<Abstand zwischen Ein-/Ausgabefeld und Einheitenfeld>] [,**UT_W**=<Breite Einheitenfeld>] [,**BC**=<Hintergrundfarbe Ein-/Ausgabefeld>] [,**FC**=<Vordergrundfarbe Ein-/Ausgabefeld>] [,**BC_ST**=<Hintergrundfarbe Kurztext>] [,**FC_ST**=<Vordergrundfarbe Kurztext>] [,**BC_GT**=<Hintergrundfarbe Grafiktext>] [,**FC_GT**=<Vordergrundfarbe Grafiktext>] [,**BC_UT**=<Hintergrundfarbe Einheitentext>] [,**FC_UT**=<Vordergrundfarbe Einheitentext>] [,**SC1**=<Signalfarbe 1 für Progressbar>] [,**SC2**=<Signalfarbe 2 für Progressbar>] [,**SVAL1**=<Schwellwert 1 für Progressbar>] [,**SVAL2**=<Schwellwert 2 für Progressbar>] [,**DT**=<Anzeigetyp>] [,**DO**=<Anzeige-Ausrichtung>] [,**OHLP**=<Online-Hilfe>][,**LINK_TGL**=<Name der verlinkten Toggle-Variable>]}

Beispiele:

```
DEF MyVar5={TYP="R2", ST="MyVar5", VAL=123.4567, OHLP="myhelp.html", MIN=100.1,
MAX=200.9}
DEF MyVar2={TYP="I", TGL="*1,2,3", VAL=1}
DEF MyVar3={TYP="R2", TGL="*0=""Aus"", 1=$80000", VAL=1}
DEF MyVar4={TYP="R2", TGL="*MyArray",VAL=1}
DEF MyVar1={TYP="R2", TGL="%grid99", X = 0, W=300, H=200}
DEF MyVar6={TYP="R2", TGL="+$80000", VAR="$R[10]", ST="Textoffset"}
```

Erweiterte Syntax für die Definition von Softkeys

SK = {[**ST**=<Beschriftung>] [,**AC**=<Zugriffsstufe>] [,**SE**=<Status>]}

Beispiele:

```
HS1={ST="""MySk""", AC=6, SE=1}
HS3={ST="SOFTKEY_CANCEL"}
HS5={ST="[$81251,""\\sk_ok.png""]"}
HS8={ST="[""Test"",""\\sk_ok.png""]"}
```

Erweiterte Syntax für die Definition von Tabellen-Spalten

{[**TYP**=<Typ>] [,**MIN**=<Minimalwert>] [,**MAX**=<Maximalwert>] [,**LT**=<Langtext>] [,**ST**=<Kurztext>] [,**WR**=<Eingabemodus>] [,**AC**=<Zugriffsstufe>] [,**AL**=<Textausrichtung>] [,**FS**=<Schriftgröße>] [,**LI**=<Grenzwertbehandlung>] [,**UR**=<Aktualisierungsrate>] [,**HLP**=<Hilfebild>] [,**VAR**=<System- oder Anwendervariable>] >] [,**W**=<Spaltenbreite>] [,**OF1**=<Offset1>] [,**OF2**=<Offset2>] [,**OF3**=<Offset3>]}

Beispiel:

```
DEF MyGridVar={TYP="R", TGL="%MyGrid1", X=10, W=550, H=100}
//G(MyGrid1/0/5)
   {TYP="I", ST="Index", WR=1, VAR="1", W=80, OF1=1}
   {TYP="S", LT="LongText2", ST="Text", WR=1, VAR="$80000", AL=2, W=330, OF1=1}
   {TYP="R3", LT="LongText1", ST="R9,R11,R13,R15", WR=2, VAR="$R[1]", W=110, OF1=2}
//END
```

5.8 SmartOperation und MutliTouch-Bedienung

Für spezifische Anpassung an SmartOperation und für die MultiTouch-Bedienung haben Sie folgende Einstellmöglichkeiten:

- Automatische Anpassung der Feldhöhe und Feldbreite auf die sogenannte minimale bedienbare Größe der Felder und den Zeilenabstand (Maskenattribut MA).
- Optimierte und pixelgenaue Streckung der Felder bei höheren Auflösungen (Maskenattribut PA).
- Setzen der Feldhöhe und des Zeilenabstands proportional zum Font (Maskenattribut FA).
- Ermöglicht das Freiscrollen von Feldern, damit die virtuelle Tastatur nicht das Eingabefeld überdeckt (Maskenattribut KM).

Weitere Details hierzu finden Sie in Kapitel Dialogeigenschaften definieren (Seite 51), Abschnitt "Programmierung".

Dialoge

6

6.1 Aufbau und Elemente eines Dialogs

6.1.1 Dialog definieren

Definition

Ein Dialog ist ein Teil einer Bedienoberfläche, die aus Titelzeile, Dialogelementen und/oder Grafik, Ausgabezeile für Meldungen sowie 8 horizontalen und 8 vertikalen Softkeys besteht.

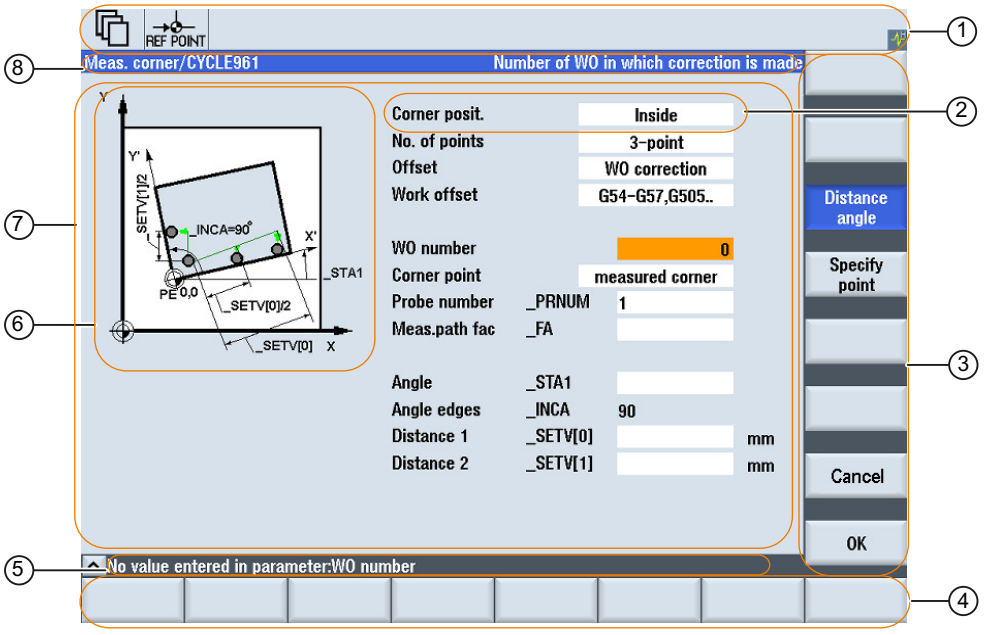
Dialogelemente sind:

- Variablen
  - Grenzwerte/Toggle-Feld
  - Vorbelegung der Variablen
- Hilfebild
- Texte
- Attribute
- System- oder Anwendervariable
- Position Kurztext
- Position Ein-/Ausgabefeld
- Farben

Eigenschaften eines Dialogs:

- Überschrift
- Grafik
- Dimension
- System- oder Anwendervariable

- Position Grafik
- Attribute



① Maschinenzustandsanzeige ("Header")
② Dialogelement
③ 8 vertikale Softkeys
④ 8 horizontale Softkeys
⑤ Ausgabe von Diagnosemeldungen
⑥ Grafik
⑦ Dialog
⑧ Titelzeile des Dialogs mit Überschrift und Langtext

Bild 6-1 Aufbau des Dialogs

Übersicht

Prinzipiell ist die Beschreibung eines Dialogs (Beschreibungsblock) wie folgt aufgebaut:

Beschreibungsblock	Kommentar	Kapitel-Verweis
`//M...`	;Anfangskennung Dialog	
`DEF Var1=...` `...`	;Variablen	Variablen (Seite 83)
`HS1=(...)` `...`	;Softkeys	Softkey-Leisten definieren (Seite 63)

`PRESS(HS1)`
`  LM...`
`END_PRESS`
	;Anfangskennung Methode
	;Aktionen
	;Endekennung Methode
		Methoden (Seite 119)

`//END`	;Endekennung Dialog	

Innerhalb des Dialog-Beschreibungsblocks werden zunächst verschiedene Variablen, die jeweils als Dialogelement im Dialog sichtbar sind, sowie horizontale und vertikale Softkeys definiert. Anschließend werden in Methoden unterschiedliche Aktionen projektiert.

Hinweis

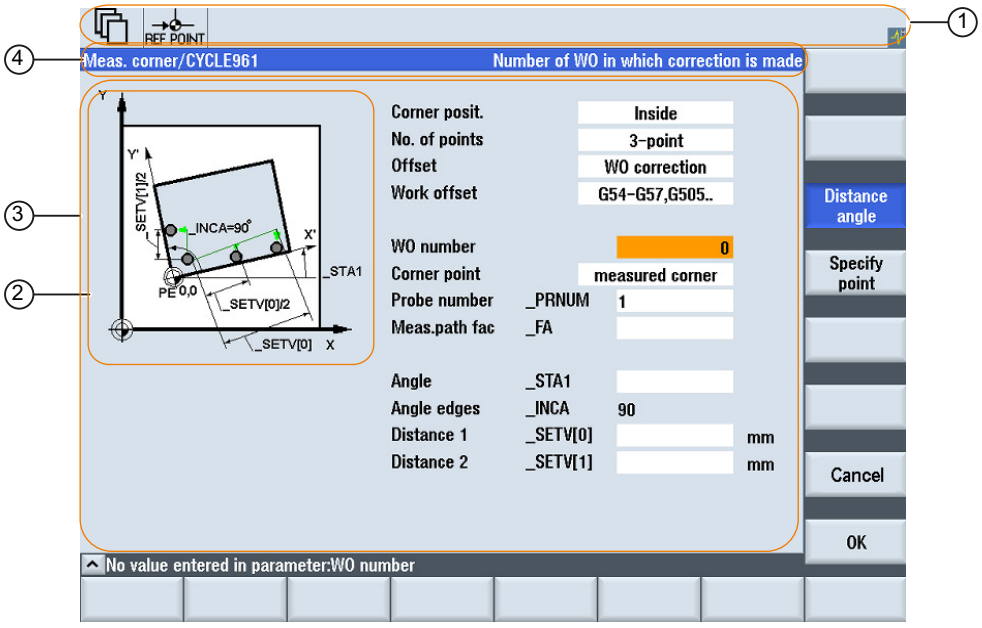
Im Bedienbereich "Maschine" ist beim Dialogwechsel folgende Reihenfolge zu beachten:

Unter der Voraussetzung, dass die Dimensionen der Folge-Maske kleiner sind als bei der vorherigen Maske oder die Position der Folge-Maske eine andere ist als die der vorherigen Maske, funktioniert der Dialogwechsel nur dann, wenn vor der Rückkehr zur ersten Maske die Folge-Maske beendet und danach die erste Maske neu geladen wird.

6.1.2 Dialogeigenschaften definieren

Beschreibung

Mit der Anfangskennung des Dialogs werden gleichzeitig die Eigenschaften des Dialogs definiert.



① Maschinenzustandsanzeige ("Header")
② Grafik
③ Dialog
④ Titelzeile des Dialogs mit Überschrift und Langtext

Bild 6-2 Eigenschaften eines Dialogs

Programmierung

Syntax:	**//M**(Bezeichner/[Überschrift]/[Grafik]/[Dimension]/[System- oder Anwendervariable]/[Position Grafik]/[Attribute]/Liste der maskenspezifischen Sprachdateien) Siehe auch Kapitel Erweiterte Projektierungssyntax (Seite 46).
Beschreibung:	Dialog definieren

Parameter:	Bezeichner			Name des Dialogs
	Überschrift			Überschrift des Dialogs als Text oder Aufruf eines Textes (z. B. $85011) aus einer sprachabhängigen Textdatei
	Grafik			Grafikdatei mit Pfad in Doppelhochkomma
	Dimension			Position und Größe des Dialogs in Pixel (Abstand von links, Abstand von oben, Breite, Höhe), bezogen auf die linke obere Ecke des Bildschirms. Die Angaben werden durch Komma getrennt.
	System- oder Anwendervariable			System- oder Anwendervariable der die aktuelle Cursorposition zugewiesen wird. Die Cursorposition kann über die System- oder Anwendervariable an die NC oder PLC gegeben werden. Die erste Variable hat den Index 1. Die Reihenfolge entspricht der Projektierreihenfolge der Variablen.
	Position Grafik			Position der Grafik (Abstand von links, Abstand von oben) in Pixel bezogen auf die linke obere Ecke des Dialogs. Die Angaben werden durch Komma getrennt.
	Attribute			Bei der Angabe der Attribute werden diese durch Komma getrennt. Mögliche Attribute sind:
		CM		Column Mode: Spaltenausrichtung
			CM0	Voreinstellung: Die Spaltenaufteilung wird für jede Zeile separat vorgenommen.
			CM1	Die Spaltenaufteilung der Zeile mit den meisten Spalten gilt für alle Zeilen.
		CB		CHANGE Block: Verhalten beim Öffnen des Dialogs: cb-Attribute, die bei einer Variablendefinition angegeben sind, haben für die Variable Vorrang vor der Pauschalangabe in der Dialogdefinition. Siehe auch AUTOHOTSPOT.
			CB0	Voreinstellung: Alle CHANGE Blöcke des Dialogs werden beim Öffnen abgearbeitet.
			CB1	CHANGE Blöcke werden nur dann abgearbeitet, wenn sich der zugehörige Wert ändert.
		XG		Einbindung einer X3D-Animation als Hilfebild (nur in der Schrittkettenprogrammierung) Beispiel: `//M(Meas/ $85605/"myx3dhelpfile.hmi,,Z_Animation,,G17"///30,10/ XG1)`
			XG0	Voreinstellung = 0
			XG1	Das Masken-Attribut XG muss bereits in der Maskendefinition auf 1 gesetzt werden, zur Laufzeit ist eine Änderung nicht mehr möglich. Hinweis: Auch die Angabe in der Masken-Eigenschaft HLP muss entsprechend gesetzt werden.
		AL		Mit dem Maskenattribut AL kann die Ausrichtung der Maskenüberschrift beeinflusst werden. Beispiel: Mittige Ausgabe der Fensterüberschrift „Set password“: `//M(MY_PWD_SET/”Set password”//"EasyPwdModalLayout"// AL2)`
			AL0	linksbündig, Voreinstellung
			AL1	rechtsbündig
			AL2	zentriert

KM		Das Freiscrollen in der Maske (SlGfwScrollArea), wenn die Virtuelle Tastatur bei MultiTouch aufgeblendet werden soll, kann folgendermaßen beeinflusst werden:
		• In der Maskendefinitionszeile mit dem Maskenattribut "KM" (Keyboard-Mode)
		Beispiel:
		`//M(MyMTMask/"MultiTouch Mask"/////KM1)`
		• Als globale Einstellung für alle Run MyScreens-Masken in der "easyscreen.ini"
		Beispiel:
		`[GENERAL]`
		`DefaultVirtualKeyboardMode=1`
		
		Hinweis:
		Wird in der Maskenprojektierung explizit das Maskenattribut KM gesetzt, so hat dies höhere Priorität als die globale Einstellung in der "easyscreen.ini".
		
		(siehe auch Kapitel SmartOperation und MutliTouch-Bedienung (Seite 48))

	KM1	Freiscrollen aktiv (default)
	KM0	Freiscrollen inaktiv
PG		Ausrichtung und Aussehen des Fenstertitels in Verbindung mit PNG Bildern (nur im Editor wirksam!) Beispiel: `//M(MyPg1SampleMask/"My PG1 Sample Mask"/////PG1)`
	PG0	(Default)
	PG1	Ausrichtung und Aussehen des Fenstertitels in Verbindung mit PNG-Bildern Pfadangabe (ganz links), Maskenname (rechts) Die Anzeige des Langtextes ist in diesem Modus nicht mehr möglich. Alternativ können Sie mit ToolTips arbeiten.

MA		Automatische Anpassung der Feldhöhe bei MultiTouch-Bedienung (MutliTouch Adjustment)
		Die Anpassung für eine MultiTouch-Bedientafel umfasst die ggf. notwendige Vergrößerung auf die sogenannte minimal bedienbare Größe (Breite, Höhe) der Felder (Ausnahmen: Progressbar, animated GIF, CustomWidget) und den Zeilenabstand.
		Die MultiTouch-Anpassung kann
		• global für alle Run MyScreens-Masken in der "easyscreen.ini" vorgenommen werden z. B.
		`[GENERAL]`
		`DefaultMultiTouchAdjustmentLevel=1`
		• oder maskenspezifisch in der Maskendefinitionszeile mit dem Maskenproperty "MA" vorgenommen werden, um dadurch die globale Einstellung in "easyscreen.ini" zu überschreiben. z. B
		`//M(MyMTMask/"MultiTouch Mask"/////MA1)`
		Wird in der Maskenprojektierung explizit das Maskenattribut MA gesetzt, so hat dies höhere Priorität als die globale Einstellung in der "easyscreen.ini".
		(siehe auch Kapitel SmartOperation und MutliTouch-Bedienung (Seite 48))

	MA0	Es findet keine Anpassung statt

	MA1	Nur die Felder werden angepasst, für die kein Abstand von oben und/oder keine Feldhöhe/Feldbreite projektiert ist – also die Felder, die Default-Positionen nutzen und somit von Run MyScreens den Abstand von oben und/oder die Feldhöhe/Feldbreite berechnen lassen. (Default)
	MA2	Es werden alle Felder angepasst unabhängig von den projektierten Feldgrößen.

PA		Optimierte und pixelgenaue Streckung der Felder bei höheren Auflösungen (Pixel Fine Adjustment)
		Bisher werden zunächst alle Positionen der Felder bezogen auf 640x480 berechnet und anschließend mit den entsprechenden horizontalen und vertikalen Streckungsfaktoren gezoomt. Dieses Verhalten hat aber den Nachteil, dass es bei "ungünstigen Streckungsfaktoren" zu Rundungsfehlern kommen kann. So kann z. B. die Feldhöhe oder der Zeilenabstand von Zeile zu Zeile um ein Pixel "springen". Um dies zu verhindern, gibt es den Modus "Pixel Fine Adjustment", bei dem die Positionen der Felder direkt in der aktueller Auflösung bestimmt werden.
		`//M(MyMask/"My Mask"/////PA1)`
		Diese Einstellung kann auch global für alle Run MyScreens-Masken in der "easyscreen.ini" vorgenommen werden z.B.
		`[GENERAL]`
		`DefaultPixelFineAdjustment=1`
		Wird in der Maskenprojektierung explizit das Maskenattribut PA gesetzt, so ist dies höherprior als die globale Einstellung in der "easyscreen.ini".
		Wird eine Maske mit Font Adjustment = FA1 dargestellt, so ist für diese Maske auch automatisch der Pixel Fine Adjustment Modus aktiv.
		(siehe auch Kapitel SmartOperation und MutliTouch-Bedienung (Seite 48))

	PA0	Streckungsverhalten wie bisher, d.h. Positionen werden bezogen auf 640x480 berechnet und dann gestreckt. (wegen Kompatibilität: Default)
	PA1	Optimierte und pixelgenaue Streckung der Felder bei höheren Auflösungen

FA		Feldhöhe und Zeilenabstand proportional zum Font setzten (Font Adjustment)
		Dieses Feature kann auch global für alle Run MyScreens-Masken in der "easyscreen.ini" vorgenommen werden z.B.
		`[GENERAL]`
		`DefaultFontAdjustment=1`
		Wird in der Maskenprojektierung explizit das Maskenattribut FA gesetzt, so hat dies höhere Priorität als die globale Einstellung in der "easyscreen.ini".
		Die Einstellungen für die DefaultLineHeight und DefaultLineSpacing haben beim aktiven Font Adjustment Modus keine Bedeutung.
		(siehe auch Kapitel SmartOperation und MutliTouch-Bedienung (Seite 48))

	FA0	Feldhöhe und Zeilenabstand werden wie bisher gestreckt (wegen Kompatibilität: Default)
	FA1	Feldhöhe und Zeilenabstand werden proportional zum Font gesetzt (setzt automatisch Maskeattribut PA=1)
	FA2	gleich wie FA1, jedoch werden zusätzlich die X-Koordinate und die Feldbreite proportional zum Font gestreckt. (setzt automatisch Maskeattribut PA=1) (Nur in Verbindung mit dem Attribut XG=1 möglich!)
NT		Navigationstyp (Navigation Type)

			NT0	NT0 = Standardmodus Die Navigation erfolgt analog zu den Masken des Bedienbereichs, in welchem die Run MyScreens Maske eingebunden wird. D. h. in den Zyklenmasken im Editor erfolgt die Navigation zeilenorientiert (siehe NT2), in allen anderen "normalen" Masken geometrisch (siehe NT1).
			NT1	geometrische Navigation (oben, unten, links, rechts), bis die Ränder der Maske erreicht werden (Standard in normaler Run MyScreens Umgebung z.B. Area "Custom")
			NT2	Zeilenorientiert, d. h. innerhalb der Zeile nach rechts bis das Ende der Zeile erreicht wird (Standard in den Zyklenmasken des Editors)
		NR		Navigationsrichtung bei Drücken der Enter-Taste (Return Navigate Direction)
			NR0	NR0 = aus (default), d. h. bei Drücken der Enter-Taste erfolgt die Navigation innerhalb der Maske in tabularischer Reihenfolge (Standard in normaler Run MyScreens Umgebung)
			NR1	Bei Drücken der Enter-Taste verhält sich die Navigation wie bei Drücken der Pfeiltaste nach oben
			NR2	wie NR1, jedoch nach unten
			NR3	wie NR1, jedoch nach links
			NR4	wie NR1, jedoch nach rechts
		TA		Tooltip-Ausrichtung
			TA0	automatisch (default)
			TA1	links
			TA2	rechts
			TA3	oben
			TA4	unten
			TA5	links oben
			TA6	links unten
			TA7	rechts oben
			TA8	rechts unten

	MC			Maskenhintergrundfarbe (Mask Color)
				Beispiel:
				Maskenhintergrundfarbe blau (= 6) setzen
				`//M(MyMask/"MyMask"//////6)`
				oder
				`PRESS(HS1)`
				`  MC=6`
				`END_PRESS`

	Liste der maskenspezifischen Sprachdateien			durch Komma getrennt

Zugriff auf die Dialogeigenschaften

Innerhalb von Methoden (z. B. PRESS Block) kann auf folgende Eigenschaften des Dialogs lesend und schreibend zugegriffen werden:

- HD = Überschrift (Header)
- HLP = Hilfebild

- VAR = System- oder Anwendervariable
- MC = Maskenhintergrundfarbe
- CM = Spaltenausrichtung (nur lesend)
- CB = Verhalten beim Öffnen (nur lesend)
- XG = Einbindung X3d (nur lesend)
- AL = Ausrichtung Maskenüberschrift (nur lesend)

Beispiel

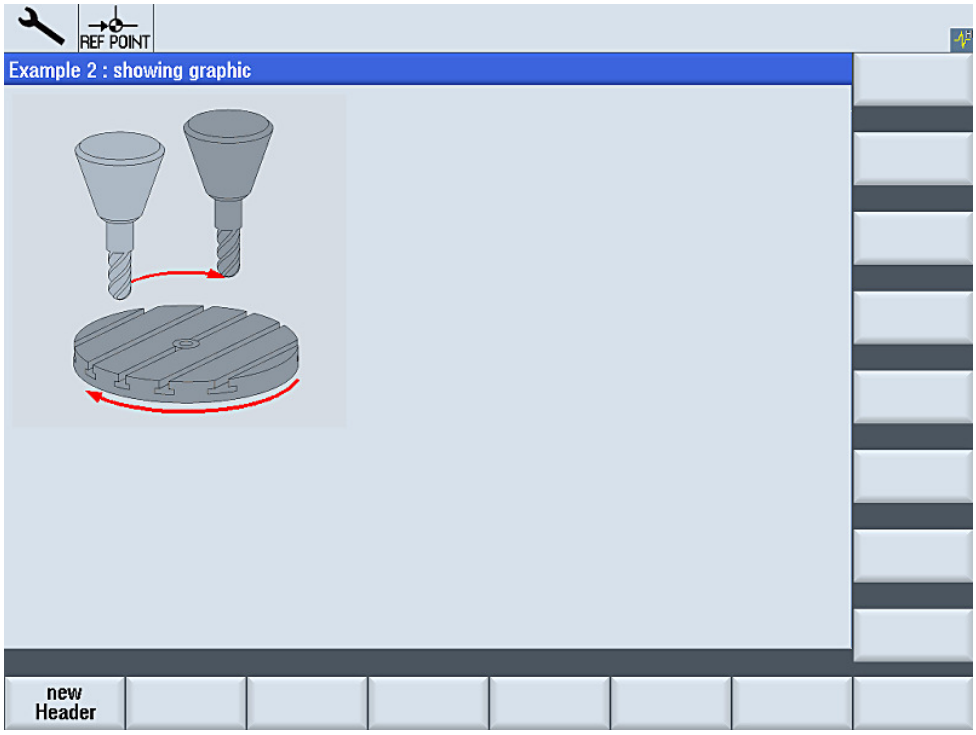


Bild 6-3 "Example 2: showing graphic"

```
//S(Start)
HS7=("Example", se1, ac7)

PRESS(HS7)
  LM("Mask2")
END_PRESS

//END
//M(Mask2/"Example 2 : showing graphic"/"example.png")
HS1=("new%nHeader")
HS2=("")
HS3=("")
```

```
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")

PRESS(HS1)
  Hd= "new Header"
END_PRESS
...
//END
```

Siehe auch

Programmierbeispiel für den Bereich "Custom" (Seite 276)

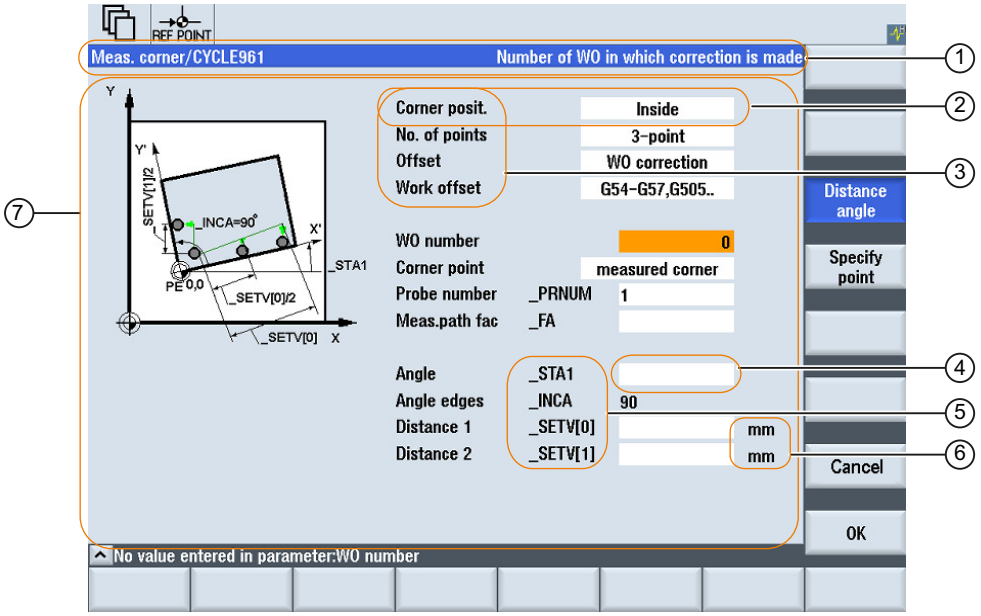
6.1.3 Dialogelemente definieren

Dialogelement

Als Dialogelement wird der sichtbare Teil einer Variablen bezeichnet, d.h. Kurztext, Grafiktext, Ein-/Ausgabefeld, Einheitentext und ToolTip. Dialogelemente füllen im Dialoghauptteil Zeilen. Pro Zeile können ein oder mehrere Dialogelemente definiert werden.

Eigenschaften von Variablen

Alle Variablen sind nur im aktiven Dialog gültig. Mit der Definition einer Variablen werden dieser Eigenschaften zugewiesen. Innerhalb von Methoden (z. B. einer PRESS-Methode) kann auf die Werte der Dialogeigenschaften zugegriffen werden.



① Titelzeile des Dialogs mit Überschrift und Langtext
② Dialogelement
③ Kurztext
④ Ein-/Ausgabefeld
⑤ Grafiktext
⑥ Einheitentext
⑦ Dialoghauptteil

Bild 6-4 Elemente eines Dialogs

Programmierung - Übersicht

In runden Klammern stehen die durch Komma zu trennenden Einzelparameter:

DEF [Bezeichner] =	**Bezeichner = Name der Variablen**
	Variablentyp
	/[Grenzwerte oder Toggle-Feld]
	/[Vorbelegung]
	/[Texte(Langtext, Kurztext|Bild, Grafiktext, Einheitentext)]
	/[Attribute]
	/[Hilfebild]
	/[System- oder Anwendervariable]
	/[Position Kurztext]
	/[Position Ein-/Ausgabefeld(Left, Top, Width, Height)]

	/[Farben]
	/[Online-Hilfe] (Seite 72)

Siehe auch

Parameter von Variablen (Seite 91)

6.1.4 Mehrspaltige Dialoge definieren

Übersicht

In einem Dialog können in einer Zeile auch mehrere Variablen dargestellt werden. Die Variablen werden in diesem Fall in der Projektierungsdatei alle innerhalb einer Definitionszeile definiert.

```
DEF VAR11 = (S///"Var11"), VAR12 = (I///"Var12")
```

Um die einzelnen Variablen in der Projektierungsdatei besser lesbar darzustellen, können die Definitionszeilen nach jeder Variablendefinition und nachfolgendem Komma umgebrochen werden.

Das Schlüsselwort "DEF" bezeichnet immer den Beginn einer neuen Zeile:

```
DEF Tnr1=(I//1/"","T ",""/wr1///,,10/20,,50),
   TOP1=(I///,"Typ="/WR2//"$TC_DP1[1,1]"/80,,30/120,,50),
   TOP2=(R3///,"L1="/WR2//"$TC_DP3[1,1]"/170,,30/210,,70),
   TOP3=(R3///,"L2="/WR2//"$TC_DP4[1,1]"/280,,30/320,,70),
   TOP4=(R3///,"L3="/WR2//"$TC_DP5[1,1]"/390,,30/420,,70)
DEF Tnr2=(I//2/"","T ",""/wr1///,,10/20,,50),
   TOP21=(I///,"Typ="/WR2//"$TC_DP1[2,1]"/80,,30/120,,50),
   TOP22=(R3///,"L1="/WR2//"$TC_DP3[2,1]"/170,,30/210,,70),
   TOP23=(R3///,"L2="/WR2//"$TC_DP4[2,1]"/280,,30/320,,70),
   TOP24=(R3///,"L3="/WR2//"$TC_DP5[2,1]"/390,,30/420,,70)
```

Hinweis

Beachten Sie bei der Gestaltung mehrspaltiger Dialoge bitte, dass sehr viele Spalten das System ggf. verlangsamen können!

6.1.5 Kennwort-Dialoge

Verwendung der Standard Kennwort-Dialoge

Folgende vordefinierten Kennwort-Dialoge können Sie in die Maskenprojektierung einbinden:

- Kennwort setzen

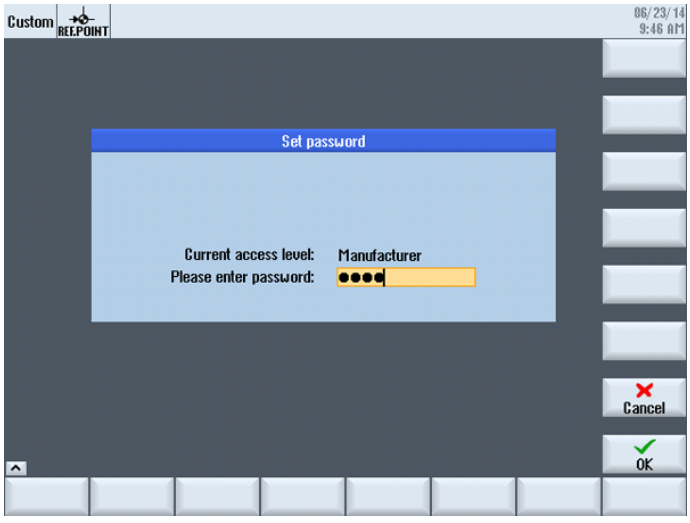


Bild 6-5 Dialog Kennwort setzen

- Kennwort ändern

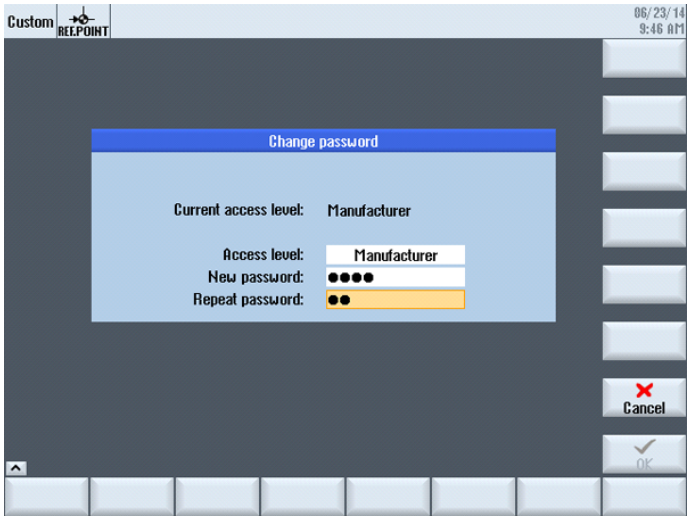


Bild 6-6 Dialog Kennwort ändern

- Kennwort löschen

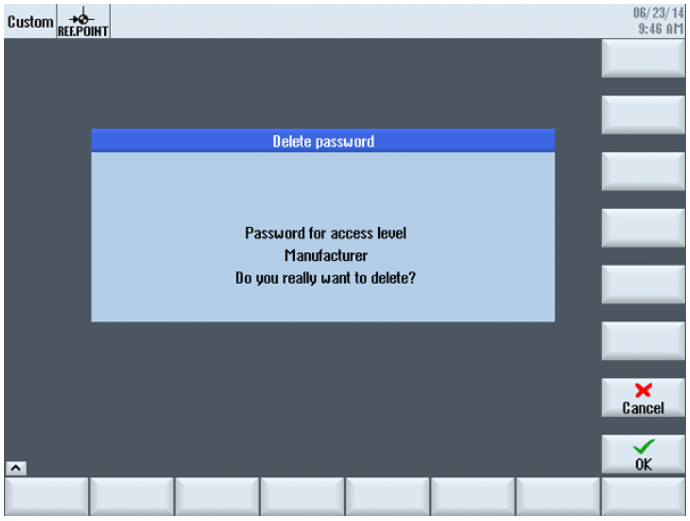


Bild 6-7 Dialog Kennwort löschen

Hinweis

Mit diesen Dialogen wird Ihnen die Kennwort-Funktionalität bereitgestellt. Die Dialoge entsprechen nicht den Dialogen von SINUMERIK Operate.

Der Aufruf der Dialoge kann entweder über die Funktion Load Softkey (LS) oder über die Funktion Load Mask (LM) erfolgen:

1. Aufruf über die Funktion Load Softkey (LS):
   `LS("Passwd","slespasswd.com",1)`
   Erweiterung der vertikalen Softkeys:
   - Softkey 1: Kennwort setzen
   - Softkey 2: Kennwort ändern
   - Softkey 3: Kennwort löschen
2. Alternativ können die drei Masken auch direkt durch den Aufruf der Funktion Load Mask (LM) angesprungen werden:
   - Kennwort setzen: `LM("PWD_SET", "slespasswd.com",1)`
   - Kennwort ändern: `LM("PWD_CHG", "slespasswd.com",1)`
   - Kennwort löschen: `LM("PWD_CLEAR", "slespasswd.com",1)`

6.1.6 Bilder/Grafik verwenden

Verwendung von Grafik

Es sind zu unterscheiden:

- Bilder/Grafik im Grafikbereich
- Hilfebilder, die zum Beispiel einzelne Variablen illustrieren und die im Grafikbereich überblendet werden.
- Weitere Hilfebilder können statt Kurztext oder Ein-/Ausgabefeld projektiert werden, die frei positionierbar sind.

Ablageorte

Das zur Auflösung des angeschlossenen Monitors passende Bild wird in den zugehörigen Auflösungsverzeichnissen in folgender Reihenfolge gesucht:

[System user-Verzeichnis]`/ico/ico<Auflösung>`
[System oem-Verzeichnis]`/ico/ico<Auflösung>`
[System addon-Verzeichnis]`/ico/ico<Auflösung>`

Wird das Bild nicht angezeigt oder nicht gefunden, kopieren Sie es in eines der folgenden Verzeichnisse für die Auflösung 640 x 480 Pixel:

[System user-Verzeichnis]`/ico/ico640`
[System oem-Verzeichnis]`/ico/ico640`
[System addon-Verzeichnis]`/ico/ico640`

Hinweis

Bei den unterschiedlichen Panel-Auflösungen werden die Bilder proportional positioniert.

6.2 Softkey-Leisten definieren

Definition

Alle horizontalen und alle vertikalen Softkeys werden jeweils zusammen als Softkey-Leiste bezeichnet. Zusätzlich zu den bestehenden Softkey-Leisten können weitere Leisten definiert werden, die bestehende Leisten teilweise oder ganz überschreiben.

Die Namen der Softkeys sind festgelegt. Es müssen nicht alle Softkeys belegt sein.

HSx x 1 - 8, Horizontale Softkeys 1 bis 8

VSy y 1 - 8, Vertikale Softkeys 1 bis 8

Prinzipiell ist die Beschreibung einer Softkey-Leiste (Beschreibungsblock) wie folgt aufgebaut:

Beschreibungsblock	Kommentar	Kapitel-Verweis
`//S...`	;Anfangskennung Softkey-Leiste	
`HSx=...`	;Softkeys definieren	

`PRESS(HSx)`
`  LM...`
`END_PRESS`
	;Anfangskennung Methode
	;Aktionen
	;Endekennung Methode
		siehe Kapitel Methoden (Seite 119)

`//END`	;Endekennung Softkey-Leiste	

Beschreibung

Mit der Definition der Softkey-Leiste werden einem Softkey zugleich Eigenschaften zugewiesen.

Programmierung

Syntax:	*//S(Bezeichner)* ... //END	;Anfangskennung der Softkey-Leiste ;Endekennung der Softkey-Leiste
	Siehe auch Kapitel Erweiterte Projektierungssyntax (Seite 46).	
Beschreibung:	Softkey-Leiste definieren	
Parameter:	Bezeichner	Name der Softkey-Leiste
	Text oder Bilddateiname	
Syntax:	**SK** = (Text[, Zugriffsstufe][, Status][, Ausrichtung des Softkey-Bildes][, Textausrichtung bezogen auf Softkey-Bild])	
Beschreibung:	Softkey definieren	
Parameter:	SK	Softkey, z. B. HS1 bis HS8, VS1 bis VS8

	Text	Text angeben	
	Bilddateiname	"\\my_pic.png" oder über separate Textdatei $85199 z. B. mit folgendem Text in der (sprachabhängigen) Textdatei: 85100 0 0 "\\my_pic.png". Die Bildgröße für die Darstellung auf einem Softkey ist abhängig vom verwendeten OP:	
		OP 08:	640 x 480 mm → 25 x 25 Pixel
		OP 010:	640 x 480 mm → 25 x 25 Pixel
		OP 012:	800 x 600 mm → 30 x 30 Pixel
		OP 015:	1024 x 768 mm → 40 x 40 Pixel
		OP 019:	1280 x 1024 mm → 72 x 72 Pixel
	Zugriffsstufe	ac0 bis ac7 (ac7: Voreinstellung)	
	Status	se1: sichtbar (Voreinstellung) se2: nicht bedienbar (graue Schrift) se3: hervorgehoben (zuletzt bedienter Softkey)	
	Ausrichtung des Softkey-Bildes	PA (PictureAlignment)	
		Gültige Werte: 0: links 1: rechts 2: zentriert 3: oben (Standard) 4: unten	
	Textausrichtung bezogen auf Softkey-Bild	TP (TextAlignedToPicture)	
		Gültige Werte: 0: Text wird nicht am Bild ausgerichtet 1: Text wird am Bild ausgerichtet (Standard)	

Hinweis

Bei der Softkey-Beschriftung wird ein Zeilenumbruch mit %n erzeugt.

Es stehen maximal 2 Zeilen zu je 9 Zeichen zur Verfügung.

Zugriffsstufe zuordnen

Der Bediener hat nur Zugang zu Informationen, die dieser Zugriffsstufe und den jeweils niedrigeren Zugriffsstufen entsprechen (siehe auch AUTOHOTSPOT.)

Beispiel

```
//S(Leiste1)                                ; Anfangskennung der Softkey-Leiste
HS1=("NEU", ac6, se2)                       ; Softkey HS1 definieren, die Beschriftung "NEU", die
                                            Schutzstufe 6 und den Status "nicht bedienbar" zuweisen
```

```
HS2=("\\bild1.png")                          ; Softkey eine Grafik zuweisen
HS3=("Exit")
HS4=(["Confirm","\\sk_ok.png"],PA0,TP1)      ; Softkey mit Text und Grafik, Text="Confirm",
                                               Bild="sk_ok.png", Ausrichtung des Softkey-Bildes:
                                               links, Text wird am Bild ausgerichtet


VS1=("Untermaske")
VS2=($85011, ac7, se2)                       ; Softkey VS2 definieren, Text aus Sprachdatei, Schutz-
                                               stufe 1 und den Status "nicht bedienbar" zuweisen
VS3=("Abbruch", ac1, se3)                    ; Softkey VS3 definieren, die Beschriftung "Abbruch",
                                               die Schutzstufe 1 und den Status "hervorgehoben" zuwei-
                                               sen
VS4=("OK", ac6, se1)                         ; Softkey VS4 definieren, die Beschriftung "OK", die
                                               Schutzstufe 6 und den Status "sichtbar" zuweisen
VS5=(SOFTKEY_CANCEL,,se1)                    ; Abbruch Standard-Softkey VS5 definieren und den Sta-
                                               tus "sichtbar" zuweisen
VS6=(SOFTKEY_OK,,se1)                        ; OK Standard-Softkey VS6 definieren und den Status
                                               "sichtbar" zuweisen
VS7=(["\\bild1.png","OEM Text"],,se1)        ; Softkey VS7 definieren, Grafik zuweisen, die Be-
                                               schriftung "OEM Text" und den Status "sichtbar" zuwei-
                                               sen
VS8=(["\\bild1.png",$83533],,se1)            ; Softkey VS8 definieren, Grafik zuweisen, Text aus
                                               Sprachdatei und den Status "sichtbar" zuweisen


PRESS(HS1)                                   ; Anfangskennung der Methode
  HS1.st="Berechnen"                         ; Softkey einen Beschriftungstext zuweisen
...
END_PRESS                                    ; Endekennung der Methode

PRESS(RECALL)                                ; Anfangskennung der Methode
  LM("Maske21")                              ; Dialog laden
END_PRESS                                    ; Endekennung der Methode
//END                                        ; Endekennung der Softkey-Leiste
```

6.2.1 Eigenschaften von Softkeys zur Laufzeit ändern

Beschreibung

Die Eigenschaften Text, Zugriffsstufe und Status eines Softkeys können zur Laufzeit in den Methoden verändert werden.

Programmierung

Syntax:	SK.st = "Text"		;Softkey mit Beschriftung
	SK.ac = Zugriffsstufe		;Softkey mit Schutzstufe
	SK.se = Status		;Softkey mit Status
	SK.pa = "Ausrichtung des Softkey-Bildes"		;Softkey mit Bild
	SK.tp = "Textausrichtung bezogen auf Softkey-Bild"		;Softkey mit Bild und Beschriftung
Beschreibung:	Eigenschaften zuweisen		
Parameter:	Text		Beschriftungstext in Hochkommas
	Zugriffsstufe		Wertebereich: 0 ... 7
	Status	1:	sichtbar und bedienbar
		2:	nicht bedienbar (graue Schrift)
		3:	hervorgehoben (zuletzt bedienter Softkey)
	Ausrichtung des Softkey-Bilde	0:	links
		1:	rechts
		2:	zentriert
		3:	oben (Standard)
		4:	unten
	Textausrichtung bezogen auf Softkey-Bild	0:	Text wird nicht am Bild ausgerichtet
		1:	Text wird am Bild ausgerichtet (Standard)

Beispiel

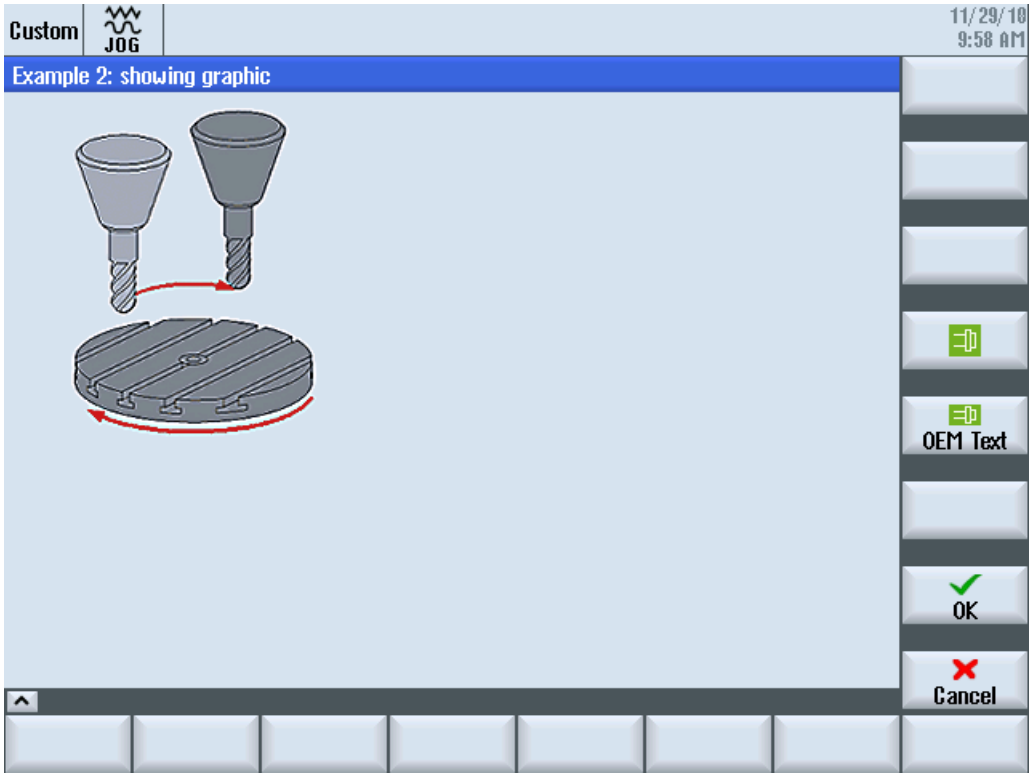


Bild 6-8 Beispiel 3: Grafik und Softkeys

```
//S(Start)
HS7=("Example", ac7, se1)

PRESS(HS7)
  LM("Maske3")
END_PRESS

//END

//M(Maske3/"Example 2: showing graphic"/"example.png")
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
```

```
VS3=("")
VS4=("\\sp_ok.png",,SE1)
VS5=(["\\sp_ok_small.png","OEM Text"],,SE1)
VS6=("")
VS7=(SOFTKEY_OK,,SE1)
VS8=(SOFTKEY_CANCEL,,SE1)
PRESS(VS4)
  EXIT
END_PRESS
PRESS(VS5)
  EXIT
END_PRESS
PRESS(VS7)
  EXIT
END_PRESS
PRESS(VS8)
  EXIT
END_PRESS
//END
```

6.2.2 Sprachabhängiger Text

Übersicht

Sprachabhängige Texte werden verwendet für:

- Softkey-Beschriftungen
- Überschriften
- Hilfetexte
- andere beliebige Texte

Die sprachabhängigen Texte für Dialoge werden in Textdateien abgelegt.

Die Textdateien befinden sich in folgenden Verzeichnissen:

- [System user-Verzeichnis]/lng
- [System oem-Verzeichnis]/lng
- [System addon-Verzeichnis]//lng

Hinweis

Die Textdateien müssen analog zu den Projektdateien abgelegt werden.

z. B.:

[System user-Verzeichnis]/lng/[Textdatei]

[System user-Verzeichnis]/proj/[Projektierungsdatei]

alsc.txt	sprachabhängige Texte für die Siemens-Standard-Zyklen
almc.txt	sprachabhängige Texte für die Siemens-Messzyklen

Die zur Programmlaufzeit verwendeten Textdateien werden in der Datei "easyscreen.ini" angegeben:

```
[LANGUAGEFILES]
LngFile03 = user.txt      ;->user<_xxx>.txt (z. B: user_eng.txt)
```

Die Datei "user.txt" ist hier als Beispiel für eine Textdatei gewählt. Der Name ist grundsätzlich frei wählbar. Je nach Sprache der in der Datei enthaltenen Texte, muss noch das passende Sprachkürzel gemäß folgender Syntax angehängt werden. Nach dem Namen wird ein Unterstrich und nachfolgend die entsprechende Sprachkennung angehängt, z. B. "user_eng.txt".

Hinweis

LngFile01 und LngFile02 dürfen Sie für Anwendertexte nicht verwenden, da diese in der Standard "easyscreen.ini" vergeben sind.

Siehe auch

AUTOHOTSPOT

Maskenspezifische Sprachdateien

Um Überschneidungen bei den verwendeten Nummernbändern in Sprachdateien zu vermeiden, sollen nur bestimmte Sprachdateien in einer Maske verwendet werden können.

In der Maskendefinitionszeile werden deshalb die zu verwendenden Sprachdateien mit Komma getrennt aufgeführt. Die höchste Priorität hat die an letzter Stelle aufgeführte Datei (analog zur Logik in der "easyscreen.ini").

Alle in der Maske verwendeten sprachabhängigen Texte werden vorrangig in diesen Sprachdateien gesucht. Wird der Text darin nicht gefunden, wird anschließend noch in den in der "easyscreen.ini" projektierten Textdateien gesucht.

Werden in der Maske keine Sprachdateien definiert, wird nur in den in der "easyscreen.ini" angegebenen Sprachdateien gesucht.

Hinweis

Ist eine Sprachdatei in der aktuell angewählten Sprache nicht vorhanden, so wird die englische Sprachdatei verwendet (default).

Beispiele:

```
//M(MyMask/$85597///// //"mytexts.txt, general.txt, mask.txt")
DEF ….
```

Dieses Verfahren können Sie auch für die Einstiegssoftkeys nutzen:

```
//S(Start, "mytexts.txt, general.txt, mask.txt")
VS1=($85597)
PRESS(VS1)
   LM("MyMask","masks.com")
END_PRESS
```

Format der Textdateien

Die Textdateien müssen im Format UTF-8 kodiert gespeichert werden.

Hinweis

Achten Sie beim Speichern der Projektierungs- und Sprachdateien darauf, dass in dem von Ihnen verwendete Editor die Kodierung auf UTF-8 eingestellt ist.

Form eines Texteintrags

Syntax:	8xxxx 0 0 "Text"		
Beschreibung:	Zuordnung zwischen Textnummer und Text in der Datei		
Parameter:	xxxx	85.000 bis 89.999 900.000 bis 999.999	Für Anwender reservierter Textidentifikations-nummern-Bereich. Die Nummern müssen ein-deutig vergeben werden.
	"Text"		Text, der im Dialog erscheint
	%n		Steuerzeichen im Text für Zeilenumbruch

Die beiden durch Leerzeichen getrennten Parameter 2 und 3 sind Steuerzeichen für die Alarmtext-Ausgabe. Sie müssen wegen der Einheitlichkeit des Textformats mit den Alarmtexten auf jeden Fall Null sein.

Beispiele für sprachabhängige Texte:

```
85000 0 0 "Rückzugsebene"
85001 0 0 "Bohrtiefe"
85002 0 0 "Gewindesteigung"
85003 0 0 "Taschenradius"
```

6.3 Online-Hilfe projektieren

6.3.1 Übersicht

Über die bereits bestehende umfangreiche Online-Hilfe hinaus haben Sie die Möglichkeit eine herstellerspezifische Online-Hilfe zu erstellen und diese in SINUMERIK Operate einzubinden.

Diese Online-Hilfe wird im HTML-Format erstellt, d. h. sie besteht aus untereinander verlinkten HTML-Dokumenten. Das gesuchte Thema wird in einem gesonderten Fenster über ein Inhalts- oder Stichwortverzeichnis aufgerufen. Ähnlich wie in einem Dokumenten-Browser (z. B. Windows-Explorer), wird in der linken Fensterhälfte eine Auswahlübersicht angezeigt und wenn Sie das gewünschte Thema anklicken, wird die Erläuterung dazu in der rechten Fensterhälfte angezeigt.

Eine kontextsensitive Anwahl von Online-Hilfeseiten ist nicht möglich.

Vorgehensweise

1. HTML-Dateien erzeugen
2. Hilfebuch erzeugen
3. Online-Hilfe in SINUMERIK Operate einbinden
4. Hilfedateien ablegen

Weitere Anwendungsfälle

Es können Online-Hilfen zu folgenden OEM-spezifischen Erweiterungen erstellt und zum SINUMERIK Operate Online-Hilfe-System ergänzt werden:

- Online-Hilfe zu **Zyklen und/oder M-Funktionen** des Maschinenherstellers, die die Programmiermöglichkeiten der SINUMERIK-Steuerungen erweitern. Diese Online-Hilfe wird genauso aufgerufen SINUMERIK Operate Online-Hilfe "Programmieren".
- Online-Hilfe zu OEM-spezifischen **Variablen** des Maschinenherstellers. Diese Online-Hilfe wird aus der Variablenansicht von SINUMERIK Operate aufgerufen.

Online-Hilfe programmieren

Für weitere Gestaltungsmöglichkeiten der Online-Hilfe können Sie das "SINUMERIK HMI Programmierpaket sl" verwenden. Mit diesem Programmierpaket ist die Entwicklung von Hochsprachenapplikationen in der Programmiersprache C++ für SINUMERIK Operate auf der NCU 7x0 möglich.

Hinweis

Das "SINUMERIK HMI Programmierpaket sl" ist eine separat zu bestellende Software-Option. Die dazugehörige Dokumentation erhalten Sie zusammen mit dem Programmierpaket.

6.3.2 HTML-Dateien erzeugen

Erzeugen Sie die Hilfedateien im HTML-Format. Es ist dabei möglich alle Informationen in einer einzelnen HTML-Datei abzulegen oder in mehrere HTML-Dateien zu separieren.

Die Dateinamen können Sie selbst vergeben, müssen aber Folgendes beachten:

- Verweise innerhalb der HTML-Dateien sollen immer mit relativen Pfaden angeben. Nur so ist sichergestellt, dass die Verweise sowohl auf dem Entwicklungsrechner, als auch auf dem Zielsystem gleichermaßen funktionieren.
- Wenn zu bestimmten Punkten innerhalb einer HTML-Datei per Link gesprungen werden soll, müssen dafür sogenannte Anker definiert werden.
  Beispiel für einen HTML-Anker:
  `<a name="myAnchor">This is an anchor</a>`
- Der Inhalt der HTML-Dokumente muss mit der Codierung UTF-8 abgelegt werden. Erst dadurch ist gewährleistet, dass die HTML-Dokumente von SINUMERIK Operate in allen unterstützten Landessprachen korrekt angezeigt werden.
- Es werden folgende Untermengen des HTML-Funktionsumfangs unterstützt:

HTML-Tags

Tag	Beschreibung	Kommentar
a	Anchor or link	Unterstützte Attribute: href und name
address	Address	
big	Larger font	
blockquote	Indented paragraph	
body	Document body	Unterstützte Attribute: bgcolor (#RRGGBB)
br	Line break	
center	Centered paragraph	
cite	Inline citation	Gleiche Wirkung wie Tag i
code	Code	Gleiche Wirkung wie Tag tt
dd	Definition data	
dfn	Definition	Gleiche Wirkung wie Tag i
div	Document division	Unterstützt werden die Standard-Satzattribute

Tag	Beschreibung	Kommentar
dl	Definition list	Unterstützt werden die Standard-Satzattribute
dt	Definition term	Unterstützt werden die Standard-Satzattribute
em	Emphasized	Gleiche Wirkung wie Tag i
font	Font size, family, color	Unterstützte Attribute: size, face, and color (#RRGGBB)
h1	Level 1 heading	Unterstützt werden die Standard-Satzattribute
h2	Level 2 heading	Unterstützt werden die Standard-Satzattribute
h3	Level 3 heading	Unterstützt werden die Standard-Satzattribute
h4	Level 4 heading	Unterstützt werden die Standard-Satzattribute
h5	Level 5 heading	Unterstützt werden die Standard-Satzattribute
h6	Level 6 heading	Unterstützt werden die Standard-Satzattribute
head	Document header	
hr	Horizontal line	Unterstützte Attribute: width (kann als absoluter oder relativer Wert angegeben werden)
html	HTML document	
i	Italic	
img	Image	Unterstützte Attribute: src, width, height
kbd	User-entered text	
meta	Meta-information	
li	List item	
nobr	Non-breakable text	
ol	Ordered list	Unterstützt werden die die Standardattribute für Listen
p	Paragraph	Unterstützt werden die Standard-Satzattribute (Voreinstellung: left-aligned)
pre	Preformated text	
s	Strikethrough	
samp	Sample code	Gleiche Wirkung wie Tag tt
small	Small font	
span	Grouped elements	
strong	Strong	Fließtext wird stark betont, ersetzt Tag b
sub	Subscript	
sup	Superscript	
table	Table	Unterstützte Attribute: border, bgcolor (#RRGGBB), cellspacing, cellpadding, width (absolut oder relative), height
tbody	Table body	Wirkungslos
td	Table data cell	Unterstützt werden die die Standardattribute für Tabellenzellen
tfoot	Table footer	Wirkungslos
th	Table header cell	Unterstützt werden die die Standardattribute für Tabellenzellen
thead	Table header	Wird für das Drucken von Tabellen verwendet, die sich über mehrere Seiten erstrecken
title	Document title	
tr	Table row	Unterstützte Attribute: bgcolor (#RRGGBB)
tt	Typewrite font	

Tag	Beschreibung	Kommentar
u	Underlined	
ul	Unordered list	Unterstützt werden die die Standardattribute für Listen
var	Variable	Gleiche Wirkung wie Tag tt

Satzattribute

Folgende Attribute werden von den Tags div, dl, dt, h1, h2, h3, h4, h5, h6, p unterstützt:

- align (left, right, center, justify)
- dir (ltr, rtl)

Standardattribute für Listen

Folgende Attribute werden von den Tags ol und ul unterstützt:

- type (1, a, A, square, disc, circle)

Standardattribute für Tabellen

Folgende Attribute werden von den Tags td und th unterstützt:

- width (absolute, relative, no-value)
- bgcolor (#RRGGBB)
- colspan
- rowspan
- align (left, right, center, justify)
- valign (top, middle, bottom)

CSS-Eigenschaften

Die nachfolgende Tabelle enthält den unterstützten CSS-Funktionsumfang:

Property	Werte	Beschreibung
background-color	<color>	Hintergrundfarbe für Elemente
background-image	<uri>	Hintergrundbild für Elemente
color	<color>	Vordergrundfarbe für Text
text-indent	<length>px	Einrückung der ersten Zeile eines Absatzes in Pixel
white-space	normal | pre | nowrap | pre-wrap	Bestimmt wie Whitespace-Zeichen in HTML-Dokumenten behandelt werden
margin-top	<length>px	Breite des oberen Absatzrandes in Pixel
margin-bottom	<length>px	Breite des unteren Absatzrandes in Pixel
margin-left	<length>px	Breite des linken Absatzrandes in Pixel
margin-right	<length>px	Breite des rechten Absatzrandes in Pixel

Property	Werte	Beschreibung
vertical-align	baseline | sub | super | middle | top | bottom	Vertikale Ausrichtung für Text (in Tabellen werden nur die Werte middle, top, und bottom unterstützt)
border-color	<color>	Randfarbe für Texttabellen
border-style	none | dotted | dashed | dot-dash | dot-dot-dash | solid | double | groove | ridge | inset | outset	Randstil für Texttabellen
background	[<'background-color'> || <'background-image'>]	Kurzschreibweise für background Property
page-break-before	[auto | always]	Seitenumbruch vor einem Absatz/einer Tabelle
page-break-after	[auto | always]	Seitenumbruch nach einem Absatz/einer Tabelle
background-image	<uri>	Hintergrundbild für Elemente

Unterstützte CSS-Selektoren

Alle CSS 2.1 Selektorklassen werden unterstützt mit Ausnahme von sog. Pseudo-Selektorklassen wie :first-child, :visited und :hover.

6.3.3 Hilfebuch erzeugen

Das Hilfebuch ist eine XML-Datei, in der der Aufbau der Online-Hilfe festgelegt ist. In dieser Datei definieren Sie:

- HTML-Dokumente
- Inhalts- und Stichwortverzeichnis

Syntax für das Hilfebuch

Tag	Anzahl	Bedeutung	
HMI_SL_HELP	1	Root Element des XML-Dokuments	
I-BOOK I I I I I I I I I I	+	Bezeichnet ein Hilfebuch. Der Name ist frei wählbar unter der Randbedingung, dass kein vom System vordefinierter Name (wie z. B. sinumerik_alarm_plc_pmc) verwendet werden darf. Im Beispiel lautet der Name des Hilfebuchs: "hmi_myhelp" Attribute:	
		ref	Bezeichnet das HTML-Dokument, das als Einstiegsseite für das Hilfebuch angezeigt wird.
		titel	Titel des Hilfebuchs, das im Inhaltsverzeichnis angezeigt wird.
		helpdir	Verzeichnis, das die Online-Hilfe des Hilfebuchs beinhaltet.

Tag	Anzahl	Bedeutung	
I-ENTRY II II II II II II II	*	Kapitel der Online-Hilfe Attribute:	
		ref	Bezeichnet das HTML-Dokument, das als Einstiegsseite für das Kapitel angezeigt wird.
		titel	Titel des Kapitels, das im Inhaltsverzeichnis angezeigt wird.
II-INDEX_ENTRY II II II II II II II	*	Anzuzeigendes Stichwort Attribute:	
		ref	Bezeichnet das HTML-Dokument, das für diesen Stichworteintrag angesprungen wird.
		titel	Titel des Stichworts, das im Stichwortverzeichnis angezeigt wird.

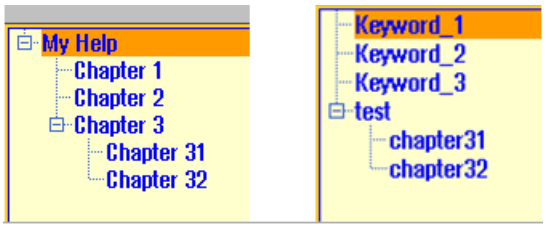
Für die Spalte "Anzahl" gilt:
* bedeutet 0 oder mehrere
+ bedeutet 1 oder mehrere

Beispiel für ein Hilfebuch

Im nachfolgenden Beispiel wird der Aufbau eines Hilfebuchs mit Namen "My Help" beschrieben. Weiterhin ist es die Basis für das Inhalts- und Stichwortverzeichnis.

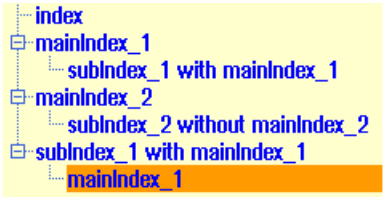
```
<?xml version="1.0" encoding="utf-8"?>
<HMI_SL_HELP language="en-US">
  <BOOK ref="index.html" title="My Help" helpdir="hmi_myhelp">
    <ENTRY ref="chapter_1.html" title="Chapter 1">
      <INDEX_ENTRY ref="chapter_1html#Keyword_1" title="Keyword_1"/>
      <INDEX_ENTRY ref="chapter_1.html#Keyword_2" title="Keyword_2"/>
    </ENTRY>
    <ENTRY ref="chapter_2.html" title="Chapter 2">
      <INDEX_ENTRY ref="chapter_2.html#Keyword_3" title="Keyword_3"/>
    </ENTRY>
    <ENTRY ref="chapter_3.html" title="Chapter 3">
      <ENTRY ref="chapter_31.html" title="Chapter 31">
        INDEX_ENTRY ref="chapter_31.html#test" title="test;chapter31"/>
      </ENTRY>
      <ENTRY ref="chapter_32.html" title="Chapter 32">
        INDEX_ENTRY ref="chapter_32.html#test" title="test;chapter32"/>
      </ENTRY>
    </ENTRY>
  </BOOK>
</HMI_SL_HELP>
```

Das Buch besteht aus drei Kapiteln, wobei das dritte Kapitel noch zwei Unterkapitel hat. Die verschiedenen Stichworte sind jeweils innerhalb des Kapitels definiert.



Sie haben folgende drei Möglichkeiten das Stichwortverzeichnis zu formatieren:

1. Einzeleintrag:
   `<INDEX_ENTRY ...title="index"/>`
2. Zwei zweistufige Einträge, wobei jeder Titel einen Haupt- und Untereintrag hat. Trennen Sie die Einträge mit einem Komma voneinander.
   `<INDEX_ENTRY ...title="mainIndex_1,subIndex_1 with mainIndex_1"/>`
3. Zweistufiger Eintrag, wobei der erste Titel der Haupteintrag und der zweite Titel der Untereintrag ist. Trennen Sie die Einträge mit einem Semikolon voneinander.
   `<INDEX_ENTRY ...title="mainIndex_2;subIndex_2 without mainIndex_1"/>`



6.3.4 Online-Hilfe in SINUMERIK Operate einbinden

Wenn Sie dass erzeugte Hilfebuch in das Online-Hilfesystem des SINUMERIK Operate einbinden möchten, benötigen Sie die Datei "slhlp.xml".

Formatbeschreibung der "slhlp.xml"

Tag	Anzahl	Bedeutung
CONFIGURATION	1	Root Element des XML–Dokuments. Kennzeichnet, dass es sich um ein Konfigurations-File handelt.
I-OnlineHelpFiles	1	Leitet die Sektion der Online-Hilfe-Bücher ein.
II-<help_book>	*	Leitet die Sektion eines Hilfe-Buchs ein.

Tag	Anzahl	Bedeutung	
III-EntriesFile III III III III III	1	Dateiname des Hilfe-Buchs mit den Inhalts- und Stichworteinträgen. Attribute:	
		value	Name der XML-Datei
		type	Datentyp des Werts (QString)
III-Technology III III III III III III III III III IIII	0, 1	Gibt die Technologie an, für die das Hilfebuch gilt. "All" gilt dabei für alle Technologien. Wenn das Hilfebuch für mehrere Technologien gilt, werden die Technologien durch ein Komma getrennt angegeben. Mögliche Werte: All, Universal, Milling, Turning, Grinding, Stroking, Punching Attribute:	
		value	Technologieangabe
		type	Datentyp des Werts (QString)
III -DisableSearch III III III III III	0, 1	Stichwortsuche für das Hilfe-Buch ausschalten. Attribute:	
		value	true, false
		type	type Datentyp des Werts (bool)
III-DisableFullTextSearch III III III III	0, 1	Volltextsuche für das Hilfe-Buch ausschalten. Attribute:	
		value	true, false
		type	type Datentyp des Werts (bool)
III-DisableIndex III III III III	0, 1	Stichwortverzeichnis für das Hilfe-Buch ausschalten. Attribute:	
		value	true, false
		type	type Datentyp des Werts (bool)
III-DisableContent III III III III	0, 1	Inhaltsverzeichnis für das Hilfe-Buch ausschalten. Attribute:	
		value	true, false
		type	type Datentyp des Werts (bool)
III-DefaultLanguage III III III III III	0, 1	Kürzel für die Sprache die angezeigt werden soll, wenn für das Hilfe-Buch die aktuelle Landessprache vorhanden ist. Attribute:	
		value	chs, deu, eng, esp, fra, ita, ...
		type	Datentyp des Werts (QString)

Für die Spalte "Anzahl" gilt:
* bedeutet 0 oder mehrere

Beispiel einer Datei "slhlp.xml"

Im nachfolgenden Beispiel wird das Hilfebuch "hmi_myhelp.xml" SINUMERIK Operate bekannt gegeben.

Das Stichwortverzeichnis ist für das Hilfebuch nicht aktiviert.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE CONFIGURATION>
<CONFIGURATION>
   <OnlineHelpFiles>
     <hmi_myHelp>
       <EntriesFile value="hmi_myhelp.xml" type="QString"/>
       <DisableIndex value="frue" type="bool"/>
     </hmi_myHelp>
   </OnlineHelpFiles>
</CONFIGURATION>
```

6.3.5 Hilfedateien ablegen

Hilfedateien im Zielsystem ablegen

1. Öffnen Sie das Verzeichnis /**oem**/sinumerik/hmi/hlp und legen Sie für die gewünschte Sprache einen neuen Ordner an. Verwenden Sie dafür die vorgegebene Sprachkennung. Die Ordnernamen müssen unbedingt kleingeschrieben werden.
   Wenn Sie z. B. eine Hilfe für die Sprachen Deutsch und Englisch einbinden, legen Sie die Ordner "deu" und "eng" an.
2. Legen Sie das Hilfebuch z. B. "hmi_myhelp.xml" jeweils in den Ordner "deu" und "eng".
3. Kopieren Sie die Hilfedateien in die Verzeichnisse, z. B. /**oem**/sinumerik/hmi/hlp/deu/hmi_myhelp für deutschsprachige und /**oem**/sinumerik/hmi/hlp/eng/hmi_myhelp für englischsprachige Hilfedateien.
4. Legen Sie die Konfigurationsdatei "slhlp.xml" in das Verzeichnis /**oem**/sinumerik/hmi/cfg.
5. Starten Sie den HMI neu.

Hinweis

Bei der Anzeige des Inhalts- und Stichwortverzeichnisses eines Hilfebuchs werden zur schnelleren Bearbeitung, unter dem Verzeichnis /siemens/sinumerik/sys_cache/hmi/hlp, die Hilfedateien im Binärformat abgelegt (slhlp_<Hilfe-Buch_*.hmi). Wenn Sie das Hilfebuch ändern, müssen Sie immer diese Dateien löschen.

6.3.6 Hilfedateien im PDF-Format

Sie haben die Möglichkeit, neben Hilfedateien im HTML-Format, Informationen auch im PDF-Format in der Bedien-Software einzubinden. Einzelne PDF-Hilfen können über Verknüpfungen aus dem Inhalts- bzw. Indexverzeichnis oder direkt aus HTML-Dateien geöffnet werden.

PDF-Hilfen ablegen

Kopieren Sie die PDF-Hilfen in eines der folgenden Verzeichnisse:

```
/oem/sinumerik/hmi/hlp/<lng>/<hmi_myhelp>
```

```
/user/sinumerik/hmi/hlp/<lng>/<hmi_myhelp>
```

PDF-Hilfen einbinden

Binden Sie in Dialogprojektierungen oder Projektierungen von Inhalts- und Indexverzeichnissen die Dateien mit der Endung "pdf" genauso ein, wie für die Endung "html":

```
<ENTRY ref="myFile.pdf" title="Help 1">
```

Verweisen Sie aus HTML-Dateien heraus per Link auf die PDF-Hilfe:

```
<a href="myFile.pdf">My Help File</a>
```

Hinweis

Es ist in der PDF-Hilfe nicht möglich, Sprungmarken kontextsensitiv anzuwählen oder Sprünge zu anderen HTML- oder PDF-Dateien auszuführen.

Die Suchfunktion ist nur innerhalb einer PDF-Datei möglich. Eine übergeordnete Suche über mehrere PDF-Hilfen wird nicht unterstützt.

Variablen

7

7.1 Variablen definieren

Variablenwert

Die wesentlichste Eigenschaft einer Variablen ist der Variablenwert.

Der Wert der Variablen kann zugewiesen werden durch:

- die Vorbelegung bei der Definition der Variablen
- die Zuordnung einer System- oder Anwendervariablen
- eine Methode

Programmierung

Syntax:	Bezeichner.**val** = Variablenwert Bezeichner = Variablenwert	
Beschreibung:	Variablenwert val (value)	
Parameter:	Bezeichner:	Name der Variablen
	Variablenwert:	Wert der Variablen
Beispiel:	`VAR3 = VAR4 + SIN(VAR5)` `VAR3.VAL = VAR4 + SIN(VAR5)`	

Variablenzustand

Mit der Eigenschaft Variablenzustand kann zur Laufzeit abgefragt werden, ob eine Variable einen gültigen Wert enthält. Diese Eigenschaft ist lesbar und schreibbar mit dem Wert FALSE = 0.

Programmierung

Syntax:	Bezeichner.**vld**	
Beschreibung:	Variablenzustand vld (validation)	
Parameter:	Bezeichner:	Name der Variablen
		Das Ergebnis der Abfrage kann sein:
	FALSE =	ungültiger Wert
	TRUE =	gültiger Wert

Beispiel:	`IF VAR1.VLD == FALSE`
	`  VAR1 = 84`
	`ENDIF`	

7.2 Anwendungsbeispiele

Hilfsvariablen

Hilfsvariablen sind interne Rechenvariablen. Die Rechenvariablen werden wie Variablen definiert, besitzen aber außer Variablenwert und Zustand keine Eigenschaften, d.h. die Hilfsvariablen sind im Dialog nicht sichtbar. Hilfsvariablen sind vom Typ VARIANT.

Programmierung

Syntax:	DEF [Bezeichner]	
Beschreibung:	interne Rechenvariablen vom Typ VARIANT	
Parameter:	Bezeichner:	Name der Hilfsvariablen
Beispiel:	`DEF OTTO ;Definition einer Hilfsvariablen`	

Syntax:	Bezeichner.val = Hilfsvariablenwert Bezeichner = Hilfsvariablenwert	
Beschreibung:	Der Wert einer Hilfsvariablen wird in einer Methode zugewiesen.	
Parameter:	Bezeichner:	Name der Hilfsvariablen
	Hilfsvariablenwert:	Inhalt der Hilfsvariablen

Beispiel:

```
LOAD
  OTTO = "Test"             ; Hilfsvariable Otto den Wert "Test" zuweisen
END_LOAD
LOAD
  OTTO = REG[9].VAL         ; Hilfsvariable Otto den Wert des Registers zuweisen
END_LOAD
```

Berechnung mit Variablen

Variablen werden nach jedem Verlassen eines Ein-/Ausgabefeldes (durch die ENTER- oder Toggle-Taste) berechnet. Die Berechnung wird in einer CHANGE-Methode projektiert und bei jeder Änderung des Wertes durchlaufen.

Ob eine Variable einen gültigen Wert hat, lässt sich über den Variablenzustand abfragen, z. B.:

```
IF VAR1.VLD == FALSE
   VAR1 = 84
ENDIF
```

Systemvariable indirekt adressieren

Eine Systemvariable kann auch indirekt, d.h. in Abhängigkeit von einer anderen Variablen, adressiert werden:

```
PRESS(HS1)
  ACHSE=ACHSE+1
  WEG.VAR="$AA_DTBW["<<ACHSE<<"]"       ;Achsadresse über Variable adressieren
END_PRESS
```

Softkey-Beschriftung ändern

Beispiel:

```
HS3.st = "Neuer Text"                   ;Softkey-Beschriftung ändern
```

7.3 Beispiel 1: Variablentyp, Texte, Hilfebild, Farben, Tooltips zuweisen

Beispiel 1a

Das nachfolgende Beispiel definiert eine Variable, bei der die Eigenschaften Variablentyp, Texte, Hilfebild und Farben gesetzt werden.

DEF Var1 = (R///,"Istwert",,"mm"//"Var1.png"////8,2)			
	Variablentyp:		REAL
	Texte:		
		Kurztext:	Istwert
		Einheitentext:	mm
	Hilfebild:		Var1.png
	Farben:		
		Vordergrundfarbe:	8 (braun)
		Hintergrundfarbe:	2 (orange)

Beispiel 1b

Das nachfolgende Beispiel definiert eine Variable, bei der die Eigenschaften Variablentyp, Vorbelegung, Texte, ToolTip, Eingabemodus und Position Kurztext gesetzt werden.

DEF Var2 = (I//5/"","Wert","","," Tooltip-Text"/wr2///20,250,50)			
	Variablentyp:		INTEGER
	Vorbelegung:		5
	Texte:		
		Kurztext:	Wert (mögliche Sprachtext-ID)
		Tooltip:	TooltipText
	Attribute:		
		Eingabemodus	lesen und schreiben
	Position Kurztext:		
		Abstand von links	20
		Abstand von oben	250
		Breite:	50

Siehe auch

Parameter von Variablen (Seite 91)

7.4 Beispiel 2: Variablentyp, Grenzwerte, Attribute, Position Kurztext zuweisen

Beispiel 2

Das nachfolgende Beispiel definiert eine Variable, bei der die Eigenschaften Variablentyp, Grenzwerte, Eingabemodus, Ausrichtung und Position gesetzt werden.

DEF Var2 = (I/0,10///wr1,al1/// , ,300)			
	Variablentyp:		INTEGER
	Grenzwerte oder Toggle-Feldeinträge:		MIN: 0 MAX: 10
	Attribute:		
		Eingabemodus	nur lesbar
		Textausrichtung Kurztext	rechtsbündig
	Position Kurztext:		
		Breite:	300

Siehe auch

Parameter von Variablen (Seite 91)

7.5 Beispiel 3: Variablentyp, Vorbelegung, System- oder Anwendervariable, Position Ein-/Ausgabefeld zuweisen

Beispiel 3

Das nachfolgende Beispiel definiert eine Variable, bei der die Eigenschaften Variablentyp, Vorbelegung, System- oder Anwendervariable und Position gesetzt werden.

DEF Var3 = (R//10////"$R[1]"//300,10,200//)			
	Variablentyp:		REAL
	Vorbelegung:		10
	System- oder Anwendervariable:		$R[1] (R-Parameter 1)
	Position Kurztext:		Standardposition relativ zu Ein-/Ausgabefeld
	Position Ein-/Ausgabe-Feld:		
		Abstand von links	300
		Abstand von oben	10
		Breite:	200

Siehe auch

Parameter von Variablen (Seite 91)

7.6 Beispiel 4: Toggle-Feld und Listen-Feld

Beispiel 4a

Verschiedene Einträge im Toggle-Feld:

```
DEF Var1 = (I/* 0,1,2,3)                ;Einfaches Toggle-Feld zum toggeln von numerischen Werten
DEF Var2 = (S/* "On", "Off")            ;Einfaches Toggle-Feld zum toggeln von Strings
DEF Var3 = (B/* 1="On", 0="Off")        ;Erweitertes Toggle-Feld zum toggeln von numerischen Werten,
                                         wobei jedem numerischen Wert ein Anzeigetext zugeordnet wird
DEF Var4 = (R/* ARR1)                   ;Toggle-Feld, welches seine zu toggelnden Werte aus einem Ar-
                                         ray bezieht
```

Beispiel 4b

Das Listenfeld entspricht der Projektierung eines Toggel-Feldes, jedoch muss zusätzlich der Anzeigetyp für Listenfeld (Variablenattribut DT = 4) gesetzt sein.

DEF VAR_LISTBOX_Text = (S/*$80000,$80001,$80002,$80003,$80004/$80001//DT4////200,,340,60)		
Variablentyp:		STRING
Grenzwerte/Toggle-Feld:		*$80000,$80001,$80002,$80003,$80004 (Liste der anzuzeigenden sprachabhängigen Texte)
Vorbelegung:		$80001
Attribute:		
	Anzeigetyp:	4 (Listenfeld)
Position Ein-/Ausgabe-Feld:		
	Abstand von Links:	200
	Breite:	340
	Höhe:	60
Farben:		
	Vordergrundfarbe:	6 (blau)
	Hintergrundfarbe:	10 (weiß)

7.7 Beispiel 5: Bild-Anzeige

Beispiel 5

Ein Bild anstatt eines Kurztextes anzeigen: Größe und Position des Bildes ist unter "Position Ein-/Ausgabefeld (Left, Top, Width, Height)" angegeben.

DEF VAR6 = (V///,"\\bild1.png" ////160,40,50,50)		
Variablentyp:		VARIANT
Texte:		
	Kurztext:	bild1.png
Position Ein-/Ausgabefeld:		
	Abstand von links:	160
	Abstand von oben:	40
	Breite:	50
	Höhe:	50

7.8 Beispiel 6: Fortschrittsbalken

Der Fortschrittsbalken ist ein spezieller Anzeigetyp des Ein-/Ausgabefeldes und ist nur zur Anzeige ohne Eingabe konzipiert.

Grundsätzlich gibt es zwei Typen von Fortschrittsbalken:

1. Fortschrittsbalken mit bis zu zwei Farbumschlägen für z.B. Temperatur- oder Auslastungsanzeige (siehe Beispiel 6a)
2. Fortschrittsbalken zur Anzeige eines Fortschrittes (ohne Farbumschlag) im Operate-Stil (siehe Beispiel 6b)

Beispiel 6a

Fortschrittsbalken mit zwei Farbumschlägen:

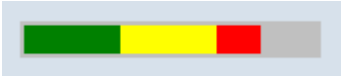


Bild 7-1 Fortschrittsbalken mit zwei Farbumschlägen

DEF PROGGY0 = (R/0,150,50,100///DT1,DO0//"$R[10]"//,,150/3,4,,,,,,9,7)		
Variablentyp:		REAL
Grenzwerte/Toggle-Feld:		
	MIN:	0
	MAX:	150
	Signalwert SVAL1 :	50
	Signalwert SVAL2:	100
Attribute:		
	Anzeigemodus DT:	1 (Fortschrittsbalken)
	Anzeigeoption DO:	0 (von links nach rechts (default))
System- oder Anwendervariable:		$R[10]
Position Ein-/Ausgabe-Feld:		
	Breite:	150
Farben:		
	Vordergrundfarbe:	3 (dunkelgrün)
	Hintergrundfarbe:	4 (hellgrau)
	Signalfarbe SC1:	9 (gelb)
	Signalfarbe SC2:	7 (rot)

Zur Verwendung eines Fortschrittsbalkens mit Farbumschlag muss der Anzeigemodus DT (DisplayType) auf 1 gesetzt werden.

Die Orientierung der Fortschrittsanzeige wird über das Attribut Anzeigeoption DO (DisplayOption) bestimmt:

0: von links nach rechts (default)

1: von rechts nach links

2: von unten nach oben

3: von oben nach unten

Für die Darstellung des Fortschrittsbalkens müssen ein MIN- und ein MAX-Wert angegeben werden (im Beispiel MIN: 0, MAX: 150).

Bereits mit dieser Einstellung wird, abhängig vom aktuellen Wert der Variable PROGGY0, ein Fortschrittsbalken mit der Vordergrundfarbe 3 (= dunkelgrün) und der Hintergrundfarbe 4 (= hellgrau) angezeigt.

Optional können ein oder zwei Signalwerte SVAL1 und SVAL2 (Parameter Grenzwert) definiert werden (im Beispiel SVAL1: 50 und SVAL2: 100). Bei diesen Signalwerten schlägt die Vordergrundfarbe der Fortschrittsanzeige um. Die entsprechenden Signalfarben werden über die Parameter SC1 und SC2 festgelegt (in obigem Beispiel SC1: 9 (= gelb) und SC2: 7 (=rot)).

Für die Angabe der Grenzwerte für die Fortschrittsanzeige gilt folgende Vorschrift:

MIN < SVAL1 < SVAL2 < MAX.

Beispiel 6b

Fortschrittsbalken ohne Farbumschlag:

Bild 7-2 Fortschrittsbalken

DEF PROGGY0 = (R/0,150///DT2,DO0//"$R[10]"//,,150/6,10)		
Variablentyp:		REAL
Grenzwerte/Toggle-Feld:		
	MIN:	0
	MAX:	150
Attribute:		
	Anzeigetyp DT:	2 (Fortschrittsbalken)
	Anzeigeoption DO:	0 (von links nach rechts (default))
System- oder Anwendervariable:		$R[10]
Position Ein-/Ausgabe-Feld:		
	Breite:	150
Farben:		
	Vordergrundfarbe:	6 (blau)
	Hintergrundfarbe:	10 (weiß)

Zur Verwendung eines Fortschrittsbalkens ohne Farbumschlag muss der Anzeigemodus DT (DisplayType) auf 2 gesetzt werden.

Die Orientierung der Fortschrittsanzeige wird über das Attribut Anzeigeoption DO (DisplayOption) bestimmt (siehe Beschreibung Beispiel 6a).

Für die Darstellung des Fortschrittsbalkens müssen ein MIN- und ein MAX-Wert angegeben werden (im Beispiel MIN: 0, MAX: 150).

Mit dieser Einstellung wird, abhängig vom aktuellen Wert der Variable PROGGY0, ein Fortschrittsbalken mit der Vordergrundfarbe 6 (= blau) und der Hintergrundfarbe 10 (= weiß) angezeigt.

7.9 Beispiel 7: Passwort-Eingabemodus (Sternchen)

Beispiel 7

Zur Realisierung eines Feldes mit verdeckter Eingabe, z. B. für die Eingabe eines Kennwortes, muss der Anzeigemodus DT auf 5 gesetzt werden. Anstatt der eingegebenen Zeichen werden Sternchen angezeigt.

Bild 7-3 Passwort-Eingabemodus (Sternchen)

DEF VAR_SET_PWD=(S//""//DT5)		
	Variablentyp:	STRING
	Vorbelegung:	Leerstring
	Attribute:	
	Anzeigemodus DT:	5 (Passwort-Eingabemodus)

7.10 Parameter von Variablen

Parameter - Übersicht

In der folgenden Übersicht werden die Parameter der Variablen kurz erklärt. Eine ausführliche Beschreibung finden Sie in den nachfolgenden Kapiteln.

Parameter	**Beschreibung**	
Variablentyp (Seite 98)	Der Variablentyp muss angegeben werden.	
	R[x]:	REAL (+ Ziffer für Nachkommastellen)
	I:	INTEGER
	S[x]:	STRING (+ Ziffer für String-Länge)
	C:	CHARACTER (Einzelzeichen)
	B:	BOOL
	V:	VARIANT

Parameter	Beschreibung	
Grenzwerte (Seite 86)	Grenzwert MIN, Grenzwert MAX Voreinstellung: leer Die Grenzwerte werden durch Komma getrennt. Grenzwerte können für die Typen I, C und R in den Formaten dezimal oder als Charakter in der Form "A", "F", angegeben werden.	
	Für die Darstellung eines Fortschrittsbalkens mit Farbumschlag (Variablen-Attribut DT = 1) können optional zwei Signalfarben SC1 und SC2 (Signal Color) projektiert werden, die bei Überschreitung der Signalwerte SVAL1 bzw. SVAL2 (Signal Value) als jeweilige Vordergrundfarbe des Balkens verwendet werden. Die Angabe der Signalwerte erfolgt als Integer-Werte. Siehe Anwendungsbeispiel zu Fortschrittsbalken (Seite 88).	
Vorbelegung (Seite 102)	Wird keine Vorbelegung projektiert und ist der Variablen keine System- oder Anwendervariablen zugeordnet, wird das erste Element des Toggle-Feldes zugewiesen. Ist kein Toggle-Feld definiert, erfolgt keine Vorbelegung, d.h. die Variable erhält den Zustand "nicht berechnet". Voreinstellung: keine Vorbelegung	
Toggle-Feld (Seite 100)	Liste mit vorgegebenen Einträgen im Ein-/Ausgabefeld: Die Liste wird durch * eingeleitet, die Einträge werden durch Komma getrennt. Den Einträgen kann ein Wert zugewiesen werden. Der Eintrag für den Grenzwert wird beim Toggle-Feld als Liste interpretiert. Wird nur ein * eingetragen, wird ein variables Toggle-Feld erzeugt. Voreinstellung: keine	
Texte (Seite 85)	Die Reihenfolge ist vorgegeben. Statt des Kurztextes kann auch ein Bild aufgeblendet werden. Voreinstellung: leer	
	Langtext:	Text in der Anzeigezeile
	Kurztext:	Name des Dialogelements
	Grafiktext:	Text bezieht sich auf Begriffe in Grafik
	Einheitentext:	Einheit des Dialogelements
	Tooltip	Dienen als Kurzinformation in einer Maskenprojektierung für Anzeige- und Toggle-Felder. Die Information wird über Klartext und Sprachtext-ID projektiert.

Parameter	Beschreibung	
Attribute (Seite 86)	Die Attribute beeinflussen folgende Eigenschaften: • Anzeigemodus • Anzeigeoption • Aktualisierungsrate • Toggle-Symbol • Tooltip • Eingabemodus • Zugriffsstufe • Textausrichtung Kurztext • Schriftgröße • Grenzwerte Die Attribute werden durch Komma getrennt, die Reihenfolge ist beliebig. Pro Komponente kann eine Festlegung erfolgen.	
	Anzeigemodus	dt0: standard (Ein-/Ausgabefeld oder Toggle-Feld) (default) dt1: Fortschrittsbalken (Progressbar) mit Farbumschlag dt2: Fortschrittsbalken (Progressbar) ohne Farbumschlag im Operate-Stil dt4: Listenfeld dt5: Passwort-Eingabemodus (Sternchen)
	Anzeigeoption	Speziell für z.B. die Anzeigemodi dt1 und dt2 (Fortschrittsbalken) ist es gegebenenfalls notwendig, in Kombination auch eine Anzeigeoption zu projektieren. do0: von links nach rechts (default) do1: von rechts nach links do2: von unten nach oben do3: von oben nach unten
	Aktualisierungs-geschwindigkeit	Mit dem Attribut UR (update rate) kann die Aktualisierung der Anzeige und somit die Bearbeitung des zugehörigen projektierten CHANGE-Blockes von Variablen oder Grid-Spalten gesteuert werden. Je nach Projektierung kann die CPU-Last drastisch reduziert und somit kürzere Reaktionszeiten der Benutzerschnittstelle erzielt werden. ur0: SlCap::standardUpdateRate() (derzeit = 200 ms, Defaultwert) ur1: 50 ms ur2: 100 ms ur3: 200 ms ur4: 500 ms ur5: 1000 ms ur6: 2000 ms ur7: 5000 ms ur8: 10000 ms
	Toggle-Symbol	tg0: Toggle-Symbol aus (default) tg1: Toggle-Symbol an Setzt man das Attribut TG auf 1, dann erscheint im Tooltip des Eingabefeldes zusätzlich das Toggle-Symbol. Beispiel:

Parameter	Beschreibung	
		`DEF OFFS = (R//123.456/,,,,"My ToolTip"/TG1)`

	Tooltip	Der Tooltip-Text kann auch zur Laufzeit durch die Variablen-Eigenschaft „TT" geändert werden.
		Beispiel:
		`PRESS(VS1)`
		`  MyVar.TT = "My new ToolTip"`
		`END_PRESS`
		oder
		`DEF MyVar=(R///,,,,"My ToolTip-text")`

	Eingabemodus	wr0: Ein-/Ausgabefeld unsichtbar, Kurztext sichtbar wr1: lesen (es ist kein Fokus für Eingabe möglich) wr2: lesen und schreiben (Zeile erscheint in weiß) wr3: wr1 mit Fokus wr4: alle Elemente der Variablen unsichtbar, kein Fokus möglich wr5: Der eingegebene Wert wird bei jeder Tastenbetätigung sofort abgespeichert (im Gegensatz zu wr2 - dort wird erst bei Verlassen des Feldes oder bei Betätigung von RETURN abgespeichert). Voreinstellung: wr2
	Zugriffsstufe	leer: immer schreibbar ac0...ac7: Schutzstufen Wenn die Zugriffsstufe nicht ausreichend ist, erscheint die Zeile grau, Standardeinstellung: ac7
	Textausrichtung Kurztext	al0: linksbündig al1: rechtsbündig al2: zentriert Voreinstellung: al0
	Schriftgröße	fs1: Standard-Schriftgröße (8 Pt) fs2: doppelte Schriftgröße Voreinstellung: fs1 Der Abstand zwischen den Zeilen ist festgelegt. Bei Standard-Schriftgröße passen 16 Zeilen in den Dialog. Grafik- und Einheitentext können nur in Standard-Schriftgröße projektiert werden.
	Grenzwerte	Damit kann überprüft werden, ob der Wert der Variablen innerhalb der angegebenen Grenzwerte MIN und MAX liegt. Voreinstellung: abhängig von angegebenen Grenzwerten li0: keine Überprüfung li1: Überprüfung gegen Min li2: Überprüfung gegen Max li3: Überprüfung gegen Min und Max
	Verhalten beim Öffnen	cb-Attribute, die bei einer Variablendefinition angegeben sind, haben für die Variable Vorrang vor der Pauschalangabe cb in der Dialogdefinition. Mehrere Attribute werden durch Komma getrenntnotiert. (siehe auch AUTOHOTSPOT)
	Aufruf der CHANGE-Methode	CB0: Die CHANGE-Methode wird mit dem Aufblenden der Maske angestoßen, wenn die Variable zudem Zeitpunkt einen gültigen Wert hat (z.B. durch Vorbelegung oder NC/PLC-Variable).

Parameter	Beschreibung	
		CB1: Die CHANGE-Methode wird mit dem Aufblenden der Maske nicht explizit angestoßen. Hat die Variable eine projektierte NC/PLC-Variable, dann wird natürlich trotzdem die CHANGE-Methode aufgerufen.
	Empty Input (EI)	Mit dem Variablen-Attribut "EI" (= Empty Input) kann festgelegt werden, wie das Eingabefeld reagieren soll, wenn ein Leerstring eingegeben wird: EI0: Standard, d.h. leere Eingaben werden im Eingabefeld akzeptiert. EI1: Leere Eingaben führen zum Fehlerzustand im Eingabefeld und der vorherige gültige Wert wird wieder gesetzt (Undo).
Hilfebild (Seite 85)	Hilfebild-Datei:	Name der png-Datei Voreinstellung: leer
	Der Name der Hilfebild-Datei steht in Doppelhochkomma. Das Bild wird automatisch aufgeblendet (anstelle der bisherigen Grafik), wenn der Cursor auf diese Variable kommt.	
System- oder Anwendervariable (Seite 87)	Der Variablen kann ein System- oder Anwenderdatum aus der NC/PLC zugeordnet werden. Die System- oder Anwendervariable steht in Doppelhochkomma. Literatur: Listenhandbuch Systemvariablen, /PGAsl/	
Position Kurztext (Seite 103)	Position Kurztext (Abstand von links, Abstand von oben, Breite) Die Positionen werden in Pixel angegeben und beziehen sich auf die linke obere Ecke des Dialoghauptteils. Die Angaben werden jeweils durch Komma getrennt.	
Position Ein-/Ausgabefeld (Seite 103)	Position Ein-/Ausgabefeld (Abstand von links, Abstand von oben, Breite, Höhe) Die Positionen werden in Pixel angegeben und beziehen sich auf die linke obere Ecke des Dialoghauptteils. Die Angaben werden jeweils durch Komma getrennt. Wird diese Position geändert, ändern sich auch die Positionen des Kurztextes, Grafiktextes und Einheitentextes.	

Parameter	Beschreibung
Farben (Seite 85)	Vordergrund- und Hintergrundfarbe für Ein-/Ausgabe-Felder, Kurztexte, Grafiktexte Einheitentexte und Signalfarben für Fortschrittsbalken: Die Farben werden durch Komma getrennt. Zur Angabe der Farbe siehe Kapitel AUTOHOTSPOT. Voreinstellung für Ein/Ausgabefeld: Vordergrundfarbe: schwarz, Hintergrundfarbe: weiß Die Standardfarben des Ein-/Ausgabefeldes sind vom Schreibmodus "wr" abhängig. Voreinstellung für Kurztext, Grafiktext, Einheitentext: Vordergrundfarbe: schwarz, Hintergrundfarbe: transparent.
	Für die Darstellung eines Fortschrittsbalkens mit Farbumschlag (Variablen-Attribut DT = 1) können optional zwei Signalfarben SC1 und SC2 (Signal Color) projektiert werden, die bei Überschreitung der Signalwerte SVAL1 bzw. SVAL2 (Signal Value) als jeweilige Vordergrundfarbe des Balkens verwendet werden. Siehe Anwendungsbeispiel zu Fortschrittsbalken (Seite 88).
	In der Variablendefinition werden die Farben in folgender Reihenfolge erwartet: 1. Vordergrundfarbe des Ein-/Ausgabe-Feldes: FC 2. Hintergrundfarbe des Ein-/Ausgabe-Feldes: BC 3. Vordergrundfarbe des Kurztextes: FC_ST 4. Hintergrundfarbe des Kurztextes: BC_ST 5. Vordergrundfarbe des Grafiktextes: FC_GT 6. Hintergrundfarbe des Grafiktextes: BC_GT 7. Vordergrundfarbe des Einheitentextes: FC_UT 8. Hintergrundfarbe des Einheitentextes: BC_UT 9. Signalfarbe 1 10. Signalfarbe 2
Online-Hilfe-Datei	Der Name der Online-Hilfe-Datei steht in Doppelhochkomma.

Ein-/Ausgabefeld mit integrierter Einheiten-Selektion	Bei Ein-/Ausgabefeldern mit integrierten Einheiten-Selektion kann zwischen unterschiedlichen Einheiten gewechselt werden. Wird mit dem Cursor auf das Eingabefeld navigiert, dann wird die integrierte Einheiten-Selektion hervorgehoben (Fokus muss nicht explizit darauf gesetzt werden). Des Weiteren wird ein ToolTip mit einem Toggel-Symbol aufgeblendet, der einen entsprechenden Hinweis auf diese Funktionalität gibt.
	Beispiele:
	`DEF VarEdit=(R////////200,,100///"VarTgl")`
	`VarTgl=(S/*0="mm",1="inch"/0//WR2////302,,40)`
	oder
	`DEF VarEdit_2={TYP="R", VAL=1.234, X=200, W=100, LINK_TGL="vartgl_2"},`
	`VARTgl_2={TYP="S", TGL="* 0=""mm"", 1=""inch""",WR=2, X=302, W=40}`

Variable: Eigenschaften ändern

Den Variablen wird in der Notation Bezeichner.Eigenschaft = Wert beim Ändern ein neuer Wert zugewiesen. Der rechts vom Gleichheitszeichen stehende Ausdruck wird ausgewertet und der Variablen oder der Eigenschaft der Variablen zugewiesen.

Bezeichner.**ac** = Zugriffsstufe	(ac: access level)
Bezeichner.**al** = Textausrichtung	(al: alignment)
Bezeichner.**bc** = Hintergrundfarbe des Ein-/Ausgabefeldes	(bc: back color)
Bezeichner.**bc_gt** = Hintergrundfarbe des Grafiktextes	(bc: back color) (gt: graphic text)
Bezeichner.**bc_st** = Hintergrundfarbe des Kurztextes	(bc: back color) (st: short text)
Bezeichner.**bc_ut** = Hintergrundfarbe des Einheitentextes	(bc: back color) (ut: unit text)
Bezeichner.**do** = Anzeigeoption	(do: display option)
Bezeichner.**dt** = Anzeigemodus	(dt: display type)
Bezeichner.**fc** = Vordergrundfarbe des Ein-/Ausgabefeldes	(fc: front color)
Bezeichner.**fc_gt** = Vordergrundfarbe des Grafiktextes	(fc: front color) (gt: graphic text)
Bezeichner.**fc_st** = Vordergrundfarbe des Kurztextes	(fc: front color) (st: short text)
Bezeichner.**fc_ut** = Vordergrundfarbe des Einheitentextes	(fc: front color) (ut: unit text)
Bezeichner.**fs** = Schriftgröße	(fs: font size)
Bezeichner.**gt** = Grafiktext	(gt: graphic text)
Bezeichner.**hlp** = Hilfebild	(hlp: help)
Bezeichner.**li** = Grenzwert	(li: limit)
Bezeichner.**lt** = Langtext	(lt: long text)
Bezeichner.**max** = Grenzwerte MAX	(max: maximum)
Bezeichner.**min** = Grenzwerte MIN	(min: minimum)
Bezeichner.**sc** = Signalfarbe	(sc: signal color)
Bezeichner.**st** = Kurztext	(st: short text)
Bezeichner.**tg** = Toggle-Symbol	(tg: toggle)
Bezeichner.**tt** = Tooltip	(tt: tooltip
Bezeichner.**typ** = Variablentyp	(typ: type)
Bezeichner.**ur** = Aktualisierungsgeschwindigkeit	(ur: update rate)
Bezeichner.**ut** = Einheitentext	(ut: unit text)
Bezeichner.**val** = Variablenwert	(val: value)
Bezeichner.**var** = System- oder Anwendervariable	(var: variable)
Bezeichner.**vld** = Variablen-Zustand	(vld: validation)
Bezeichner.**wr** = Eingabemodus	(wr: write)

Siehe auch

Erweiterte Projektierungssyntax (Seite 46)

7.11 Einzelheiten zum Variablentyp

Variablentyp INTEGER

Folgende Erweiterungen zur Bestimmung der Darstellung im Ein-/Ausgabefeld und der Speicherbenutzung sind für Typ "INTEGER" möglich:

2. Zeichen im Erweiterungsdatentyp

Darstellungsformat	
B	binär
D	dezimal mit Vorzeichen
H	hexadezimal
Keine Angabe	dezimal mit Vorzeichen

3. und/oder 4. Zeichen im Erweiterungsdatentyp

Speicherbenutzung	
B	Byte
W	Word
D	Double Word
BU	Byte ohne Vorzeichen
WU	Word ohne Vorzeichen
DU	Double Word ohne Vorzeichen

Reihenfolge der Zeichen beim Datentyp INTEGER

1. "I" Grundsätzliche Kennzeichnung als INTEGER
2. Darstellungsformat
3. Speicherbenutzung
4. "U" ohne Vorzeichen

Gültige INTEGER Typfestlegungen:	
IB	Integervariable 32 Bit in binär Darstellung
IBD	Integervariable 32 Bit in binär Darstellung
IBW	Integervariable 16 Bit in binär Darstellung
IBB	Integervariable 8 Bit in binär Darstellung
I	Integervariable 32 Bit in dezimal Darstellung mit Vorzeichen
IDD	Integervariable 32 Bit in dezimal Darstellung mit Vorzeichen
IDW	Integervariable 16 Bit in dezimal Darstellung mit Vorzeichen
IDB	Integervariable 8 Bit in dezimal Darstellung mit Vorzeichen
IDDU	Integervariable 32 Bit in dezimal Darstellung ohne Vorzeichen
IDWU	Integervariable 16 Bit in dezimal Darstellung ohne Vorzeichen
IDBU	Integervariable 8 Bit in dezimal Darstellung ohne Vorzeichen

Gültige INTEGER Typfestlegungen:	
IH	Integervariable 32 Bit in hexadezimal Darstellung
IHDU	Integervariable 32 Bit in hexadezimal Darstellung
IHWU	Integervariable 16 Bit in hexadezimal Darstellung
IHBU	Integervariable 8 Bit in hexadezimal Darstellung

Variablentyp VARIANT

Der Variablentyp VARIANT ist durch den Datentyp der letzten Wertzuweisung bestimmt. Beginnt der zugewiesene oder eingegebene Wert mit '-'. '+', '.' oder einer Zahl ('0'-'9'), dann wird der Wert als numerisch interpretiert. In allen anderen Fällen als String.

Er kann mit der Funktion ISNUM oder ISSTR abgefragt werden. Der Typ VARIANT ist in erster Linie dazu geeignet, wahlweise Variablennamen oder numerische Werte in den NC-Code zu schreiben.

Programmierung

Der Datentyp der Variablen kann überprüft werden:

Syntax:	**ISNUM** (VAR)	
Parameter:	VAR	Name der Variablen, deren Datentyp überprüft werden soll.
		Das Ergebnis der Abfrage kann sein:
	FALSE =	keine numerische Variable (Datentyp = STRING)
	TRUE =	numerische Variable (Datentyp = REAL)

Syntax:	**ISSTR** (VAR)	
Parameter:	VAR	Name der Variablen, deren Datentyp überprüft werden soll.
		Das Ergebnis der Abfrage kann sein:
	FALSE =	numerische Variable (Datentyp = REAL)
	TRUE =	keine numerische Variable (Datentyp = STRING)
Beispiel:	`IF ISNUM(VAR1) == TRUE` `IF ISSTR(REG[4]+2) == TRUE`	

Der Anzeigemodus der Variablen kann geändert werden:

- Beim Typ INTEGER kann die Darstellungsart geändert werden.

B	binär
D	dezimal mit Vorzeichen
H	hexadezimal
ohne Vorzeichen	
Zusätzlich jeweils U für unsigned	

- Beim Typ REAL kann nur die Anzahl der Nachkommastellen geändert werden.
  Eine Änderung des Typs ist nicht erlaubt und führt zu einer Fehlermeldung in der Datei "easyscreen_log.txt".
  Beispiel:
  ```
  Var1.typ = "IBW"
  Var2.typ = "R3"
  ```

Zahlenformate

Zahlen können entweder in Binär-, Dezimal-, Hexadezimal- oder Exponentialschreibweise dargestellt werden:

binär	B01110110
dezimal	123.45
hexadezimal	HF1A9
exponential	-1.23EX-3
Beispiele:	
	`VAR1 = HF1A9`
	`REG[0]= B01110110`
	`DEF VAR7 = (R//-1.23EX-3)`

Hinweis

Bei der Code-Generierung durch die Funktion "GC" werden nur Zahlenwerte in Dezimal- oder Exponentialdarstellung und **nicht** in Binär- und Hexadezimaldarstellung berücksichtigt.

Siehe auch

Parameter von Variablen (Seite 91)

7.12 Einzelheiten zum Toggle-Feld

Beschreibung

Mit der Toggle-Feld-Erweiterung können Texte (Einträge im Toggle-Feld) abhängig von NC-/PLC-Variablen angezeigt werden. Eine Variable, die eine Toggle-Feld-Erweiterung benutzt, ist nur lesbar. Bei Drücken der INSERT-Taste wird die Liste des Toggle-Feldes aufgeklappt.

Programmierung

Syntax:	**DEF VAR1=(IB/+ $85000/15////"DB90.DBB5")** oder **DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run")**	
Beschreibung:	Beim Anzeigen des Dialogs wird der Inhalt der Textnummer $85015 im Ein-Ausgabefeld ausgegeben. In der Systemvariablen DB90.DBB5 wird die Vorbelegung 15 eingetragen. Ändert sich der Wert in der Systemvariablen DB90.DBB5, so wird bei jeder Änderung die angezeigte Textnummer neu gebildet $(85000 + <DB90.DBB5>).	
Parameter:	Variablentyp	Typ der in System- oder Anwendervariable spezifizierten Variablen
	Textnummer	Nummer (Basis) des sprachabhängigen Textes, der als Basisnummer gilt.
	System- oder Anwendervariable	System- oder Anwendervariable (Offset), über die die endgültige Textnummer (Basis + Offset) gebildet wird.

Bilder abhängig von Toggle-Feld

Das Toggle-Feld wird mit wechselnder Grafik überblendet: Hat das Merkerbyte den Wert 1, wird "bild1.png" angezeigt, hat das Merkerbyte den Wert 2, wird "bild2.png" angezeigt.

```
DEF VAR1=(IDB/*1="\\bild1.png", 2="\\bild2.png"//,$85000/wr1//"MB[130]"//160,40,50,50)
```

Lage und Größe des Bildes werden unter "Position Ein-Ausgabefeld(Left, Top, Width, Height)" angegeben.

Virtuelle Toggle-Taste

Bei Toggle-Feldern ohne projektierte Liste, z.B. `DEF NoTglList=(R/*)` gibt es keine Liste. Das nächste Element wird erst mit dem Betätigen der Toggle-Taste bzw. dem Durchlaufen der zugehörigen CHANGE()-Methode der entsprechenden Variable erzeugt.

Für diesen Fall wird für die Bedienung mit einem Touch-Panel eine kleine virtuelle Tastatur rechts neben dem variablen Toggle-Feld eingeblendet, die nur aus der Toggle-Taste besteht.

Die Anzeige der virtuellen Toggle-Taste kann auch unabhängig von einem Touch-Panel über folgenden Eintrag in der Konfigurationsdatei "slguiconfig.ini" erzwungen werden.

```
[VirtualKeyboard]
; Forces easyscreen virtual toggle keyboard function
ForceEasyscreenVirtualToggleKey = true
```

Siehe auch

Parameter von Variablen (Seite 91)

7.13 Einzelheiten zur Vorbelegung

Übersicht

Je nachdem, ob dem Feld einer Variablen (Ein-Ausgabefeld oder Toggle-Feld) eine Vorbelegung, eine System- oder Anwendervariable oder beides zugeordnet ist, ergeben sich verschiedene Zustände der Variablen (nicht berechnet: Toggeln ist erst möglich, wenn der Variablen ein gültiger Wert zugeordnet ist).

Wirkung der Vorbelegungen

Bedingung			**Reaktion des Feldtyps**
Feldtyp	**Vorbelegung**	**System- oder Anwendervariable**	
E-/A-Feld	Ja	Ja	Schreiben der Vorbelegung in System- oder Anwendervariable
	Nein	Ja	System- oder Anwendervariable als Vorbelegung verwenden
	Fehler	Ja	nicht berechnet, System- oder Anwendervariable wird nicht beschrieben/verwendet
	Ja	Nein	Vorbelegung
	Nein	Nein	nicht berechnet
	Fehler	Nein	nicht berechnet
	Ja	Fehler	nicht berechnet
	Nein	Fehler	nicht berechnet
	Fehler	Fehler	nicht berechnet
Toggle	Ja	Ja	Schreiben der Vorbelegung in System- oder Anwendervariable
	Nein	Ja	System- oder Anwendervariable als Vorbelegung verwenden
	Fehler	Ja	nicht berechnet, System- oder Anwendervariable wird nicht beschrieben/verwendet
	Ja	Nein	Vorbelegung
	Nein	Nein	Vorbelegung = erstes Element des Toggle-Feldes
	Fehler	Nein	nicht berechnet
	Ja	Fehler	nicht berechnet
	Nein	Fehler	nicht berechnet
	Fehler	Fehler	nicht berechnet

Siehe auch

Parameter von Variablen (Seite 91)

7.14 Einzelheiten zu Position Kurztext, Position Ein-Ausgabefeld

Übersicht

Kurz- und Grafiktext sowie Ein-Ausgabefeld und Einheitentext bilden jeweils eine Einheit, d.h. Positionsangaben für Kurztext wirken sich auch auf Grafiktext und Angaben für das Ein-Ausgabefeld und auf Einheitentext aus.

Programmierung

Die projektierte Positionsangabe überschreibt den Standardwert, d.h. es kann auch nur ein einzelner Wert geändert werden. Werden für nachfolgende Dialogelemente keine Positionsangaben projektiert, werden die Angaben des letzten Dialogelements übernommen.

Werden für kein Dialogelement Positionen angegeben, wird die Voreinstellung verwendet. Die Spaltenbreite für den Kurztext und das Ein-Ausgabefeld wird für jede Zeile im Standardfall aus Spaltenanzahl und maximaler Zeilenbreite bestimmt, d.h. Spaltenbreite = maximale Zeilenbreite/Spaltenanzahl.

Die Breite des Grafik- und Einheitentexts ist fest und für die Anforderungen der Programmierunterstützung optimiert. Wenn Grafik- oder Einheitentext projektiert wurde, wird die Breite des Kurztext oder Ein-Ausgabefeldes entsprechend verkürzt.

Die Reihenfolge von Kurztext und Ein-Ausgabefeld kann durch die Positionsangabe getauscht werden.

Abstand zwischen Ein-/Ausgabe-Feld und Einheiten-Feld und Breite des Einheitenfeldes

Den Abstand zwischen einem Ein-/Ausgabe-Feld und einem Einheiten-Feld sowie die Breite des Einheiten-Feldes können Sie projektieren.

In der Definitions-Zeile geben Sie dazu im Abschnitt für die Ein-/Ausgabe-Position per Komma getrennt den Abstand vom Ein-/Ausgabe-Feld zum Einheiten-Feld (z. B. 7 Pixel) und/oder die Breite des Einheiten-Feldes (z. B. 60 Pixel) an:

```
DEF VarDT=(R3//0.000/,"DT",,"s"////0,,24/39,,71,,7,60)
```

Oder:

```
DEF VarDT={TYP="R3", VAL="0.000", ST="DT", UT="s", TXT_X=0,
TXT_W=24, X=39, W=71, UT_DX=7, UT_W=60}
```

In diesem Fall - der Abstand zwischen Ein-/Ausgabe-Feld / Einheiten-Feld und/oder die Breite des Einheiten-Feldes ist projektiert - sind folgende Punkte zu beachten:

- Die projektierte Breite des Ein-/Ausgabe-Feldes enthält **nicht** die feste Breite des Einheiten-Feldes (fest 50 Pixel). D. h. Sie projektieren direkt die Breite des Ein-/Ausgabe-Feldes.
- Ist keine Breite für das Einheiten-Feld projektiert, gilt die Breite von 50 Pixeln default.
- Ist kein Abstand zwischen Ein-/Ausgabe-Feld / Einheiten-Feld projektiert, gilt der Abstand von 0 Pixeln default.

Besonderheit beim assoziierten Toggle-Feld

Voraussetzungen:

- Zu einer Variablen ist ein assoziiertes Toggle-Feld projektiert,
- für die Variable ist ein Abstand zwischen Ein-/Ausgabe-Feld und Einheiten-Feld und/oder eine Breite für das Einheiten-Feld projektiert und
- die Variable hat keinen Einheitentext.

In diesem Fall wird das assoziierte Toggle-Feld an die projektierte Position des Einheiten-Feldes der Variable positioniert.

Eine eventuell für den Ein-/Ausgabe-Anteil des assoziierten Toggle-Feldes projektierte Position wird dabei ignoriert.

Beispiel

In nachfolgendem Beispiel wird das assoziierte Toggle-Feld **F_Unit** automatisch mit einem Abstand von **7 Pixeln** zum Ein-/Ausgabe-Feld der Variable **VarF** positioniert und mit eine Breite von **59 Pixeln** gesetzt.

```
DEF VarF=(R//0.0/,"F",,,////0,,24/39,,85,,7,59///"F_Unit"),  F_Unit
= (I/*3="mm/min", 1="mm/U"/3// ////181,,155)
```

Siehe auch

Parameter von Variablen (Seite 91)

7.15 Verwendung von Strings

Stringketten

Bei der Projektierung können auch Strings verwendet werden, um die Anzeige von Text dynamisch zu gestalten oder unterschiedliche Texte für die Code-Generierung zusammenzusetzen.

Regeln

Bei der Verwendung von Stringvariablen sind folgende Regeln zu beachten:

- Verknüpfungen werden von links nach rechts abgearbeitet.
- Verschachtelte Ausdrücke werden von innen nach außen aufgelöst.
- Groß-/Kleinschreibung wird ignoriert.
- Stringvariablen werden generell linksbündig angezeigt.

Strings können durch eine einfache Zuweisung eines Leerstrings gelöscht werden.

Strings können rechts vom Gleichheitszeichen durch den Operator "<<" angehängt werden. Doppelhochkommas (") im String werden durch zwei aufeinander folgende Doppelhochkomma gekennzeichnet. Strings können in IF-Anweisungen auf Gleichheit geprüft werden.

Beispiel

Vorbelegung für die folgenden Beispiele:

```
VAR1.VAL = "Dies ist ein"
VAR8.VAL = 4
VAR14.VAL = 15
VAR2.VAL = "Fehler"
$85001 = "Dies ist ein"
$85002 = "Alarmtext"
```

Bearbeiten von Strings:

- Zusammensetzten von Strings:
  ```
  VAR12.VAL = VAR1 << " Fehler."   ;Ergebnis: "Dies ist ein Fehler"
  ```
- Löschen einer Variablen:
  ```
  VAR10.VAL = ""   ;Ergebnis: Leerstring
  ```
- Setzten einer Variablen mit einer Textvariablen:
  ```
  VAR11.VAL = VAR1.VAL   ;Ergebnis: "Dies ist ein"
  ```
- Datentypanpassung:
  ```
  VAR13.VAL ="Dies ist der " << (VAR14 - VAR8) << ". Fehler"
              ;Ergebnis: "Dies ist der 11. Fehler"
  ```
- Behandlung von numerischen Werten:
  ```
  VAR13.VAL = "Fehler " << VAR14.VAL << ": " << $85001 << $85002
           ;Ergebnis: "Fehler 15: Dies ist ein Alarmtext"
  IF VAR15 == "Fehler"   ;Strings in IF - Anweisung
    VAR16 = 18.1234
           ;Ergebnis: VAR16 gleich 18.1234,
           ;wenn VAR15 gleich "Fehler" ist.
  ENDIF
  ```
- Doppelhochkomma innerhalb eines Strings:
  ```
  VAR2="Hallo dies ist ein " Test""
     ;Ergebnis: Hallo dies ist ein " Test"
  ```
- Strings von System- oder Anwendervariablen abhängig von Variableninhalten:
  ```
  VAR2.Var = "$R[" << VAR8 << "]"   ;Ergebnis: $R[4]
  ```

Siehe auch

STRING-Funktionen (Seite 178)

7.16 Variable CURPOS

Beschreibung

Mit der Variablen CURPOS ist es möglich, die Position des Cursors im aktiven Eingabefeld des aktuellen Dialogs abzurufen oder zu manipulieren. Die Variable zeigt an, wieviele Zeichen vor dem Cursor stehen. Ist der Cursor am Anfang des Eingabefeldes positioniert nimmt CURPOS den Wert 0 an. Ändert man den Wert von CURPOS, so wird der Cursor im Eingabefeld an die entsprechende Stelle gestellt.

Um auf Änderungen des Variablenwertes reagieren zu können, ist es möglich, sie mit Hilfe einer CHANGE-Methode auf Änderungen zu überwachen. Ändert sich der Wert von CURPOS, wird dann die CHANGE-Methode angesprungen und die enthaltenen Anweisungen ausgeführt.

7.17 Variable CURVER

Beschreibung

Die Eigenschaft CURVER (Current Version) erlaubt die Anpassung der Programmierung zur Behandlung der unterschiedlichen Versionen. Die Variable CURVER ist nur lesbar.

Hinweis

Bei der Code-Generierung wird automatisch mit der neuesten Version generiert, auch wenn vorher mit einer älteren Version rückübersetzt wurde. Das Kommando "GC" generiert immer die neueste Version. Im generierten Code wird im Nutzkommentar bei Versionen > 0 eine zusätzliche Kennung der generierten Version eingefügt.

Regeln

Es ist immer der neueste Dialog mit allen seinen Variablen zu sehen.

- Bisherige Variablen dürfen nicht verändert werden.
- Neue Variablen werden in beliebiger Reihenfolge in die bisherige (Zyklen-) Programmierung eingefügt.
- Ein Entfernen von Variablen aus einem Dialog von einer Version zur nächsten ist nicht erlaubt.
- Der Dialog muss alle Variablen aller Versionen enthalten.

Beispiel

```
(IF CURVER==1 ...)                ; CURVER wird bei Rückübersetzung automatisch mit
                                    der Version des rückübersetzten Codes gesetzt.
```

7.18 Variable ENTRY

Beschreibung

Mit der Variablen ENTRY kann geprüft werden, wie der Dialog aufgerufen wurde.

Programmierung

Syntax:	**ENTRY**	
Beschreibung:	Die Variable ENTRY ist nur lesbar.	
Rückgabewert:		Das Ergebnis der Abfrage kann sein:
	0 =	Keine Programmierunterstützung
	1 =	Start einer Maske über Softkey ; es wurde noch kein Code generiert (Vorbelegung, wie projektiert)
	2 =	Start einer Maske über Softkey ; es wurde Code generiert (Vorbelegung aus dem zuletzt von dieser Maske generierten Code)
	3 =	Rückübersetzen mit Nutzkommentar (#-Zeilen)
	4 =	Code_typ = 0 : NC-Code mit Nutzkommentar (#-Zeilen)
	5 =	Code_typ = 1 : NC-Code ohne Nutzkommentar (#-Zeilen)

Beispiel

```
IF ENTRY == 0
  DLGL("Der Dialog wurde nicht unter Programmierung aufgerufen")
ELSE
  DLGL("Der Dialog wurde unter Programmierung aufgerufen")
ENDIF
```

7.19 Variable ERR

Beschreibung

Mit der Variable ERR kann geprüft werden, ob die jeweilige vorhergehende Zeile fehlerfrei ausgeführt wurden.

Programmierung

Syntax:	**ERR**
Beschreibung:	Die Variable ERR ist nur lesbar.

Rückgabewert:		Das Ergebnis der Abfrage kann sein:
	FALSE =	vorherige Zeile wurde fehlerfrei ausgeführt
	TRUE =	vorherige Zeile wurde nicht fehlerfrei ausgeführt

Beispiel

```
VAR4 = Gewinde[VAR1,"KDM",3]          ; Wert aus Array ausgeben
IF ERR == TRUE                        ; Abfrage ob Wert im Array gefunden wurde
  VAR5 = "Fehler beim Arrayzugriff"
                                      ; Wurde der Wert im Array nicht gefunden,
                                      wird der Variablen der Wert "Fehler beim Ar-
                                      rayzugriff" zugewiesen.
ELSE
  VAR5 = "Alles OK"                   ; Wurde der Wert im Array gefunden, wird der
                                      Variablen der Wert "Alles OK" zugewiesen.
ENDIF
```

7.20 Variable FILE_ERR

Beschreibung

Mit der Variable FILE_ERR kann geprüft werden, ob der vorangegangene GC- oder CP-Befehl fehlerfrei ausgeführt wurde.

Programmierung

Syntax:	**FILE_ERR**	
Beschreibung:	Die Variable FILE_ERR ist nur lesbar.	
Rückgabewert:		Mögliche Ergebnisse sind:
	0 =	Operation in Ordnung
	1 =	Laufwerk/Pfad nicht vorhanden
	2 =	Pfad-/Datei-Zugriffsfehler
	3 =	Laufwerk nicht bereit
	4 =	falscher Dateiname
	5 =	Datei ist schon geöffnet
	6 =	Zugriff verweigert
	7 =	Zielpfad nicht vorhanden oder nicht zulässig
	8 =	Kopierquelle entspricht Ziel
	10 =	interner Fehler: Im Fall von FILE_ERR = 10 handelt es sich um einen Fehler, der nicht in die anderen Kategorien eingeordnet werden kann.

Beispiel

```
CP("D:\source.mpf","E:\target.mpf")
                                                ; Kopieren von source.mpf nach
                                                E:\target.mpf
IF FILE_ERR > 0                                 ; Abfrage, ob Fehler aufgetreten ist
   IF FILE_ERR == 1                             ; Abfragen bestimmter Fehlernummern
                                                und Ausgabe des dazugehörigen Fehler-
                                                textes
      VAR5 = "Laufwerk/Pfad nicht vorhanden"
   ELSE
      IF FILE_ERR == 2
         VAR5 = "Pfad-/Datei-Zugriffsfehler"
      ELSE
         IF FILE_ERR == 3
            VAR5 = "falscher Dateiname"
         ENDIF
      ENDIF
   ENDIF
ELSE
   VAR5 = "Alles OK"                            ; wenn kein Fehler in CP (oder GC)
                                                aufgetreten ist, Ausgabe von "Alles
                                                OK"
ENDIF
```

7.21 Variable FOC

Beschreibung

Mit der Variablen FOC wird der Eingabefokus (aktuelles aktives Ein-/Ausgabefeld) in einem Dialog gesteuert. Die Reaktion von Cursor links, rechts, aufwärts, abwärts sowie PGUP, PGDN sind fest vordefiniert.

Hinweis

FOC darf nicht mit einem Navigationsereignis ausgelöst werden. Die Cursor-Position darf nur in Softkey-PRESS Methoden, CHANGE Methoden, ... verändert werden.

Variablen mit dem Eingabemodus wr = 0 und wr = 4 und Hilfsvariablen können nicht fokussiert werden.

Programmierung

Syntax:	**FOC**	
Beschreibung:	Die Variable kann gelesen und geschrieben werden.	
Rückgabewert:	Lesen	Als Ergebnis wird der Name der fokussierten Variable geliefert.
	Schreiben	Es kann entweder ein String oder ein numerischer Wert zugewiesen werden. Ein String wird als Variablenname und ein numerischer Wert als Variablenindex interpretiert.

Beispiel

```
IF FOC == "Var1"              ; Fokus lesen
  REG[1] = Var1
ELSE
  REG[1] = Var2
ENDIF

FOC = "Var1"                  ; Der Eingabefokus wird Variable 1 zugewiesen.
FOC = 3                       ; Der Eingabefokus wird dem 3. Dialog-Element mit
                              WR ≥ 2 zugewiesen.
```

Siehe auch

FOCUS (Seite 123)

7.22 Variable S_ALEVEL

Beschreibung

Die aktuelle Zugriffsstufe kann mit der Masken-Eigenschaft S_ALEVEL in der Projektierung abgefragt werden.

Programmierung

Syntax:	**S_ALEVEL**
Beschreibung:	Abfrage der aktuellen Zugriffsstufe
Rückgabewert:	0: System 1: Hersteller 2: Service 3: Anwender 4: Schlüsselschalter 3 5: Schlüsselschalter 2 6: Schlüsselschalter 1 7: Schlüsselschalter 0

Beispiel

```
REG[0] = S_ALEVEL
```

Siehe auch

ACCESSLEVEL (Seite 120)

7.23 Variable S_CHAN

Beschreibung

Mit der Variablen S_CHAN kann die Nummer des aktuellen Kanals für Zwecke der Anzeige oder einer Auswertung ermittelt werden.

Programmierung

Syntax:	**S_CHAN**
Beschreibung:	Abfrage der aktuellen Kanalnummer
Rückgabewert:	Kanalnummer

Beispiel

```
REG[0] = S_CHAN
```

Siehe auch

CHANNEL (Seite 122)

7.24 Variable S_CONTROL

Beschreibung

Der aktuelle Steuerungsname kann mit der Masken-Eigenschaft S_CONTROL in der Projektierung abgefragt werden.

Programmierung

Syntax:	**S_CONTROL**
Beschreibung:	Abfrage des aktuellen Steuerungsnamens
Rückgabewert:	Der Steuerungsname ist der Sektionsname in der "mmc.ini"

Beispiel

```
REG[0] = S_CONTROL
```

Siehe auch

CONTROL (Seite 122)

7.25 Variable S_LANG

Beschreibung

Die aktuelle Sprache kann mit der Masken-Eigenschaft S_LANG in der Projektierung abgefragt werden.

Programmierung

Syntax:	**S_LANG**
Beschreibung:	Abfrage der aktuellen Sprache
Rückgabewert:	Sprachürzel aus resources.xml z.B. "deu", "eng", etc.

Beispiel

```
REG[0] = S_LANG
```

Siehe auch

LANGUAGE (Seite 124)

7.26 Variable S_NCCODEREADONLY

Beschreibung

Die Masken-Eigenschaft S_NCCODEREADONLY ist ausschließlich von Bedeutung, wenn im Editor ein Zyklus mit einem "Run MyScreens" Dialog rückübersetzt wird. Mit S_NCCODEREADONLY ermitteln Sie, ob ein rückübersetzter NC-Code aus dem Editor geändert werden kann oder nicht.

Für den Rückgabewert gilt

- TRUE: Der rückübersetzter NC-Code aus dem Editor kann geändert werden
- FALSE: Der rückübersetzte NC-Code aus dem Editor kann nicht geändert werden (readonly), da er z. B. schon im Vorlauf ist (= TRUE).

7.27 Variable S_RESX und S_RESY

Beschreibung

Die aktuelle Auflösung bzw. deren X- und Y-Komponente kann mit den Masken-Eigenschaften S_RESX und S_RESY in der Projektierung abgefragt werden.

Beispiel

```
REG[0] = S_RESY
```

Siehe auch

RESOLUTION (Seite 128)

Programmier-Befehle

8

8.1 Operatoren

Übersicht

Bei der Programmierung können folgende Operatoren verwendet werden:

- Mathematische Operatoren
- Vergleichsoperatoren
- Logische (Boolesche) Operatoren
- Bit-Operatoren
- Trigonometrische Funktionen

8.1.1 Mathematische Operatoren

Übersicht

Mathematische Operatoren	Bezeichnung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
MOD	Modulo-Operation
( )	Klammern
AND	UND-Operator
OR	ODER-Operator
NOT	NOT-Operator
ROUND	Zahlen mit Nachkommastellen runden

Beispiel: `VAR1.VAL = 45 * (4 + 3)`

ROUND

Der Operator ROUND wird zur Rundung von Zahlen mit bis zu 12 Nachkommastellen während der Abarbeitung eines Dialogs verwendet. Die Nachkommastellen können von den Variablenfeldern nicht in die Anzeige übernommen werden.

Verwendung

ROUND wird durch zwei Parameter vom Benutzer gesteuert:

```
VAR1 = 5,2328543
VAR2 = ROUND( VAR1, 4 )
```

Ergebnis: `VAR2 = 5,2339`

VAR1 beinhaltet dabei die zu rundende Zahl. Der Parameter "4" gibt die Anzahl der Nachkommastellen im Ergebnis an, das in VAR2 abgelegt wird.

Trigonometrische Funktionen

Trigonometrische Funktionen	Bezeichnung
SIN(x)	Sinus von x
COS(x)	Cosinus von x
TAN(x)	Tangens von x
ATAN(x, y)	Arcus Tangens von x/y
SQRT(x)	Quadratwurzel von x
ABS(x)	Absolutwert von x
SDEG(x)	Umrechnung in Grad
SRAD(x)	Umrechnung in Radiant
CALC_ASIN(x)	Arcus Sinus von x
CALC_ACOS(x)	Arcus Cosinus von x

Hinweis

Die Funktionen arbeiten im Bogenmaß. Zur Umrechnung können die Funktionen SDEG() und SRAD() benutzt werden.

Beispiel: `VAR1.VAL = SQRT(2)`

Mathematische Funktionen

Mathematische Funktionen	Bezeichnung
CALC_CEIL(x)	ermittelt nächsthöhere ganze Zahl von x (aufrunden)
CALC_FLOOR(x)	ermittelt nächstniedrigere ganze Zahl von x (abrunden)
CALC_LOG(x)	ermittelt den (natürlichen) Logarithmus zur Basis e von x
CALC_LOG10(x)	ermittelt den Logarithmus zur Basis 10 von x
CALC_POW(x, y)	ermittelt x hoch y (x zur y-ten Potenz)
CALC_MIN(x, y)	ermittelt die kleinere Zahl von x und y
CALC_MAX(x, y)	ermittelt die kleinere Zahl von x und y

Funktion Random: Zufallszahl

Syntax:	**RANDOM** (unterer Grenzwert, oberer Grenzwert)	
Beschreibung:	Die Funktion **RANDOM** liefert eine Pseudo-Zufallszahl in einem vorgegebenen Bereich.	
Parameter:	unterer Grenzwert	unterer Grenzwert >= -32767, unterer Grenzwert < oberer Grenzwert
	oberer Grenzwert	oberer Grenzwert <= 32767

Beispiel

```
REG[0] = RANDOM(-10,10)         ; Mögliches Ergebnis = -3
```

Konstanten

Konstanten	
PI	3.14159265358979323846
FALSE	0
TRUE	1

Beispiel: `VAR1.VAL = PI`

Vergleichsoperatoren

Vergleichsoperatoren	
==	gleich
<>	ungleich
>	größer
<	kleiner
>=	größer oder gleich
<=	kleiner oder gleich

Beispiel:

```
IF VAR1.VAL == 1
  VAR2.VAL = TRUE
ENDIF
```

Bedingungen

Die Schachtelungstiefe ist unbegrenzt.

Bedingung mit einem Befehl:

```
IF
...
ENDIF
```

Bedingung mit zwei Befehlen:

```
IF
...
ELSE
...
ENDIF
```

8.1.2 Bit-Operatoren

Übersicht

Bit-Operatoren	Bezeichnung
BOR	bitweise OR
BXOR	bitweise XOR
BAND	bitweise AND
BNOT	bitweise NOT
SHL	Bits links verschieben
SHR	Bits rechts verschieben

Operator SHL

Mit dem Operator SHL (SHIFT LEFT) werden Bits nach links geschoben. Dabei kann sowohl der zu schiebende Wert als auch die Anzahl der Schiebeschritte direkt oder als Variable angegeben werden. Wenn die Grenze des Datenformats erreicht ist, werden die Bits ohne Fehlermeldung darüber hinausgeschoben.

Verwendung

Syntax:	variable = wert **SHL**schrittzahl	
Beschreibung:	Schiebe Links	
Parameter:	wert	zu schiebender Wert
	schrittzahl	Anzahl der Schiebeschritte

Beispiel

```
PRESS(VS1)
```

```
  VAR01 = 16 SHL 2           ; Ergebnis = 64
  VAR02 = VAR02 SHL VAR04    ; Inhalt von VAR02 wird in ein 32-Bit unsigned gewan-
                             delt und um den Inhalt von VAR04 Bits nach links gescho-
                             ben. Anschließend wird der 32-Bit Wert wieder in das
                             Format der Variable VAR02 zurückgewandelt.
END_PRESS
```

Operator SHR

Mit dem Operator SHR (SHIFT RIGHT) werden Bits nach rechts geschoben. Dabei kann sowohl der zu schiebende Wert als auch die Anzahl der Schiebeschritte direkt oder als Variable angegeben werden. Wenn die Grenze des Datenformats erreicht ist, werden die Bits ohne Fehlermeldung darüber hinausgeschoben.

Verwendung

Syntax:	variable = wert **SHR**schrittzahl		
Beschreibung:	Schiebe Rechts		
Parameter:	wert	zu schiebender Wert	
	schrittzahl	Anzahl der Schiebeschritte	

Beispiel

```
PRESS(VS1)
  VAR01 = 16 SHR 2           ; Ergebnis = 4
  VAR02 = VAR02 SHR VAR04    ; Inhalt von VAR02 wird in ein 32-Bit unsigned ge-
                             wandelt und um den Inhalt von VAR04 Bits nach
                             rechts geschoben. Anschließend wird der 32-Bit
                             Wert wieder in das Format der Variable VAR02 zu-
                             rückgewandelt.
END_PRESS
```

8.2 Methoden

Übersicht

In Dialogen und dialogabhängigen Softkey-Leisten (Softkey-Leisten, die von einem neu projektierten Dialog aus aufgerufen werden) können durch verschiedene Ereignisse (Eingabefeld verlassen, Betätigung von Softkeys) bestimmte Aktionen ausgelöst werden. Diese Aktionen werden in Methoden projektiert.

Die grundsätzliche Programmierung einer Methode sieht folgendermaßen aus:

Beschreibungsblock	Kommentar	Kapitel-Verweis
`PRESS(HS1)`	;Anfangskennung Methode	
`LM...` `  LS...`	;Funktionen	siehe Kapitel Funktionen (Seite 132)
`Var1.st = ...`	;Änderung von Eigenschaften	siehe Kapitel Softkey-Leisten definieren (Seite 63) und Kapitel Dialogeigenschaften definieren (Seite 51)

`Var2 = Var3 + Var4`
`...`
`EXIT`
	;Berechnung mit Variablen	siehe Kapitel Variablen definieren (Seite 83)

`END_PRESS`	;Endekennung Methode	

8.2.1 ACCESSLEVEL

Beschreibung

Die ACCESSLEVEL-Methode wird durchlaufen, wenn sich bei geöffneter Maske die aktuelle Zugriffsstufe geändert hat.

Programmierung

Syntax:	ACCESSLEVEL <Anweisungen> END_ACCESSLEVEL
Beschreibung:	Zugriffsstufe
Parameter:	- keine -

Siehe auch

Variable S_ALEVEL (Seite 110)

8.2.2 CHANGE

Beschreibung

CHANGE-Methoden werden durchlaufen, wenn sich ein Variablenwert ändert. D.h. innerhalb einer CHANGE-Methode werden Variablen-Berechnungen projektiert, die sofort bei einer Variablenänderung durchlaufen werden.

Es wird unterschieden zwischen elementspezifischer und globaler CHANGE-Methode:

- Die **elementspezifische CHANGE-Methode** wird durchlaufen, wenn sich der Wert der spezifizierten Variablen ändert. Ist einer Variablen eine System- oder Anwendervariable zugeordnet, kann in einer CHANGE-Methode der Wert der Variablen zyklisch aktualisiert werden.
- Die **globale CHANGE-Methode** wird durchlaufen, wenn sich der Wert einer beliebigen Variablen ändert und keine elementspezifische CHANGE-Methode projektiert ist.

Programmierung "elementspezifisch"

Syntax:	CHANGE(Bezeichner) ... END_CHANGE	
Beschreibung:	Änderung des Wertes der spezifizierten Variablen	
Parameter:	Bezeichner	Name der Variablen

Beispiel

```
DEF VAR1=(I//////"DB20.DBB1")        ; Var1 wird eine Systemvariable zugeordnet
CHANGE(VAR1)
  IF VAR1.Val <> 1
  VAR1.st="Werkzeug OK!"             ; Ist der Wert der Systemvariablen ≠ 1, lautet
                                     der Kurztext der Variablen: Werkzeug OK!
    otto=1
  ELSE
    VAR1.st="Achtung Fehler!"        ; Ist der Wert der Systemvariablen = 1, lautet
                                     der Kurztext der Variablen: Achtung Fehler!
    otto=2
  ENDIF
  VAR2.Var=2
END_CHANGE
```

Programmierung "global"

Syntax:	CHANGE() ... END_CHANGE
Beschreibung:	Änderung eines beliebigen Variablenwertes
Parameter:	- keine -

Beispiel

```
CHANGE()
  EXIT                                ; Ändert sich ein beliebiger Variablenwert,
                                      wird der Dialog verlassen.
END_CHANGE
```

8.2.3 CHANNEL

Beschreibung

Die CHANNEL-Methode wird durchlaufen, wenn sich bei geöffneter Maske der aktuelle Kanal geändert hat, d. h. eine Kanalumschaltung stattgefunden hat.

Programmierung

Syntax:	CHANNEL <Anweisungen> END_CHANNEL
Beschreibung:	Kanalumschaltung
Parameter:	- keine -

Siehe auch

Variable S_CHAN (Seite 111)

8.2.4 CONTROL

Beschreibung

Die CONTROL-Methode wird durchlaufen, wenn sich bei geöffneter Maske die aktuelle Steuerung geändert hat, d.h. typischerweise bei einer 1:n-Umschaltung.

Programmierung

Syntax:	CONTROL <Anweisungen> END_CONTROL
Beschreibung:	Umschaltung Steuerung
Parameter:	- keine -

Siehe auch

Variable S_CONTROL (Seite 112)

8.2.5 FOCUS

Beschreibung

Die FOCUS-Methode wird durchlaufen, wenn im Dialog der Fokus (Cursor) auf ein anderes Feld positioniert wird.

Die Methode FOCUS darf nicht mit einem Navigationsereignis ausgelöst werden. Die Cursor-Position darf nur in Softkey PRESS-Methoden, CHANGE-Methoden, ... verändert werden. Die Reaktion der Cursor-Bewegungen ist fest vordefiniert.

Hinweis

Innerhalb der FOCUS-Methode darf nicht auf eine andere Variable positioniert werden und es darf kein neuer Dialog geladen werden.

Programmierung

Syntax:	FOCUS ... END_FOCUS
Beschreibung:	Cursor positionieren
Parameter:	- keine -

Beispiel

```
FOCUS
  DLGL("Der Fokus wurde auf die Variable" << FOC << "gestellt.")
END_FOCUS
```

Siehe auch

Variable FOC (Seite 109)

8.2.6 LANGUAGE

Beschreibung

Die LANGUAGE-Methode wird durchlaufen, wenn sich bei geöffneter Maske die aktuelle Sprache geändert hat.

Programmierung

Syntax:	LANGUAGE <Anweisungen> END_LANGUAGE
Beschreibung:	Sprache
Parameter:	- keine -

Siehe auch

Variable S_LANG (Seite 112)

8.2.7 LOAD

Beschreibung

Die LOAD-Methode wird durchlaufen, nachdem die Variablen- und Softkey-Definitionen (DEF Var1= ..., HS1= ...) interpretiert wurden. Der Dialog ist zu diesem Zeitpunkt noch nicht aufgeblendet.

Programmierung

Syntax:	LOAD ... END_LOAD
Beschreibung:	Laden
Parameter:	- keine -

Beispiel

```
LOAD                          ; Anfangskennung
  Maske1.Hd =  $85111         ; Text für Dialogüberschrift aus Sprachdatei zuweisen
  VAR1.Min = 0                ; Variable Grenzwert MIN zuweisen
  VAR1.Max = 1000             ; Variable Grenzwert MAX zuweisen
END_LOAD                      ; Endekennung
```

Siehe auch

Grafikelemente definieren (Seite 189)

8.2.8 UNLOAD

Beschreibung

Die UNLOAD-Methode wird durchlaufen, bevor ein Dialog entladen wird.

Programmierung

Syntax:	UNLOAD ... END_UNLOAD
Beschreibung:	Entladen
Parameter:	- keine -

Beispiel

```
UNLOAD
  REG[1] = VAR1            ; Variable in Register ablegen
END_UNLOAD
```

8.2.9 OUTPUT

Beschreibung

Die OUTPUT-Methode wird durchlaufen, wenn die Funktion "GC" aufgerufen wird. Innerhalb einer OUTPUT-Methode werden Variable und Hilfsvariable als NC-Code projektiert. Die Verkettung einzelner Elemente einer Codezeile erfolgt mit einem Leerzeichen.

Hinweis

Der NC-Code kann mit den Dateifunktionen in einer extra Datei generiert und zur NC verschoben werden.

Programmierung

Syntax:	OUTPUT(Bezeichner) ... END_OUTPUT	
Beschreibung:	Variablen im NC-Programm ausgeben	
Parameter:	Bezeichner	Name der OUTPUT-Methode

Satznummern und Ausblendkennungen

Die OUTPUT-Methode darf keine Zeilennummern und Ausblendkennungen enthalten, wenn die bei aktiver Programmierunterstützung im Teileprogramm direkt gesetzten Zeilennummern und Ausblendkennungen bei Rückübersetzungen erhalten bleiben sollen.

Änderungen mit dem Editor im Teileprogramm bewirken folgendes Verhalten:

Bedingung	Verhalten
Anzahl Sätze bleibt unverändert.	Satznummern bleiben erhalten.
Anzahl der Sätze wird kleiner.	Die größten Satznummern werden gestrichen.
Anzahl der Sätze wird größer.	Neue Sätze bekommen keine Satznummern.

Beispiel

```
OUTPUT(CODE1)
  "CYCLE82(" Var1.val "," Var2.val "," Var3.val ","Var4.val "," Var5.val "," Var6.val ")"
END_OUTPUT
```

8.2.10 PRESS

Beschreibung

Die Methode PRESS wird durchlaufen, wenn der entsprechende Softkey gedrückt wurde.

Programmierung

Syntax:	PRESS(Softkey) ... END_PRESS
Bezeichnung:	Drücken eines Softkeys

Parameter:	Softkey	Name des Softkeys: HS1 - HS8 und VS1 - VS8	
	RECALL	Taste <RECALL>	
	ENTER	Taste <ENTER>, siehe PRESS(ENTER) (Seite 127)	
	TOGGLE	Taste <TOGGLE>, siehe PRESS(TOGGLE) (Seite 128)	
	PU	Page Up	Bild aufwärts
	PD	Page Down	Bild abwärts
	SL	Scroll Left	Cursor links
	SR	Scroll Right	Cursor rechts
	SU	Scroll Up	Cursor aufwärts
	SD	Scroll Down	Cursor abwärts

Beispiel

```
HS1 = ("andere Softkey-Leiste")
HS2 = ("keine Funktion")
PRESS (HS1)
  LS("Leiste1")                    ; andere Softkey-Leiste laden
  Var2 = Var3 + Var1
END_PRESS
PRESS (HS2)
END_PRESS
PRESS(PU)
  INDEX = INDEX -7
  CALL("UP1")
END_PRESS
```

8.2.11 PRESS(ENTER)

Beschreibung

Die PRESS(ENTER)-Methode wird immer dann aufgerufen, wenn bei einer Variable mit Ein-/Ausgabefeld mit Eingabemodus WR3 oder WR5 die Enter-Taste gedrückt wird:

- WR3: Navigation auf Feld und Dücken der Enter-Taste
- WR5: Im Eingabemodus, Übernahme des Wertes mit der Enter-Taste

Programmierung

Syntax:	PRESS(ENTER) <Anweisungen> END_PRESS
Beschreibung:	Enter-Taste gedrückt
Parameter:	- keine -

8.2.12 PRESS(TOGGLE)

Beschreibung

Die PRESS(TOGGLE)-Methode wird immer dann aufgerufen, wenn unabhängig von der aktuell fokussierten Variable die Toggle-Taste gedrückt wird.

Bei Bedarf, kann mit Hilfe der Masken-Eigenschaft FOC ermittelt werden, welche Variable aktuell den Fokus hat.

Programmierung

Syntax:	PRESS(TOGGLE) <Anweisungen> END_PRESS
Beschreibung:	Toggle-Taste gedrückt
Parameter:	- keine -

Beispiel

```
PRESS(TOGGLE)
   DLGL("Toggle key pressed at variable " << FOC)        ; mit der Masken-Eigenschaft FOC wird ermit-
                                                         telt, welche Variable aktuell den Fokus hat
END_PRESS
```

8.2.13 RESOLUTION

Beschreibung

Die RESOLUTION-Methode wird durchlaufen, wenn sich bei geöffneter Maske die aktuelle Auflösung geändert hat, d.h. typischerweise bei einer TCU-Umschaltung.

Programmierung

Syntax:	RESOLUTION <Anweisungen> END_RESOLUTION
Beschreibung:	Auflösung
Parameter:	- keine -

Siehe auch

Variable S_RESX und S_RESY (Seite 113)

8.2.14 RESUME

Beschreibung

Die RESUME-Methode wird aufgerufen, wenn die Maske z. B. durch einen Area-Wechsel unterbrochen wurde und nun wieder aufgeblendet wird. Dabei werden die Wertänderungen wieder bearbeitet, ggf. die Timer gestartet und der RESUME-Block abgearbeitet.

Hinweis

Die Funktionen LS, LM, DLGL, EXIT und EXITLS dürfen in den Methoden SUSPEND bzw. RESUME nicht projektiert werden. Bei Verwendung der Funktionen in diesen Methoden wird die Ausführung der Funktionen verhindert.

Programmierung

Syntax:	RESUME <Anweisung> END_RESUME
Beschreibung:	Maske wieder aktiv
Parameter:	- keine -

8.2.15 SUSPEND

Beschreibung

Die SUSPEND-Methode wird aufgerufen, wenn die Maske unterbrochen und nicht entladen wird. Dies ist z. B. dann der Fall, wenn die Maske durch einfachen Area-Wechsel verlassen, aber nicht explizit entladen wird. Die Maske bleibt dabei im Hintergrund erhalten, jedoch werden dabei keine Wertänderung und ggf. Timer bearbeitet. Die Maske ist pausiert.

Hinweis

Die Funktionen LS, LM, DLGL, EXIT und EXITLS dürfen in den Methoden SUSPEND bzw. RESUME nicht projektiert werden. Bei Verwendung der Funktionen in diesen Methoden wird die Ausführung der Funktionen verhindert.

Programmierung

Syntax:	SUSPEND <Anweisung> END_SUSPEND
Beschreibung:	Unterbrechung der Maske
Parameter:	- keine -

Beispiel

```
SUSPEND
   MyVar1 = MyVar1 + 1
END_SUSPEND
```

8.2.16 Beispiel: Versionsverwaltung mit OUTPUT-Methoden

Übersicht

Bestehende Dialoge können im Zuge von Erweiterungen um zusätzliche Variablen ergänzt werden. Die zusätzlichen Variablen erhalten in den Definitionen hinter dem Variablennamen in runden Klammern eine Versionskennzahl: (0 = Original, wird nicht geschrieben), 1 = Version 1, 2 = Version 2, ...

Beispiel:

```
DEF var100=(R//1)              ; Original, entspricht Version 0
DEF var101(1)=(S//"Hallo")     ; Ergänzung ab Version 1
```

Beim Schreiben der OUTPUT-Methode kann auf einen bestimmten Versionsstand, bezogen auf die Gesamtheit der Definitionen, Bezug genommen werden.

Beispiel:

```
OUTPUT(NC1)                    ; Nur die Variablen des Originals werden in der OUT-
                               PUT-Methode angeboten.
OUTPUT(NC1,1)                  ; Die Variablen des Originals und die Ergänzungen
                               mit Versionskennzeichen 1 werden in der OUTPUT-Me-
                               thode angeboten.
```

Die OUTPUT-Methode für das Original benötigt keine Versionskennzeichnung, es kann jedoch auch 0 geschrieben werden. OUTPUT(NC1) entspricht OUTPUT(NC1,0). Versionskennzeichen n in der OUTPUT-Methode umfasst alle Variablen des Originals 0, 1, 2, ... bis einschließlich n.

Programmierung mit Versionskennung

```
//M(XXX)                          Version 0 (Default)
DEF var100=(R//1)
DEF var101=(S//"Hallo")
DEF TMP
VS8=("GC")
PRESS(VS8)
   GC("NC1")
END_PRESS

OUTPUT(NC1)
var100",,"var101
END_OUTPUT

; ************ Version 1, ergänzte Definition ***************
//M(XXX)
DEF var100=(R//1)
DEF var101=(S//"Hallo")
DEF var102(1)=(V//"HUGO")
DEF TMP
VS8=("GC")
PRESS(VS8)
   GC("NC1")
END_PRESS
...

OUTPUT(NC1)                        Original und zusätzlich die neue Version
var100","var101
END_OUTPUT
...
```

```
OUTPUT(NC1,1)                    Version 1
var100","var101"," var102
END_OUTPUT
```

8.3 Funktionen

Übersicht

In Dialogen und dialogabhängigen Softkey-Leisten stehen verschiedene Funktionen zur Verfügung, die durch Ereignisse, z. B. Eingabefeld verlassen, Softkeys betätigen, ausgelöst und in Methoden projektiert werden.

Unterprogramme

Sich wiederholende oder auch andere Projektierungsanweisungen, die einen bestimmten Vorgang zusammenfassen, können in Unterprogrammen projektiert werden. Unterprogramme können jederzeit ins Haupt- oder in andere Unterprogramme geladen und beliebig oft abgearbeitet werden, d.h. die Anweisungen müssen nicht mehrfach projektiert werden. Als Hauptprogramm gelten die Beschreibungsblöcke der Dialoge oder Softkey-Leiste.

Externe Funktionen

Mit Hilfe von externen Funktionen können weitere, anwenderspezifische Funktionen eingebracht werden. Die externen Funktionen werden in einer DLL-Datei hinterlegt und durch einen Eintrag in den Definitionszeilen der Projektierungsdatei bekannt gemacht.

PI-Dienste

Mit der Funktion PI_START können PI-Dienste (Programm Instanz Dienste) von der PLC im NC-Bereich gestartet werden.

Siehe auch

Function (FCT) (Seite 152)

PI-Dienste (Seite 167)

8.3.1 Antriebsparameter lesen und schreiben: RDOP, WDOP, MRDOP

Beschreibung

Mit Hilfe der Funktionen RDOP, WDOP und MRDOP können Sie Antriebsparameter lesen bzw. schreiben.

Hinweis

Antriebsparameter nicht schneller als im 1-Sekunden-Takt lesen, besser langsamer.

Grund: Die Kommunikation zu den Antrieben kann sonst gegebenenfalls extrem gestört werden oder gar Ausfälle verursachen.

Hinweis

Sollte es beim Lesen oder Schreiben von Antriebsparametern zu Fehlern kommen, dann wird die Maskenproperty ERR entsprechend gesetzt.

Programmierung

Syntax:	**RDOP**("Bezeichner des Antriebsobjektes", "Parameternummer")	
Beschreibung:	Lesen eines Antriebsparameters (Drive Object Parameter)	
Parameter:	Bezeichner des Antriebsobjektes	Der Bezeichner des Antriebsobjektes kann der Spalte "DO Name" in "Antriebssystem Diagnose" im Bedienbereich Diagnose (> ETC > HSK 8: Antriebssystem) entnommen werden (siehe Abbildung unten).
	Parameternummer	

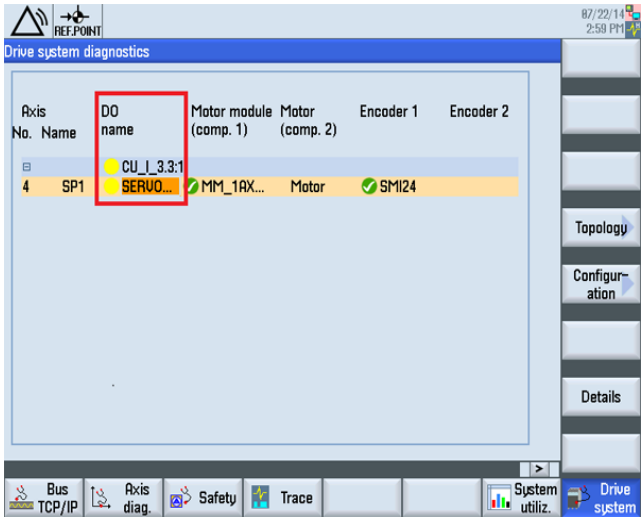


Bild 8-1 Bezeichner des Antriebsobjektes

Syntax:	**MRDOP**("Bezeichner des Antriebsobjektes", "Parameternummer1"*"Parameternummer"[*...], Registerindex)	
Beschreibung:	Multi Read von Antriebsparametern. Mit dem Kommando MRDOP können mehrere Antriebsparameter eines Antriebsobjektes mit einem Zugriff in das Register übertragen werden. Dieser Zugriff ist deutlich schneller als das Lesen über Einzelzugriffe.	
Parameter:	Bezeichner des Antriebsobjektes	Der Bezeichner des Antriebsobjektes kann z. B. dem Grundbild vom Bedienbereich „Inbetriebnahme“ entnommen werden
	Parameternummer1 ... n	Bei den Variablennamen gilt "*" als Trennzeichen. In der Reihenfolge wie die Variablennamen im Kommando stehen, werden die Werte in die Register REG[Registerindex] übernommen.
	Registerindex	Der Wert der ersten Variable steht in REG[Registerindex]. Der Wert der zweiten Variable steht in REG[Registerindex + 1]

Syntax:	**WDOP**("Bezeichner des Antriebsobjektes", "Parameternummer", Wert)	
Beschreibung:	Schreiben eines Antriebsparameters (Drive Object Parameter)	
Parameter:	Bezeichner des Antriebsobjektes	Der Bezeichner des Antriebsobjektes kann z. B. dem Grundbild vom Bedienbereich „Inbetriebnahme“ entnommen werden
	Parameternummer	Parameternummer
	Wert	Zu schreibender Wert

Beispiele

Motortemperatur r0035 des Antriebsobjektes "SERVO_3.3:2" lesen:

```
MyVar=RDOP("SERVO_3.3:2","35") ;
```

Motortemperatur r0035 und Drehmomentistwert r0080 des Antriebsobjektes "SERVO_3.3:2" lesen und die jeweiligen Ergebnisse ab Registerindex 10 ablegen:

```
MRDOP("SERVO_3.3:2","35*80",10)
```

8.3.2 Aufruf Unterprogramm (CALL)

Beschreibung

Mit der Funktion CALL kann ein geladenes Unterprogramm von jeder beliebigen Stelle einer Methode aufgerufen werden. Eine Verschachtelung, d.h. der Aufruf eines Unterprogramms von einem Unterprogramm aus, ist zulässig.

Programmierung

Syntax:	**CALL**("Bezeichner")	
Beschreibung:	Unterprogramm aufrufen	
Parameter:	Bezeichner	Name des Unterprogramms

Beispiel

```
//M(MASKE1)
DEF VAR1 = ...
DEF VAR2 = ...
CHANGE(VAR1)
  ...
  CALL("MY_UP1")              ; Unterprogramm aufrufen und abarbeiten
  ...
END_CHANGE

CHANGE(VAR2)
  ...
  CALL("MY_UP1")              ; Unterprogramm aufrufen und abarbeiten
  ...
END_CHANGE

SUB(MY_UP1)
                              ;do something
END_SUB
//END
```

8.3.3 Block definieren (//B)

Beschreibung

Unterprogramme werden in der Programmdatei mit der Blockkennung //B gekennzeichnet und durch //END beendet. Pro Blockkennung können mehrere Unterprogramme definiert werden.

Hinweis

Die im Unterprogramm verwendeten Variablen müssen in dem Dialog definiert sein, in dem das Unterprogramm aufgerufen wird.

Programmierung

Ein Block hat den folgenden Aufbau:

Syntax:	**//B**(Blockname) **SUB**(Bezeichner) END_SUB [SUB(Bezeichner) ... END_SUB] ... //END	
Beschreibung:	Unterprogramm definieren	
Parameter:	Blockname	Name der Blockkennung
	Bezeichner	Name des Unterprogramms

Beispiel

```
//B(PROG1)                        ; Blockanfang
SUB(UP1)                          ; Unterprogramm-Anfang
  ...
  REG[0] = 5                      ; Register 0 mit dem Wert 5 belegen
  ...
END_SUB                           ; Unterprogramm-Ende
SUB(UP2)                          ; Unterprogramm-Anfang
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN
  ENDIF
  VAR1.val="Otto"
END_SUB                           ; Unterprogramm-Ende
//END                             ; Blockende
```

8.3.4 Check Variable (CVAR)

Beschreibung

Mit Hilfe der Funktion CVAR (Check Variable) kann abgefragt werden, ob alle oder nur bestimmte Variablen oder Hilfsvariablen eines Dialogs fehlerfrei sind.

Eine Abfrage, ob Variable einen gültigen Wert enthalten, ist z.B. sinnvoll, bevor mit der Funktion GC NC-Code erzeugt wird.

Eine Variable ist fehlerfrei, wenn der Zustand der Variablen Bezeichner.vld = 1 ist.

Programmierung

Syntax:	**CVAR**(VarN)	
Beschreibung:	Variablen auf gültigen Inhalt prüfen	
Parameter:	VarN	Auflistung der zu prüfenden Variablen. Es können bis zu 29 Variablen durch Komma getrennt überprüft werden. Dabei ist die maximale Zeichenlänge von 500 zu beachten. Das Ergebnis der Abfrage kann sein:
	1 =	TRUE (alle Variablen haben gültigen Inhalt)
	0 =	FALSE (mindestens eine Variable hat keinen gültigen Inhalt)

Beispiel

```
IF CVAR == TRUE                          ; Prüfung aller Variablen
  VS8.SE = 1                             ; Sind alle Variablen fehlerfrei, ist der
                                         Softkey VS8 sichtbar
ELSE
  VS8.SE = 2                             ; Enthält eine Variable einen fehlerhaf-
                                         ten Wert, ist der Softkey VS8 nicht be-
                                         dienbar
ENDIF

IF CVAR("VAR1", "VAR2") == TRUE
                                         ; Prüfung der Variablen VAR1 und VAR2
  DLGL ("VAR1 und VAR2 sind OK")
                                         ; Sind VAR1 und VAR2 fehlerfrei ausge-
                                         füllt, erscheint in der Dialogzeile "VAR1
                                         und VAR2 sind OK"
ELSE
  DLGL ("VAR1 und VAR2 sind nicht OK")
                                         ; Sind VAR1 und VAR2 nicht fehlerfrei aus-
                                         gefüllt, erscheint in der Dialogzeile
                                         "VAR1 und VAR2 sind nicht OK"
ENDIF
```

8.3.5 Dateifunktion Copy Program (CP)

Beschreibung

Die Funktion CP (Copy Program) kopiert Dateien innerhalb des HMI-Dateisystems oder innerhalb des NC-Dateisystems.

Programmierung

Syntax:	**CP**("Quelldatei", "Zieldatei")	
Beschreibung:	Datei kopieren	
Parameter:	Quelldatei	Vollständige Pfadangabe der Quelldatei
	Zieldatei	Vollständige Pfadangabe der Zieldatei

Über den Rückgabewert (VAR1 als Hilfsvariable definiert) kann abgefragt werden, ob die Funktion erfolgreich war:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)
```

Beispiel

Anwendungsfall mit Rückgabewert:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)
CP("CF_CARD:/wks.dir/MESS_BILD.WPD/MESS_BILD.MPF","//NC/WKS.DIR/AAA.WPD/HOHO2.MPF",VAR1)
CP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/wks.dir/WST1.WPD/MESS.MPF",VAR1)        ; WPD muss existieren
```

Anwendungsfall ohne Rückgabewert:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF")
CP("CF_CARD:/mpf.dir/MYPROG.MPF","//NC/MPF.DIR/HOHO.MPF")
CP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/XYZ/MYPROG.MPF")                          ; XYZ muss existieren
```

Siehe auch

Unterstützung von FILE_ERR: Variable FILE_ERR (Seite 108)

8.3.6 Dateifunktion Delete Program (DP)

Beschreibung

Die Funktion DP (Delete Program) löscht eine Datei des passiven HMI-Dateisystems oder des aktiven NC-Dateisystems.

Programmierung

Syntax:	**DP**("Datei")	
Beschreibung:	Datei löschen	
Parameter:	Datei	Vollständige Pfadangabe der zu löschenden Datei

Beispiel

Für diese Funktion wird folgende Syntax der Datenhaltung benutzt:

- mit Rückgabewert

  ```
  DP("//NC/MPF.DIR/MYPROG.MPF",VAR1)
  DP("//NC/WKS.DIR/TEST.WPD/MYPROG.MPF",VAR1)
  DP("//NC/CMA.DIR/MYPROG.SPF",VAR1)
  ```

  VAR1 = 0 Datei wurde gelöscht.

  VAR1 = 1 Datei wurde nicht gelöscht.
- ohne Rückgabewert:

  ```
  DP("//NC/MPF.DIR/MYPROG.MPF")
  DP("//NC/WKS.DIR/TEST.WPD/MYPROG.MPF")
  DP("//NC/CMA.DIR/MYPROG.SPF")
  ```

8.3.7 Dateifunktion Exist Program (EP)

Beschreibung

Die Funktion EP (Exist Program) prüft, ob sich ein bestimmtes NC-Programm im Dateisystem der NC oder im HMI-Dateisystem unter dem angegebenen Pfad befindet.

Programmierung

Syntax:	**EP**("Datei")	
Beschreibung:	Existenz des NC-Programms prüfen	
Parameter:	Datei	Vollständige Pfadangabe der Datei im NC- oder HMI-Dateisystem
Rückgabewert:	Name einer Variablen, der das Ergebnis der Abfrage zugeordnet werden soll.	

Die Funktion EP beherrscht die neue Syntax und die alte Logik (mit angepasster Syntax).

Die Datei wird direkt mit einem qualifizierenden Namen angesprochen:

```
//NC/MPF.DIR/XYZ.MPF
```

oder
`CF_CARD: /MPF.DIR/XYZ.MPF` (zeigt auf /user/sinumerik/data/prog)
oder
`LOC:` (entspricht CF_CARD)

Beispiele für neue Syntax:

```
EP("//NC/WKS.DIR/TEST.WPD/XYZ.MPF",VAR1)
EP("CF_CARD:/mpf.dir/XYZ.MPF",VAR1)
EP("LOC:/mpf.dir/XYZ.MPF",VAR1)
; mit Rückgabewert:
; VAR1 = 0 Datei existiert.
; VAR1 = 1 Datei existiert nicht.
```

Beispiel für alte Syntax:

```
EP("/MPF.DIR/CFI.MPF", VAR1)
; mit Rückgabewert:
; VAR1 = M Datei liegt im HMI-Dateisystem.
; VAR1 = N Datei liegt im NC-Dateisystem.
; VAR1 = B Datei liegt im HMI- und NC-Dateisystem.
```

Beispiel

```
EP("/MPF.DIR/GROB.MPF",VAR1)                                    ; alt - Pfad jetzt mit / anstatt  \

IF VAR1 == "M"
   DLGL("Datei befindet sich im HMI-Dateisystem")
ELSE
   IF VAR1 == "N"
      DLGL("Datei befindet sich im NC-Dateiverzeichnis")
   ELSE
     DLGL("Datei befindet sich weder im HMI- noch NC-Datei-
system")
   ENDIF
ENDIF
```

8.3.8 Dateifunktion Move Program (MP)

Beschreibung

Die Funktion MP (Move Program) kopiert Dateien innerhalb des HMI-Dateisystems oder innerhalb des NC-Dateisystems.

Programmierung

Syntax:	**MP**("Quelle", "Ziel")	
	`MP("CF_CARD:/MPF.DIR/MYPROG.MPF","//NC/MPF.DIR")`	
Beschreibung:	Datei verschieben	
Parameter:	Quelldatei	Vollständige Pfadangabe
	Zieldatei	Vollständige Pfadangabe
Rückgabewert	Ergebnis der Abfrage	

Beispiele

Mit Rückgabewert:

```
MP("//NC/MPF.DIR/123.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)          ;innerhalb NC
MP("//NC/MPF.DIR/123.MPF",VAR0,VAR1)                              ;Ziel über Variable
MP(VAR4,VAR0,VAR1)                                                ;Quelle und Ziel über Variable
MP("CF_CARD:/mpf.dir/myprog.mpf","//NC/MPF.DIR/123.MPF",VAR1)     ;von CF-Karte in NC
MP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/XYZ/123.mpf",VAR1)           ;von NC in CF-Karte

; mit Rückgabewert:
; VAR1 = 0 ausgeführt
; VAR1 = 1 nicht ausgeführt
```

8.3.9 Dateifunktion Select Program (SP)

Beschreibung

Die Funktion SP (Select Program) wählt eine Datei des aktiven NC-Dateisystems an, um sie abzuarbeiten, d.h. die Datei muss vorher in die NC geladen werden.

Programmierung

Syntax:	**SP**("Datei")	
Bezeichnung:	Programm wählen	
Parameter:	"Datei"	Vollständige Pfadangabe der NC-Datei

Beispiel

Für diese Funktion wird folgende Syntax der Datenhaltung benutzt:

- mit Rückgabewert

  `SP("//NC/MPF.DIR/MYPROG.MPF", VAR1)`

  VAR1 = 0 Datei wurde geladen.

VAR1 = 1 Datei wurde nicht geladen.

- ohne Rückgabewert:

```
SP("//NC/MPF.DIR/MYPROG.MPF")
```

8.3.10 Dateizugriffe: RDFILE, WRFILE, RDLINEFILE, WRLINEFILE

Beschreibung

Für den lesenden und schreibenden Zugriff auf Dateien mit INI-Syntax stehen Ihnen die Funktionen RDFILE und WRFILE zur Verfügung.

Für den lesenden und schreibenden Zugriff auf einzelne Zeilen einer Datei stehen Ihnen die Funktionen RDLINEFILE und WRLINEFILE zur Verfügung

Programmierung

Syntax:	**RDFILE**("Filename + Pfad", "Section", "Key")	
Beschreibung:	Lesen aus einer Datei	
Parameter:	Filename + Pfad	Pfad- und Dateiname
	Section	Sektion in der INI-Datei
	Key	Schlüssel in der INI-Datei

Syntax:	**WRFILE**(Value, "Filename + Pfad", "Section", "Key")	
Beschreibung:	Schreiben in eine Datei	
Parameter:	Value	zu schreibender Wert
	Filename + Pfad	Pfad- und Dateiname
	Section	Sektion in der INI-Datei
	Key	Schlüssel in der INI-Datei

Syntax:	**RDLINEFILE**("Filename + Pfad", Zeilennummer)	
Beschreibung:	Lesen einer Zeile aus einer Datei	
Parameter:	Filename + Pfad	Pfad- und Dateiname
	Zeilennummer	Zeilennummer Die Nummerierung beginnt bei 0 für die erste Zeile.

Syntax:	**WRLINEFILE**(Wert, "Filename + Pfad")
Beschreibung:	Schreiben einer Zeile an das Ende einer Datei

Parameter:	Wert	zu schreibender Wert
	Filename + Pfad	Pfad- und Dateiname

Hinweis

- Die Dateien dürfen nicht im Dateisystem der NC (Datenhaltung) liegen.
- Existiert die Datei nicht, wird das Dateiende erreicht oder kommt es zu sonstigen Fehlern, dann wird die Variable FILE_ERR und ERR entsprechend gesetzt. Ob eine Datei existiert, können Sie vorab mit der Dateifunktion Exist Program (EP) überprüfen.
- Die bearbeitete Datei wird in der Kodierung "UTF-8" (ohne BOM (Byte Order Mask) erzeugt. Beim Lesen einer Datei wird diese in der Kodierung "UTF-8" erwartet.
- Mit der Dateifunktion Delete Program (DP) können Sie die Datei bei Bedarf explizit löschen.

Beispiele

Lesen aus einer INI-Datei:

Voraussetzung/Annahme:

Inhalt von File "C:/tmp/myfile.ini":

```
<...>
[MyData]

MyName=Daniel

<...>
```

```
MyVar = RDFILE("C:/tmp/myfile.ini", "MyData", "MyName")
```

Ergebnis:

MyVar enthält jetzt den Wert "Daniel".

Schreiben in eine INI-Datei:

Voraussetzung/Annahme:

VARS=12

```
WRFILE(VARS, "C:/tmp/myfile.ini", "MySession", "NrOfSessions")
```

Ergebnis:

Inhalt von File "C:/tmp/myfile.ini":

```
<...>
[MySession]

NrOfSessions=12

<...>
```

Lesen der 4. Zeile einer Datei:

Voraussetzung/Annahme:

Inhalt von c:/tmp/myfile.mpf:

```
R[0]=0
F3500 G1
MYLOOPIDX1: X0 y-50 Z0
  X150 Y50 Z10
  R[0]=R[0]+1
GOTOB MYLOOPIDX1
M30
```

```
MyVar = RDLINEFILE("C:/tmp/myfile.mpf", 4)
```

Ergebnis:

MyVar enthält jetzt den Wert " `R[0]=R[0]+1`"

Schreiben ans Ende einer Datei:

Voraussetzung/Annahme:

```
VARX=123
VARY=456
```

```
WRLINEFILE("F100 X" << VARX << " Y" << VARY, "C:/tmp/mypp.mpf")
```

Ergebnis:

Inhalt von c:/tmp/mypp.mpf:

```
<...>
F100 X123 Y456
```

8.3.11 Dialog Line (DLGL)

Beschreibung

In der Dialogzeile des Dialogs können abhängig von bestimmten Situationen kurze Texte (Meldungen oder Eingabehilfen) ausgegeben werden.

Mögliche Anzahl an Zeichen bei Standard-Schriftgröße: ca. 50 Zeichen

Hinweis

Die Funktionen LS, LM, DLGL, EXIT und EXITLS dürfen in den Methoden SUSPEND bzw. RESUME nicht projektiert werden. Bei Verwendung der Funktionen in diesen Methoden wird die Ausführung der Funktionen verhindert.

Programmierung

Syntax:	**DLGL**("String")	
Beschreibung:	Text in der Dialogzeile ausgeben	
Parameter:	String	Text, der in der Dialogzeile erscheint

Beispiel

```
IF Var1 > Var2
                              ; In der Dialogzeile erscheint der Text "Wert zu
                              groß!", wenn Variable1>Variable2 ist.
  DLGL("Wert zu groß!")
ENDIF
```

8.3.12 DEBUG

Beschreibung

Mit der Funktion DEBUG steht Ihnen in der Entwurfsphase von Run MyScreen Anwendermasken eine Analysehilfe zur Verfügung. Mit der Funktion DEBUG wird ein in Klammern übergebener Ausdruck zur Laufzeit ausgewertet. Das Ergebnis wird in das Logbuch "easyscreen_log.txt“ als eigenständiger Eintrag angehängt.

Jedem Eintrag wird in eckigen Klammern der aktuelle Zeitstempel vorangestellt (siehe Beispiel unten).

Es wird empfohlen, die DEBUG-Ausgaben nach Möglichkeit in zeitkritischen Teilen wieder zu entfernen. Insbesondere nach Abschluss der Entwurfsphase wird empfohlen die DEBUG-Ausgaben auszukommentieren. Die Schreibzugriffe in das stetig anwachsende Logbuch "easyscreen_log.txt“ führen zu einer Verlangsamung.

Programmierung

Syntax:	**DEBUG**(Ausdruck)
Beschreibung:	Eintrag im Logbuch "easyscreen_log.txt“ vornehmen
Parameter:	Auszuwertender Ausdruck, aus dem ein Eintrag im Logbuch generiert wird

Beispiel

```
IF Var1 > Var2
  DEBUG("Value of ""Var1"": " << Var1)
                                        ; Eintrag in der "easyscreen_log_txt:
                                        [10:22:40.445] DEBUG: Value of "Var1":
                                        123.456
ENDIF
```

8.3.13 Dialog verlassen (EXIT)

Beschreibung

Mit der Funktion EXIT wird ein Dialog verlassen und in den Haupt-Dialog zurückgekehrt. Existiert kein Haupt-Dialog, verlassen Sie die neu gestalteten Bedienoberflächen und kehren zur Standardapplikation zurück.

Hinweis

Die Funktionen LS, LM, DLGL, EXIT und EXITLS dürfen in den Methoden SUSPEND bzw. RESUME nicht projektiert werden. Bei Verwendung der Funktionen in diesen Methoden wird die Ausführung der Funktionen verhindert.

Programmierung (ohne Parameter)

Syntax:	**EXIT**
Beschreibung:	Dialog verlassen
Parameter:	- keine -

Beispiel

```
PRESS(HS1)
  EXIT
END_PRESS
```

Beschreibung

Wurde der aktuelle Dialog mit Übergabevariable aufgerufen, kann der Wert der Variablen geändert und in den Ausgangsdialog zurückgegeben werden.

Die Variablenwerte werden jeweils den Variablen zugeordnet, die mit der Funktion "LM" vom Ausgangsdialog an den Folgedialog übergeben wurden. Es können bis zu 20 Variablenwerte jeweils durch Komma getrennt übergeben werden.

Hinweis

Die Reihenfolge der Variablen oder Variablenwerte muss entsprechend der Reihenfolge der Übergabevariablen der LM-Funktion erfolgen, damit die Zuordnung eindeutig ist. Werden einige Variablenwerte nicht angegeben, werden diese Übergabevariablen nicht verändert. Die geänderten Übergabevariablen sind sofort nach der Funktion LM im Ausgangsdialog gültig.

Programmierung mit Übergabevariable

Syntax:	**EXIT**[(VARx)]	
Beschreibung:	Dialog verlassen mit Übergabe einer oder mehrerer Variablen	
Parameter:	VARx	Bezeichnung der Variablen

Beispiel

```
//M(Maske1)
...
PRESS(HS1)
  LM("MASKE2","CFI.COM",1, POSX, POSY,
DURCHMESSER)
                                        ; Maske1 unterbrechen und Maske2 aufblen-
                                        den. Dabei die Variablen POSX, POSY und
                                        DURCHMESSER übergeben.
  DLGL("Maske2 beendet")                ; Nach der Rückkehr aus Maske2 erscheint
                                        in der Dialogzeile von Maske1 der Text:
                                        Maske2 beendet.
END_PRESS
...
//END

//M(Maske2)
...
PRESS(HS1)
  EXIT(5, , BERECHNETER_DURCHMESSER)
                                        ; Maske2 verlassen und zurück zu Maske1
                                        in die Zeile nach LM kehren. Dabei der Va-
                                        riablen POSX den Wert 5 und der Variablen
                                        DURCHMESSER den Wert der Variablen BERECH-
                                        NETER_DURCHMESSER zuweisen. Die Variable
                                        POSY behält ihren aktuellen Wert.
END_PRESS
...
```

```
//END
```

8.3.14 Dynamische Manipulation der Listen von Toggle- oder ListBox-Feldern

Beschreibung

Die Funktionen LISTADDITEM, LISTINSERTITEM, LISTDELETEITEM und LISTCLEAR dienen zur dynamischen Manipulation der Listen von Toggle- oder ListBox-Feldern.

Diese Funktionen wirken nur auf Variablen, die ihre eigene Liste haben wie z.B.

- "einfache" Liste
  `DEF VAR_AC1 = (I/* 0,1,2,3,4,5,6,7,8)` oder
- "erweiterte" Liste
  `DEF VAR_AC2 = (I/* 0="AC0", 1="AC1", 2="AC2", 3="AC3", 4="AC4", 5="AC5", 6="AC6", 7="AC7", 8="AC8").`

Wenn die Variable auf ein Array zeigt, z. B. `DEF VAR_AC3 = (I/* MYARRAY)`, stehen diese Funktionen nicht zur Verfügung, da sonst das globale Array verändert werden würde.

Eine Variable muss in der DEF-Zeile mindestens einen Wert definiert haben. Damit wird der Typ "einfache" oder "erweiterte" Liste festgelegt.

Danach ist es jedoch erlaubt die Liste komplett zu löschen und gegebenenfalls komplett neu aufzubauen. Der Typ "einfach" oder "erweitert" muss aber beibehalten werden bzw. er kann nicht dynamisch geändert werden.

Programmierung

Syntax:	**LISTINSERTITEM**(Variablename, Position, ItemValue[, ItemDispValue])	
Beschreibung:	Einfügen eines Elementes an einer bestimmten PositionLesen aus einer Datei	
Parameter:	Variablenname	
	Position	Position, an der ein Element in die Liste hinzugefügt werden soll
	ItemValue	Wert des Listeneintrages
	ItemDispValue	Wert, wie er in der Liste dargestellt werden soll

Syntax:	**LISTADDITEM**(Variablename, ItemValue[, ItemDispValue])
Beschreibung:	Anhängen eines Elementes ans Ende der Liste

Parameter:	Variablenname	
	ItemValue	Wert des Listeneintrages
	ItemDispValue	Wert, wie er in der Liste dargestellt werden soll

Syntax:	**LISTDELETEITEM**(Variablename, Position)	
Beschreibung:	Löschen eines bestimmten Elementes	
Parameter:	Variablenname	
	Position	Position des zu löschenden Elementes in der Liste

Syntax:	**LISTCOUNT**(Variablename)	
Beschreibung:	Liefert die aktuelle Anzahl der Elemente der Liste	
Parameter:	Variablenname	

Syntax:	**LISTCLEAR**(Variablename)	
Beschreibung:	Löschen der kompletten Liste	
Parameter:	Variablenname	

Beispiele

Hinweis

Die nachfolgenden Beispiele bauen aufeinander auf. Für das Nachvollziehen der jeweiligen Ergebnisse ist die Reihenfolge der Beispiele relevant.

Voraussetzung/Annahme:

```
DEF VAR_AC = (I/* 0="Off",1="On"/1/,"Switch"/WR2)
```

Anhängen eines Elementes -1="Undefined" ans Ende der Liste:

```
LISTADDITEM("VAR_AC", -1, """Undefined""")
```

Ergebnis: `0="Off", 1="On", -1="Undefined"`

Einfügen eines Elementes 99="Maybe" an Position 2:

```
LISTINSERTITEM("VAR_AC", 2, 99, """Maybe""")
```

Ergebnis: `0="Off", 1="On", 99="Maybe", -1="Undefined"`

Ermitteln der aktuellen Anzahl von Listenelementen:

```
REG[10]=LISTCOUNT("VAR_AC")
```

Ergebnis: REG[10] = 4

Löschen des Elementes an Position 1:

```
LISTDELETEITEM("VAR_AC", 1)
```

Ergebnis: `0="Off", 99="Maybe", -1="Undefined"`

Löschen der kompletten Liste:

```
LISTCLEAR("VAR_AC")
```

Ergebnis: Liste ist leer

8.3.15 Evaluate (EVAL)

Beschreibung

Die Funktion EVAL wertet einen übergebenen Ausdruck aus und führt ihn dann aus. Damit können Ausdrücke erst zur Laufzeit erstellt werden. Dies ist z.B. nützlich für indizierte Zugriffe auf Variable.

Programmierung

Syntax:	**EVAL**(exp)	
Beschreibung:	Ausdruck auswerten	
Parameter:	exp	Logischer Ausdruck

Beispiel

```
VAR1=(S)
VAR2=(S)
VAR3=(S)
VAR4=(S)
CHANGE()
  REG[7] = EVAL("VAR"<<REG[5])  ; Der Ausdruck in der Klammer ergibt VAR3, wenn der
                                Wert von REG[5] gleich 3 ist. REG[7] wird somit der
                                Wert von VAR3 zugewiesen.
  IF REG[5] == 1
    REG[7] = VAR1
```

```
  ELSE
    IF REG[5] == 2
      REG[7] = VAR2
    ELSE
      IF REG[5] == 3
        REG[7] = VAR3
      ELSE
        IF REG[5] == 4
          REG[7] = VAR4
        ENDIF
      ENDIF
    ENDIF
  ENDIF
END_CHANGE
```

8.3.16 Exit Loading Softkey (EXITLS)

Beschreibung

Mit der Funktion EXITLS verlassen Sie die aktuelle Bedienoberfläche und laden eine definierte Softkey-Leiste.

Hinweis

Die Funktionen LS, LM, DLGL, EXIT und EXITLS dürfen in den Methoden SUSPEND bzw. RESUME nicht projektiert werden. Bei Verwendung der Funktionen in diesen Methoden wird die Ausführung der Funktionen verhindert.

Programmierung

Syntax:	**EXITLS**("Softkey-Leiste"[, "Pfadname"])	
Beschreibung:	Beim Verlassen Softkey-Leiste laden	
Parameter:	Softkey-Leiste	Name der zu ladenden Softkey-Leiste
	Pfadname	Verzeichnispfad der zu ladenden Softkey-Leiste

Beispiel

```
PRESS(HS1)
    EXITLS( "Leiste1", "AEDITOR.COM" )
END_PRESS
```

8.3.17 Function (FCT)

Beschreibung

Die externen Funktionen werden in einer DLL-Datei hinterlegt und durch einen Eintrag in den Definitionszeilen der Projektierungsdatei bekannt gemacht.

Hinweis

Die externe Funktion muss mindestens einen Rückgabeparameter haben.

Programmierung

Syntax:	**FCT**Funktionsname = ("Datei"/Typ der Rückgabe/Typen der festen Parameter/Typen der variablen Parameter)	
	FCT InitConnection = ("c:\tmp\xyz.dll"/I/R,I,S/I,S)	
Beschreibung:	Der Aufruf einer externen Funktion kann z. B. aus der LOAD-Methode oder in der PRESS-Methode durchgeführt werden.	
Parameter:	Funktionsname	Name der externen Funktion
	Datei	Vollständige Pfadangabe der DLL-Datei
	Typ der Rückgabe	Datentyp des Rückgabewerts
	Typ der festen Parameter	Value Parameter
	Typ der variablen Parameter	Referenzparameter
	Die Datentypen werden durch Komma getrennt.	

Der Aufruf der externen Funktion kann z. B. aus der LOAD-Methode oder in der PRESS-Methode durchgeführt werden.

Beispiel:

```
press(vs4)
RET = InitConnection(VAR1,13,"Servus",VAR2,VAR17)
end_press
```

Struktur der externen Funktion

Die externe Funktion muss eine bestimmte vorgegebene Signatur beachten:

Syntax:	extern "C" dllexport void InitConnection (ExtFctStructPtr FctRet, ExtFctStructPtr FctPar, char cNrFctPar)
Beschreibung:	DLL-Export nur bei Windows Implementierung Spezifizierer und Übergabeparameter sind fest vorgegeben. Durch die übergebenen Strukturen werden die eigentlichen Aufrufparameter weitergereicht.

Parameter:	cNrFctPar	Anzahl der Aufrufparameter = Anzahl der Strukturelemente in FctPar
	FctPar	Zeiger auf ein Feld von Strukturelementen, die die jeweiligen Aufrufparameter mit Datentyp enthalten.
	FctRet	Zeiger auf eine Struktur für die Rückgabe des Funktionswertes mit Datentyp.

Definition der Übergabestruktur

```
union CFI_VARIANT
    (
    char                                b;
    short int                           i;
    double                              r;
    char*                               s;
    )
typedef struct ExtFctStructTag
    (
    char                                cTyp;
    union CFI_VARIANT                   value;
    )ExtFctStruct;
typedef struct ExtFct* ExtFctStructPtr;
```

Soll die externe Funktion unabhängig von der Plattform (Windows, Linux) entwickelt werden, so darf das Schlüsselwort __declspec(dllexport) nicht verwendet werden. Dieses Schlüsselwort ist lediglich unter Windows erforderlich. Unter Qt kann man beispielsweise folgendes Makro benutzen:

```
#ifdef Q_WS_WIN
       #define MY_EXPORT __declspec(dllexport)
#else
       #define MY_EXPORT
#endif
```

Die Deklaration der Funktion lautet wie folgt:

```
extern "C" MY_EXPORT void InitConnection
(ExtFctStructPtr FctRet, ExtFctStructPtr FctPar,char cNrFctPar)
```

Werden die mit "Run MyScreens" projektierten Bilder auf NCU und PCU/PC eingesetzt, so muss die Extension der Binärdatei weggelassen werden:

```
FCT InitConnection = ("xyz"/I/R,I,S/I,S)
```

Bei Weglassen der absoluten Pfadinformation sucht "Run MyScreens" zunächst im projektierten Verzeichnis nach der Binärdatei.

8.3.18 Generate Code (GC)

Beschreibung

Die Funktion GC (Generate Code) generiert NC-Code aus der OUTPUT-Methode.

Programmierung

Syntax:	**GC**("Bezeichner"[,"Zieldatei"][,Opt],[Append])	
Beschreibung:	NC-Code generieren (die Funktion GC ist nur innerhalb der NC möglich)	
Parameter:	Bezeichner	Name der OUTPUT-Methode, der als Grundlage zur Code-Generierung dient
	Zieldatei	Pfadangabe der Zieldatei für HMI- oder NC-Dateisystem. Ist die Zieldatei nicht angegeben (nur innerhalb der Programmierunterstützung möglich), wird der Code an die Stelle geschrieben, an der der Cursor innerhalb der aktuell geöffneten Datei steht.
	Opt	Option zu Kommentargenerierung
	0:	(Voreinstellung) Code mit Kommentar für Rückübersetzbarkeit erzeugen (siehe auch Rückübersetzen (Seite 172)).
	1:	Im generierten Code keine Kommentare erzeugen. **Hinweis:** Dieser Code ist nicht rückübersetzbar (siehe auch Rückübersetzen (Seite 172)).
	Append	Dieser Parameter ist nur von Bedeutung, wenn eine Zieldatei angegeben ist.
	0:	(Voreinstellung) Wenn die Datei bereits existiert, wird der alte Inhalt gelöscht.
	1:	Wenn die Datei bereits existiert, wird der neue Code an den Anfang der Datei geschrieben.
	2:	Wenn die Datei bereits existiert, wird der neue Code an das Ende angehängt.

Beispiel

```
//M(TestGC/"Codegenerierung:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
```

```
END_OUTPUT


PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                ; NC-Code aus der OUTPUT-Methode in die Datei
                                    \MPF.DIR\MESSEN.MPF schreiben:
                                    Cycle123(123, -6)
                                    M30
END_PRESS
```

Ablegen der Zieldatei

Die Funktion GC ist nur innerhalb der NC möglich. Verwenden Sie zum Verschieben der Datei die Funktionen CP bzw. MP. Sehen Sie hierzu die folgenden Beispiele:

```
; Windows directory
  GC("Code","//NC/MPF.DIR/NC1.MPF")
  MP("//NC/MPF.DIR/NC1.MPF","d:/tmp/WIN1.MPF")


; LOCAL_DRIVE
  GC("Code","//NC/MPF.DIR/NC1.MPF")
  MP("//NC/MPF.DIR/NC1.MPF","LOCAL_DRIVE:/WIN_LD1.MPF")
```

Rückübersetzen

- **Keine Angabe der Zieldatei:**
  Die Funktion GC kann nur in der Programmierunterstützung verwendet werden und schreibt den NC-Code in die aktuell im Editor geöffnete Datei. Das Rückübersetzen des NC-Codes ist möglich. Wird die Funktion GC ohne Angabe der Zieldatei unter "Run MyScreens" projektiert, erfolgt bei der Ausführung eine Fehlermeldung.
- **Angabe der Zieldatei:**
  Der generierte Code aus der OUTPUT-Methode wird in die Zieldatei eingetragen. Ist die Zieldatei nicht vorhanden, wird sie im NC-Dateisystem angelegt. Liegt die Zieldatei im HMI-Dateisystem, wird die Datei auf der Festplatte abgelegt. Nutzkommentarzeilen (notwendige Informationen für die Rückübersetzung) werden nicht angelegt, d. h. eine Rückübersetzung ist nicht möglich.

Besonderheiten beim Rückübersetzen

In Unter-Dialogen kann die Funktion GC nicht aufgerufen werden, da in Unter-Dialogen Variablen verwendet werden können, die von Haupt-Dialogen stammen, jedoch bei einem direkten Aufruf nicht vorhanden wären.

Bei manuellen Eingriffen auf den generierten Code mit dem Editor darf die Anzahl Zeichen für Werte, die von der Code-Generierung erzeugt wurden, nicht verändert werden. Eine solche Änderung verhindert die Rückübersetzbarkeit.

Abhilfe:

1. Rückübersetzen
2. Änderung mit Hilfe des projektierten Dialogs eingeben (z. B. 99 → 101)
3. GC

8.3.19 Kennwort-Funktionen

Übersicht

Die folgenden Kennwort-Funktionen stehen zur Verfügung:

- Kennwort setzen
- Kennwort löschen
- Kennwort ändern

Funktion HMI_LOGIN: Neues Kennwort setzen

Syntax:	**HMI_LOGIN**(Passwd)	
Beschreibung:	Mit der Funktion HMI_LOGIN() wird ein Passwort an die NCK gesendet, mit dem die aktuelle Zugriffsstufe eingestellt wird.	
Parameter:	Passwd	Kennwort

Beispiel

```
REG[0] = HMI_LOGIN("CUSTOMER")          ; Ergebnis = TRUE, wenn Kennwort erfolgreich
                                        gesetzt, sonst FALSE
```

Funktion HMI_LOGOFF: Kennwort löschen

Syntax:	**HMI_LOGOFF**
Beschreibung:	Mit der Funktion HMI_LOGOFF kann die aktuelle Zugriffstufe zurückgesetzt werden.
Parameter:	- keine -

Beispiel

```
REG[0] = HMI_LOGOFF                     ; Ergebnis = TRUE, wenn Kennwort erfolgreich
                                        gelöscht, sonst FALSE
```

Funktion HMI_SETPASSWD: Kennwort ändern

Syntax:	**HMI_SETPASSWD**(AC, Passwd)	
Beschreibung:	Mit der Funktion HMI_SETPASSWD kann man die das Kennwort der aktuellen oder einer kleineren Kennwort-Stufe ändern.	
Parameter:	AC	Zugriffsstufe
	Passwd	Kennwort

Beispiel

```
REG[0] = HMI_SETPASSWD(4,"MYPWD")      ; Ergebnis = TRUE, wenn Kennwort erfolgreich
                                       geändert, sonst FALSE
```

8.3.20 Load Array (LA)

Beschreibung

Mit der Funktion LA (Load Array) kann ein Array aus einer anderen Datei geladen werden.

Programmierung

Syntax:	**LA**(Bezeichner [, Datei])	
Beschreibung:	Array aus Datei laden	
Parameter:	Bezeichner	Name des nachzuladenden Arrays
	Datei	Datei, in dem das Array definiert ist

Hinweis

Soll ein Array in der aktuellen Projektierungsdatei durch ein Array aus einer anderen Projektierungsdatei ersetzt werden, müssen die Arrays gleichnamig sein.

Beispiel

```
                                       ; Auszug aus Datei   maske.com
DEF VAR2 = (S/*ARR5/"Aus"/,"Toggle-Feld")
PRESS(HS5)
  LA("ARR5","arrayext.com")            ; Array ARR5 aus Datei arrayext.com laden
  VAR2 = ARR5[0]                       ; Statt "Aus"/"Ein" erscheint im Toggle-
                                       Feld von VAR2
                                       "Oben"/"Unten"/"Rechts"/"Links"
```

```
END_PRESS
//A(ARR5)
("Aus"/"Ein")
//END
                                                    ; Auszug aus Datei arrayext.com
//A(ARR5)
("Oben"/"Unten"/"Rechts"/"Links")
//END
```

Hinweis

Beachten Sie, dass einer Variablen ein gültiger Wert zugewiesen werden muss, nachdem mit der LA-Funktion dem Toggle-Feld der Variablen ein anderes Array zugeordnet wurde.

8.3.21 Load Block (LB)

Beschreibung

Mit der Funktion LB (Load Block) können Blöcke mit Unterprogrammen zur Laufzeit geladen werden. Vorzugsweise sollte LB in einer LOAD-Methode projektiert werden, damit die geladenen Unterprogramme jederzeit aufgerufen werden können.

Hinweis

Unterprogramme können auch direkt in einem Dialog definiert werden; dann müssen sie nicht geladen werden.

Programmierung

Syntax:	**LB**("Blockname"[,"Datei"])	
Beschreibung:	Unterprogramm zur Laufzeit laden	
Parameter:	Blockname	Name der Blockkennung
	Datei	Pfadangabe der Projektierungsdatei Voreinstellung = aktuelle Projektierungsdatei

Beispiel

```
LOAD
```

```
  LB("PROG1")              ; Block "PROG1" wird in der aktuellen Projektierungsdatei
                           gesucht und anschließend geladen.
  LB("PROG2","XY.COM")     ; Block "PROG2" wird in der Projektierungsdatei XY.COM ge-
                           sucht und anschließend geladen.
END_LOAD
```

8.3.22 Load Mask (LM)

Beschreibung

Mit der Funktion LM wird ein neuer Dialog geladen. Abhängig vom Modus des Dialogwechsels (siehe Tabelle unten) kann mit dieser Funktion auch ein Message-Dialog realisiert werden.

Hinweis

Die Funktionen LS, LM, DLGL, EXIT und EXITLS dürfen in den Methoden SUSPEND bzw. RESUME nicht projektiert werden. Bei Verwendung der Funktionen in diesen Methoden wird die Ausführung der Funktionen verhindert.

Haupt-Dialog / Unter-Dialog

Ein Dialog, der einen anderen Dialog aufruft und selbst nicht beendet wird, wird als Haupt-Dialog bezeichnet. Ein Dialog, der von einem Haupt-Dialog aus aufgerufen wird, wird als Unter-Dialog bezeichnet.

Programmierung

Syntax:	**LM**("Bezeichner"[,"Datei"] [,MSx [, VARx]])
Beschreibung:	Dialog laden

Parameter:	Bezeichner	Name des zu ladenden Dialogs
	Datei	Pfadangabe (HMI-Dateisystem oder NC-Dateisystem) der Projektierungsdatei; Standardeinstellung: aktuelle Projektierungsdate
	MSx	Modus des Dialogwechsels
	0:	(Voreinstellung) Der aktuelle Dialog wird verworfen, der neue Dialog wird geladen und angezeigt. Bei EXIT kehren Sie zur Standardapplikation zurück. Über den Parameter MSx kann bestimmt werden, ob beim Dialogwechsel der aktuelle Dialog beendet wird, oder nicht. Bleibt der aktuelle Dialog bestehen, können Variablen in den neuen Dialog übergeben werden. Der Vorteil des Parameters MSx besteht darin, dass die Dialoge beim Wechsel nicht immer neu initialisiert werden müssen, sondern Daten und Layout des aktuellen Dialogs erhalten bleiben und die Datenübergabe erleichtert wird
	1:	Der aktuelle Haupt-Dialog wird ab der Funktion LM unterbrochen, der neue Unter-Dialog wird geladen und angezeigt (z. B. für die Realisierung eines Message-Dialogs). Bei EXIT wird der Unter-Dialog beendet und es wird zur Unterbrechungsstelle des Haupt-Dialogs zurückgekehrt. Im Haupt-Dialog wird bei der Unterbrechung der UNLOAD-Block nicht abgearbeitet.
	VARx	Voraussetzung: MS1 Auflistung von Variablen, die vom Haupt-Dialog an den Unter-Dialog übergeben werden können. Es können bis zu 20 Variablen, jeweils durch Komma getrennt, übergeben werden.

Hinweis

Der Parameter VARx übergibt jeweils nur den Wert der Variablen, d.h. die Variablen können im Unter-Dialog gelesen und geschrieben werden, sind aber dort nicht sichtbar. Die Rückgabe der Variable vom Unter-Dialog zum Haupt-Dialog ist über die Funktion EXIT möglich.

Beispiel

MASKE1(Haupt-Dialog): Aussprung in den Unter-Dialog MASKE2 mit Variablenübergabe

```
PRESS(HS1)

  LM("MASKE2","CFI.COM",1, POSX, POSY, DURCHMESSER)
                                    ; Maske1 unterbrechen und Maske2 aufblenden:
                                    Dabei werden die Variablen POSX, POSY und
                                    DURCHMESSER übergeben.
  DLGL("Maske2 beendet")            ; Nach der Rückkehr aus Maske2 erscheint in der
                                    Dialogzeile von Maske1 der Text: Maske2 beendet.

END_PRESS
```

```
MASKE2(Unter-Dialog): Rücksprung nach Haupt-Dialog MASKE1 mit Übergabe der Variable-
ninhalte
PRESS(VS8)

  EXIT(POSX, POSY, DURCHMESSER)
                                    ; Maske2 abblenden und Maske1 fortsetzen: Da-
                                    bei werden die geänderten Variablen POSX, POSY
                                    und DURCHMESSER übergeben.
END_PRESS
```

8.3.23 Load Softkey (LS)

Beschreibung

Mit der Funktion LS kann eine andere Softkey-Leiste eingeblendet werden.

Hinweis

Die Funktionen LS, LM, DLGL, EXIT und EXITLS dürfen in den Methoden SUSPEND bzw. RESUME nicht projektiert werden. Bei Verwendung der Funktionen in diesen Methoden wird die Ausführung der Funktionen verhindert.

Programmierung

Syntax:	**LS**("Bezeichner"[, "Datei"][, Merge])	
Beschreibung:	Softkey-Leiste einblenden	
Parameter:	Bezeichner	Name der Softkey-Leiste
	Datei	Pfadangabe (HMI-Dateisystem oder NC-Dateisystem) der Projektierungsdatei Voreinstellung: aktuelle Projektierungsdatei
	Merge	
	0:	Alle bestehenden Softkeys werden gelöscht, die neu projektierten Softkeys werden eingetragen.
	1:	Voreinstellung Nur die neu projektierten Softkeys überschreiben die vorhandenen Softkeys, die anderen Softkeys (= Softkeys der HMI-Applikation) bleiben mit ihrer Funktionalität und Text erhalten.

Beispiel

```
PRESS(HS4)
```

```
  LS("Leiste2",,0)    ; Leiste2 überschreibt die vorhandene Softkey-Leiste, die ein-
                      geblendeten Softkeys werden gelöscht.
END_PRESS
```

Hinweis

Solange der Interpreter noch keinen Dialog aufgeblendet hat, d.h. noch keine LM-Funktion abgearbeitet wurde, kann in der Methode PRESS des Beschreibungsblocks des Einstiegssoftkeys und der Softkey-Leiste nur jeweils ein LS- oder LM-Kommando projektiert werden und keine andere Aktion.

Die Funktionen LS und LM dürfen ausschließlich innerhalb einer Softkey-PRESS-Methode aufgerufen werden, nicht jedoch als Reaktion auf Navigationstasten (PU, PD, SL, SR, SU, SD).

8.3.24 Load Grid (LG)

Beschreibung

Die Tabellenbeschreibung (Grid) kann dynamisch innerhalb von Methoden (z. B. LOAD) mittels der LG-Methode bereitgestellt werden.

Damit mit der LG-Methode eine Tabelle zugewiesen werden kann, muss die Variable bereits als Grid-Variable definiert sein und auf eine gültige vorhandene Tabelle verweisen.

Programmierung

Syntax:	**LG**(Gridname, Variablenname [,Dateiname])	
Beschreibung:	Tabelle laden	
Parameter:	Gridname	Name der Tabelle (Grid) in Hochkommata
	Variablenname	Name der Variable, der die Tabelle zugewiesen werden soll, in Hochkommata
	Dateiname	Name der Datei, in der die Tabelle (Grid) definiert ist, in Hochkommata. Muss nur angegeben werden, wenn die Tabelle nicht innerhalb der Datei definiert ist, in der auch die Variable definiert ist

Beispiel

```
//M(MyGridSample/"My Grid Sample")
DEF MyGridVar=(R/% MyGrid1///WR2////100,,351,100)
LOAD
   LG("MyGrid1","MyGridVar","mygrids.com")
END_LOAD
```

```
Inhalt von mygrids.com:
//G(MyGrid1/0/5)
(I///,"MyGrid1"/wr1//"1"/80/1)
(R3///"LongText1","R1-R4"/wr2//"$R[1]"/80/1)
(IBB///"LongText2","M2.2-M2.5"/wr2//"M2.2"/80/,1)
(R3///"LongText3","R9,R11,R13,R15"/wr2//"$R[9]"/110/2)
//END
```

8.3.25 Mehrfachauswahl SWITCH

Beschreibung

Mit einem SWITCH-Kommando können Sie eine Variable auf verschiedene Werte prüfen.

Der in der SWITCH-Anweisung projektierte Ausdruck, wird nacheinander mit allen in den nachfolgenden CASE-Anweisungen projektierten Werten verglichen.

Bei Ungleichheit wird ein Vergleich mit der jeweils nachfolgenden CASE-Anweisung durchgeführt. Folgt eine DEFAULT-Anweisung und war bisher keine CASE-Anweisung gleich dem in der SWITCH-Anweisung projektierten Ausdruckes, so werden die der DEFAULT-Anweisung folgenden Anweisungen abgearbeitet.

Bei Gleichheit werden die nachfolgenden Anweisungen solange abgearbeitet, bis entweder ein CASE-, DEFAULT- oder END_SWITCH-Anweisung folgt.

Beispiel

```
FOCUS
   SWITCH (FOC)
      CASE "VarF"
         DLGL("Variable ""VarF"" has the input focus.")
      CASE "VarZ"
         DLGL("Variable ""VarZ"" has the input focus.")
      DEFAULT
         DLGL("Any other variable has the input focus.")
   END_SWITCH
END_FOCUS
```

8.3.26 Multiple Read NC PLC (MRNP)

Beschreibung

Mit diesem Kommando MRNP können mehrere NC-/PLC-Variablen mit einem Zugriff in Register übertragen werden. Dieser Zugriff ist deutlich schneller als das Lesen über Einzelzugriffe. Die NC-/PLC-Variablen müssen innerhalb eines MRNP Kommandos aus demselben Bereich sein.

Bereiche der NC-/PLC-Variablen sind folgendermaßen gegliedert:

- Allgemeine NC-Daten ($MN..., $SN.., /nck/...)
- Kanalspezifische NC-Daten ($MC..., $SC.., /channel/...)
- PLC-Daten (DB..., MB.., /plc/...)
- Achsspezifische NC-Daten der gleichen Achse ($MA..., $SA..)

Programmierung

Syntax:	**MRNP**(Variablenname1*Variablenname2[* ...], Registerindex)
Beschreibung:	mehrere Variablen lesen
Parameter:	Bei den Variablennamen gilt "*" als Trennzeichen. In der Reihenfolge wie die Variablennamen im Kommando stehen, werden die Werte in die Register REG[Registerindex] und folgende übernommen. In diesem Zusammenhang gilt: Der Wert der ersten Variable steht in REG[Registerindex]. Der Wert der zweiten Variable steht in REG[Registerindex + 1] usw.

Hinweis

Es ist darauf zu achten, dass die Liste der Variablen auf 500 Zeichen und die Anzahl der Register begrenzt ist.

Beispiel

```
MRNP("$R[0]*$R[1]*$R[2]*$R[3]",1)      ;REG[1] bis REG[4] wird mit dem Wert der Va-
                                        riablen $R[0] bis $R[3] beschrieben.
```

NC-Variable

Es stehen alle Maschinen- und Setting-Daten sowie R-Parameter, aber nur bestimmte Systemvariablen zur Verfügung (siehe auch: AUTOHOTSPOT).

Es sind alle globalen und kanalspezifischen Anwendervariablen (GUDs) zugänglich. Die lokalen und programmglobalen Anwendervariablen können nicht verarbeitet werden.

Maschinendaten	
Globales Maschinendatum	$MN_...
Achsspezifisches Maschinendatum	$MA_...
Kanalspezifisches Maschinendatum	$MC_...

Setting-Daten	
Globales Settingdatum	$SN_...
Achsspezifisches Settingdatum	$SA_...
Kanalspezifisches Settingdatum	$SC_...

Systemvariablen	
R-Parameter 1	$R[1]

PLC-Variable

Es stehen alle PLC-Daten zur Verfügung.

PLC-Daten	
Byte y Bit z von Datenbaustein x	DBx.DBXy.z
Byte y von Datenbaustein x	DBx.DBBy
Word y von Datenbaustein x	DBx.DBWy
Doubleword y v. Datenbaustein x	DBx.DBDy
Real y von Datenbaustein x	DBx.DBRy
Merkerbyte x Bit y	Mx.y
Merkerbyte x	MBx
Merkerword x	MWx
Merkerdoubleword x	MDx
Eingangsbyte x Bit y	Ix.y oder Ex.y
Eingangsbyte x	IBx oder EBx
Eingangsword x	IWx oder EWx
Eingangsdoubleword x	IDx oder EDx
Ausgangsbyte x Bit y	Qx.y oder Ax.y
Ausgangsbyte x	QBx oder ABx
Ausgangsword x	QWx oder AWx
Ausgangsdoubleword x	QDx oder ADx
String y mit Länge z aus Datenbaustein x	DBx.DBSy.z

8.3.27 P_LOGICAL_FILEPATH

Beschreibung

Die Funktion P_LOGICAL_FILEPATH gibt den Namen und den logischen Pfad des aktuell bearbeiteten Teilprogramms im Editor zurück.

Hinweis

Die Funktion ist nur im Kontext von Zyklusmasken verfügbar.

Programmierung

Syntax:	P_LOGICAL_FILEPATH()	
Beschreibung:	Name und logischen Pfad des aktuell bearbeiteten Teilprogramms im Editor ermitteln z. B. "//NC/MPF.DIR/HUGO.MPF"	
Parameter:	- keine -	

Siehe auch

P_REAL_FILEPATH (Seite 166)

8.3.28 P_REAL_FILEPATH

Beschreibung

Die Funktion P_REAL_FILEPATH gibt den Namen und den realen Pfad des aktuell bearbeiteten Teilprogramms im Editor zurück.

Hinweis

Die Funktion ist nur im Kontext von Zyklusmasken verfügbar.

Programmierung

Syntax:	P_REAL_FILEPATH()	
Beschreibung:	Name und realer Pfad des aktuell bearbeiteten Teilprogramms im Editor ermitteln z. B. "/_N_MPF_DIR/_N_HUGO_MPF"	
Parameter:	- keine -	

Siehe auch

P_LOGICAL_FILEPATH (Seite 166)

8.3.29 PI-Dienste

Beschreibung

Mit der Funktion PI_START können PI-Dienste (Programm Instanz Dienste) von der PLC im NC-Bereich gestartet werden.

Hinweis

Eine Liste der verfügbaren PI-Dienste finden Sie im Funktionshandbuch Grundfunktionen.

Programmierung

Syntax:	**PI_START**("Übergabestring")	
Beschreibung:	PI-Dienst ausführen	
Parameter:	"Übergabestring"	Der Übergabestring ist im Gegensatz zur OEM-Dokumentation in doppelte Hochkommas zu setzen.

Beispiel

```
PI_START("/NC,001,_N_LOGOUT")
```

Hinweis

Kanalabhängige PI-Dienste beziehen sich immer auf den aktuellen Kanal.

PI-Dienste der Werkzeugfunktionen (TO-Bereich), beziehen sich immer auf den TO-Bereich, der dem aktuellen Kanal zugeordnet wird.

8.3.30 System- oder Anwendervariable lesen (RNP) und schreiben (WNP)

Beschreibung

Mit dem Befehl RNP (Read NC PLC) können NC- oder PLC-Variable oder Maschinendaten gelesen werden.

Programmierung

Syntax:	**RNP** ("System- oder Anwendervariable", Wert)	
Beschreibung:	NC- oder PLC-Variable oder Maschinendaten lesen	
Parameter:	System- oder Anwendervariable	Name der NC- oder PLC-Variablen
	Wert	Wert, der in die System- oder Anwendervariable geschrieben werden soll. Ist der Wert vom Typ String, muss er in Doppelhochkomma geschrieben werden.

Beispiel

```
VAR2=RNP("$AA_IN[2]")         ; NC-Variable lesen
```

Beschreibung

Mit dem Befehl WNP (Write NC PLC) können NC- oder PLC-Variable oder Maschinendaten geschrieben werden.

Zugriffe auf NC-/PLC-Variable werden bei jeder Bearbeitung der Funktion WNP neu ausgeführt, d.h. dass ein NC-/PLC-Zugriff in einer CHANGE-Methode immer ausgeführt wird. Dies ist sinnvoll, wenn eine System- oder Anwendervariable häufig den Wert ändert. Soll ein NC-/PLC-Zugriff nur einmalig ausgeführt werden, muss dies in einer LOAD- oder UNLOAD-Methode projektiert werden.

Programmierung

Syntax:	**WNP**("System- oder Anwendervariable", Wert)	
Beschreibung:	NC- oder PLC-Variable oder Maschinendaten schreiben	
Parameter:	System- oder Anwendervariable	Name der NC- oder PLC-Variablen
	Wert	Wert, der in die System- oder Anwendervariable geschrieben werden soll. Ist der Wert vom Typ String, muss er in Doppelhochkomma geschrieben werden.

Beispiel

```
WNP("DB20.DBB1",1)        ; PLC-Variable schreiben
```

8.3.31 RESIZE_VAR_IO und RESIZE_VAR_TXT

Beschreibung

Mit den Kommandos `RESIZE_VAR_IO()` und `RESIZE_VAR_TXT()` kann die Geometrie des Ein-/Ausgabe- oder des Text-Anteiles einer Variablen verändert werden.

Nach dem Setzen der neuen Geometrie, werden alle Felder der Maske so positioniert, als ob die Maske von vornherein mit diesen Positionen projektiert worden wäre. Somit werden alle Felder, abhängig von der Projektierung, zueinander korrekt ausgerichtet.

Programmierung

Syntax:	**RESIZE_VAR_IO**(Variablenname, [X], [Y], [Breite], [Höhe]) **RESIZE_VAR_TXT**(Variablenname, [X], [Y], [Breite], [Höhe])	
Beschreibung:	Verändern der Geometrie des Ein-/Ausgabe- oder des Text-Anteiles einer Variable.	
Parameter:	Variablenname	Name der Variable, deren Geometrie des Ein-/Ausgabe- oder des Text-Anteiles geändert werden soll.
	X	X-Koordinate Links-Oben
	Y	Y-Koordinate Lnks-Oben
	Breite	Breite
	Höhe	Höhe

Hinweis

Für jeden nicht angegebenen Wert von X, Y, Breite oder Höhe wird der bisherige Wert beibehalten. Gibt man -1 für einen dieser Werte an, dann wird der von "Run MyScreens" vorgegebene Anteil der Standard-Position gesetzt.

Beispiel

```
RESIZE_VAR_IO("MyVar1", 200, , 100)    ; Der IO-Anteil der Variable "MyVar1" wird an
                                       die X-Position 200 Pixel verschoben, die Brei-
                                       te wird auf 100 Pixel gesetzt. Die Y-Position
                                       und die Höhe des vorherigen Wertes werden bei-
                                       behalten.
```

8.3.32 Register (REG)

Beschreibung Register

Register werden benötigt, um Daten zwischen verschiedenen Dialogen auszutauschen. Register sind jeweils einem Dialog zugeordnet und werden beim Laden des ersten Dialogs erzeugt und mit 0 oder Leer-String belegt.

Hinweis

Register dürfen nicht direkt in einer OUTPUT-Methode zur NC-Code Generierung verwendet werden.

Programmierung

Syntax:	**REG**[x]	
Beschreibung:	Register definieren	
Parameter:	x	Register-Index mit x = 0...19; Typ: REAL oder STRING = VARIANT Die Register mit x ≥ 20 sind bereits von Siemens belegt.

Beschreibung Registerwert

Die Zuordnung von Werten zu Registern wird in einer Methode projektiert.

Hinweis

Wird von einem Dialog ein weiterer Dialog mit der Funktion LM erzeugt, wird der Inhalt von Registern automatisch in den neuen Dialog übernommen und steht für weitere Berechnungen im zweiten Dialog zur Verfügung.

Programmierung

Syntax:	Bezeichner.val = Registerwert oder Bezeichner = Registerwert	
Beschreibung:		
Parameter:	Bezeichner	Name des Registers
	Registerwert	Wert des Registers

Beispiel

```
UNLOAD
```

```
  REG[0] = VAR1          ; Register 0 den Wert der Variablen1 zuweisen
END_UNLOAD

UNLOAD
  REG[9].VAL = 84        ; Register 9 den Wert 84 zuweisen
END_UNLOAD
                         ; Im nachfolgenden Dialog können diese Register dann in ei-
                         ner Methode wieder lokalen Variablen zugeordnet werden.
LOAD
  VAR2 = REG[0]
END_LOAD
```

Beschreibung Register-Zustand

Mit der Eigenschaft Zustand kann in der Projektierung abgefragt werden, ob ein Register einen gültigen Wert enthält.

Die Abfrage des Register-Zustands kann genutzt werden, um in ein Register nur dann einen Wert zu schreiben, wenn ein Dialog als Haupt-Dialog verwendet wird.

Programmierung

Syntax:	Bezeichner.vld	
Beschreibung:	Die Eigenschaft ist nur lesbar.	
Parameter:	Bezeichner	Name des Registers
Rückgabewert:		Das Ergebnis der Abfrage kann sein:
	FALSE =	ungültiger Wert
	TRUE =	gültiger Wert

Beispiel

```
IF REG[15].VLD == FALSE    ; Gültigkeit des Registerwertes abfragen
  REG[15] = 84
ENDIF
VAR1 = REG[9].VLD          ; Var1 den Wert der Zustandsabfrage von REG[9] zuweisen.
```

8.3.33 RETURN

Beschreibung

Mit der Funktion RETURN kann die Abarbeitung des aktuellen Unterprogramms vorzeitig abgebrochen und zur Absprungstelle des letzten CALL-Befehls zurückgekehrt werden.

Wird im Unterprogramm kein RETURN projektiert, wird das Unterprogramm bis zum Ende ausgeführt und dann zur Absprungstelle zurückgekehrt.

Programmierung

Syntax:	**RETURN**	
Beschreibung:	zur Absprungstelle zurückkehren	
Parameter:	- keine -	

Beispiel

```
//B(PROG1)                ; Blockanfang
SUB(UP2)                  ; Unterprogramm Anfang
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN                ; Falls der Variablenwert = Otto ist, wird der Variablen
                          der Wert "Hans" zugewiesen und das Unterprogramm an dieser
                          Stelle beendet.
  ENDIF
  VAR1.val="Otto"         ; Falls der Variablenwert ≠ Otto ist, wird der Variablen
                          der Wert "Otto" zugewiesen.
END_SUB                   ; Unterprogramm Ende
//END                     ; Blockende
```

8.3.34 Rückübersetzen

Beschreibung

In der Programmierunterstützung ist es möglich, den mit der Funktion GC erzeugten NC-Code **zurückzuübersetzen** und die Variablenwerte wieder im Ein-/Ausgabefeld des zugehörigen Eingabedialogs anzuzeigen.

Programmierung

Variable aus dem NC-Code werden in den Dialog übernommen. Dabei werden die Variablenwerte aus dem NC-Code mit den berechneten Variablenwerten aus der Projektierungsdatei verglichen. Ist keine Übereinstimmung vorhanden, wird eine Fehlermeldung im Logbuch ausgeben, da Werte im generierten NC-Code geändert wurden.

Ist eine Variable mehrmals im NC-Code vorhanden, wird beim Rückübersetzen immer das letzte Vorkommen dieser Variable ausgewertet. Zusätzlich wird eine Warnung im Logbuch ausgegeben.

Variablen, die bei der Code-Generierung nicht im NC-Code verwendet werden, sind als Nutzkommentar gespeichert. Mit Nutzkommentar werden alle zur Rückübersetzung notwendigen Informationen bezeichnet. Nutzkommentar darf nicht geändert werden.

Hinweis

Der Block aus NC-Code und Nutzkommentar kann nur rückübersetzt werden, wenn er am Anfang einer Zeile beginnt.

Beispiele:

Im Programm steht folgender NC-Code:

```
DEF VAR1 = (I//101)
OUTPUT(CODE1)
  "X" VAR1 " Y200"
  "X" VAR1 " Y0"
END_OUTPUT
```

Im Teileprogramm wird folgender Code abgelegt:

```
;NCG#TestGC#\cus.dir\aeditor.com#CODE1#1#3#
X101 Y200
X101 Y0
;#END#
```

Der Editor liest beim Rückübersetzen:

```
X101 Y200
X222 Y0          ; Der Wert für X wurde im Teileprogramm geändert (X101 → X222)
```

Im Eingabedialog wird folgender Wert für VAR1 vorgegeben: `VAR1 = 222`

Siehe auch

Generate Code (GC) (Seite 154)

8.3.35 Rückübersetzen ohne Nutzkommentar

Beschreibung

In der Programmierunterstützung ist es möglich, den mit der Funktion GC erzeugten NC-Code **ohne Nutzkommentare zurückzuübersetzen** und die Variablenwerte wieder im Ein-/Ausgabefeld des zugehörigen Eingabedialogs anzuzeigen.

Programmierung

Um die beim regulären Code-Generieren entstehenden Kommentarzeilen zu unterdrücken, kann der GC-Befehl auf folgende Weise ausgeführt werden:

```
GC("CODE1",D_NAME,1)
```

Der entstehende Code ist regulär nicht rückübersetzbar. Um so erzeugte Zyklenaufrufe trotzdem rückübersetzen zu können, sind folgende Schritte notwendig:

- **Datei "easyscreen.ini" erweitern**
  In der Datei "easyscreen.ini" wird die Sektion `[RECOMPILE_INFO_FILES]` eingeführt. In dieser Sektion werden alle ini-Dateien aufgelistet, die Beschreibungen für ohne Nutzkommentar rückzuübersetzende Zyklen enthalten:
  ```
  [RECOMPILE_INFO_FILES]
  IniFile01 = cycles1.ini
  IniFile02 = cycles2.ini
  ```
  Es können mehrere ini-Dateien angegeben werden, deren Namen jeweils frei wählbar sind.
- **Ini-Datei für Zyklenbeschreibung anlegen**
  Legen Sie die ini-Datei mit den Zyklenbeschreibungen unter folgendem Pfad ab:
  [System user-Verzeichnis]/cfg
  [System oem-Verzeichnis]/cfg
  [System addon-Verzeichnis]/cfg
  Für jeden Zyklus ist ein eigener Abschnitt nötig. Der Abschnittsname entspricht dem Namen des Zyklus:
  ```
  [Cycle123]
  Mname = TestGC
  Dname = testgc.com
  OUTPUT = Code1
  Anzp = 3
  Version = 0
  Code_typ = 1
  Icon = cycle123.png
  Desc_Text = This is describing text
  ```

Mname	Maskenname
Dname	Name der Datei, in der die Maske definiert ist
OUTPUT	Name der betreffenden Output-Methode
Anzp	Anzahl der Parameter der rückzuübersetzenden Maske (alle mit DEF angelegten Variablen, auch Hilfsvariablen)
Version	(optional) Versionsangabe für Zyklus

Icon	(optional) Icon für Anzeige im Schrittkettenprogramm, Format *.png Bildgröße für entsprechende Auflösung: 640 x 480 mm → 16 x 16 Pixel 800 x 600 mm → 20 x 20 Pixel 1024 x 768 mm → 26 x 26 Pixel 1280 x 1024 mm → 26 x 26 Pixel 1280 x 768 mm → 26 x 26 Pixel Ablage: [System user-Verzeichnis]/ico/ico<Auflösung> **Hinweise:** • Bei 1280-Auflösungen wird der Ordner für 1024 x 768 mm verwendet (nur für Schrittkettenprogramme geeignet). • Die Bildgröße hängt nicht nur von der Auflösung ab, sondern auch von der im Editor eingestellen Schriftgröße.
Desc_Text	(optional) Erklärungstext für Anzeige im Schrittkettenprogramm, max. 17 Zeichen Stringlänge (nur für Schrittkettenprogramme geeignet)
Param_Text	(optional) Text für die Parameterliste T=%1 D=%2

Hinweis

Hinweise zu Icon, Desc_Text und Param_Text:

1. Das Anwender-Icon wird im G-Code-Editor immer angezeigt. Im ShopMill-/ShopTurn-Editor wird immer das G-Code-Icon angezeigt. Dies dient zur besseren Unterscheidung von ShopMill-/ShopTurn-Schritten und den NICHT-ShopMill-/ShopTurn-Schritten.
2. Der Run MyScreens-Schritt ist abhängig von der Editor-Einstellung "Zyklen als Arbeitsschritt darstellen". Das gilt für den G-Code- sowie für den JobShop-Editor.
   Das heißt, bei "Zyklen als Arbeitsschritt darstellen" = "JA" wird der "Desc_Text" der Run MyScreens-Projektierung genommen, bei "NEIN" wird der Zyklus dargestellt.
3. Desc_Text und Param_Text
   – können über die $xxxxx-Notation sprachabängig angegeben werden
   – über %1, %2, ... kann auf die Parameter des generierten Zyklen-Aufrufes zugegriffen werden. Dabei wird der Platzhalter durch den Parameter ersetzt, der sich in der generierten Parameterliste an der Position befindet, die hinter dem '%' angegeben ist.

Beispiel

```
//M(TestGC/"Codegenerierung:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
LOAD
  VAR1 = 123
  VAR2 = -6
```

```
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                          ;NC-Code aus der OUTPUT-Methode in die
                                              Datei \MPF.DIR\MESSEN.MPF schreiben:
                                              Cycle123(123, -6)
                                              M30
END_PRESS
```

Randbedingungen

- Die Funktionalität "Rückübersetzung ohne Nutzkommentar" hat nicht den vollen Funktionsumfang wie "Rückübersetzen mit Nutzkommentar".
  Beim "Rückübersetzen ohne Nutzkommentar" werden typische Zyklenaufrufe wie z. B. MYCYCLE(PAR1, PAR2, PAR3, ...) unterstützt. Innerhalb der Zeile des Funktionsaufrufes darf jedoch kein Nutzkommentar stehen. Optionale Parameter, die nicht beim Funktionsaufruf übergeben werden und vom Typ String S sind, müssen jedoch mindestens mit leeren Anführungsstrichen z. B. "" angegeben werden. "Run MyScreens" versucht sonst diese Parameter durch Kommas aufzufüllen, um anschließend den so "aufgefüllten Zyklenaufruf" rückzuübersetzen.
- Parameter vom Typ String dürfen innerhalb des zu übergebenden Strings kein Komma und Semikolon beinhalten.
- Bei "Rückübersetzen ohne Nutzkommentar" müssen alle im OUTPUT-Methode enthaltenen Variablen immer innerhalb der Klammern sein, um die Funktionalität "fehlende Zyklenparameter mit Kommas auffüllen" wirken lassen zu können.

Beispiel:

Erlaubt:

```
OUTPUT
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")"
END_OUTPUT
```

Nicht erlaubt (die Variable MYCOMMENT ist nach der schließenden Klammer positioniert.):

```
OUTPUT
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")" MYCOMMENT
END_OUTPUT
```

8.3.36 Search Forward, Search Backward (SF, SB)

Beschreibung

Mit der Funktion **SF, SB (Search Forward, Search Backward)** kann im aktuellen NC-Programm des Editors von der aktuellen Cursorposition nach einem String gesucht und dessen Wert ausgegeben werden.

Programmierung

Syntax:	**SF**("String")	
Bezeichnung:	**S**earch **F**orward: von der aktuellen Cursor Position vorwärts suchen	
Syntax:	**SB**("String")	
Bezeichnung:	**S**earch **B**ackward: von der aktuellen Cursor Position rückwärts suchen	
Parameter:	String	zu suchender Text

Regeln bei der Suche

- Vor und nach der Einheit aus zu suchenden String und dessen Wert muss im aktuellen NC-Programm ein Leerzeichen stehen.
- Der Suchbegriff wird nicht in Kommentaren und nicht innerhalb eines Strings gesucht.
- Der auszugebende Wert muss ein numerischer Ausdruck sein, Ausdrücke der Form "X1=4+5" werden nicht erkannt.
- Hexadezimalkonstanten der Form X1='HFFFF', Binärkonstanten der Form X1='B10010' und Exponentialkonstanten der Form X1='-.5EX-4' werden erkannt.
- Der Wert eines Strings kann ausgegeben werden, wenn zwischen String und Wert folgendes steht:
  - nichts
  - Leerzeichen
  - Gleichheitszeichen

Beispiel

Folgende Schreibweisen sind möglich:

```
X100 Y200               ; Die Variable Abc erhält den Wert 200
Abc = SB("Y")
X100 Y 200              ; Die Variable Abc erhält den Wert 200
Abc = SB("Y")
X100 Y=200              ; Die Variable Abc erhält den Wert 200
Abc = SB("Y")
```

8.3.37 SL_HMI_INSTALL_DIR

Beschreibung

Die Funktion SL_HMI_INSTALL_DIR gibt den Pfad des Installationsverzeichnisses von SINUMERIK Operate zurück.

Programmierung

Syntax:	SL_HMI_INSTALL_DIR()	
Beschreibung:	Installationsverzeichnis von SINUMERIK Operate ermitteln	
Parameter:	- keine -	

8.3.38 SL_TEMP_DIR

Beschreibung

Die Funktion SL_TEMP_DIR gibt einen Pfad für temporäre Dateien in SINUMERIK Operate zurück.

Hinweis

Der Inhalt dieses Verzeichnis ist nicht persistent, d. h. es wird mit jedem Start von SINUMERIK Operate gelöscht.

Programmierung

Syntax:	SL_TEMP_DIR()	
Beschreibung:	Pfad des nicht persistenten (flüchtigen) Verzeichnisses von SINUMERIK Operate ermitteln	
Parameter:	- keine -	

8.3.39 STRING-Funktionen

Übersicht

Die folgenden Funktionen erlauben die Bearbeitung von Strings:

- Ermitteln der Länge von Strings
- Suchen eines Zeichens in einem String
- Stringteil von links extrahieren
- Stringteil von rechts extrahieren

- Stringteil aus der Stringmitte extrahieren
- Ersetzen von Stringteilen
- Vergleichen von Strings
- Einfügen eines Strings in einen String
- Entfernen eines Strings aus einem String
- Entfernen von Leerzeichen (von links bzw. von rechts)
- Einfügen von Werten oder Strings mit Formatierungskennungen

Funktion LEN: Länge eines Strings

Syntax:	**LEN**(string | varname)	
Beschreibung:	Anzahl der Zeichen eines Strings bestimmen	
Parameter:	string	Jeder gültige String Ausdruck. Bei einem Leerstring wird NULL zurückgegeben.
	varname	Jeder gültige und deklarierte Variablenname
	Nur einer der beiden möglichen Parameter ist zulässig.	

Beispiel

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HALLO"
  VAR02=LEN(VAR01)          ; Ergebnis = 5
END_LOAD
```

Funktion INSTR: Zeichen in Zeichenkette suchen

Syntax:	**INSTR**(Start, String1, String2 [,Richtung])	
Beschreibung:	Zeichen suchen	
Parameter:	Start	Startposition, von der aus string1 in string2 gesucht wird. Wenn die suche am Anfang von string2 beginnen soll, ist 0 anzugeben.
	String1	Zeichen, welches gesucht wird.
	String2	Zeichenkette, in der gesucht wird
	Richtung (optional)	Richtung, in der gesucht wird 0: von links nach rechts (Voreinstellung) 1: von rechts nach links
	Ist string1 nicht in string2 enthalten, so wird eine 0 zurückgeliefert.	

Beispiel

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HALLO/WELT"
  VAR02=INST(1,"/",VAR01)        ; Ergebnis = 6
END_LOAD
```

Funktion LEFT: String von links

Syntax:	**LEFT**(string, länge)	
Beschreibung:	LEFT liefert eine Zeichenkette zurück, welche die angegebene Anzahl Zeichen von der linken Seite eines Strings enthält.	
Parameter:	string	Zeichenkette oder Variable mit der zu verarbeitenden Zeichenkette
	länge	Anzahl der Zeichen, welche ausgelesen werden sollen

Beispiel

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HALLO/WELT"
  VAR02=LEFT(VAR01,5)           ; Ergebnis = "HALLO"
END_LOAD
```

Funktion RIGHT: String von rechts

Syntax:	**RIGHT**(string, länge)	
Beschreibung:	RIGHT liefert eine Zeichenkette zurück, welche die angegebene Anzahl Zeichen von der rechten Seite eines Strings enthält.	
Parameter:	string	Zeichenkette oder Variable mit der zu verarbeitenden Zeichenkette
	länge	Anzahl der Zeichen, welche ausgelesen werden sollen

Beispiel

```
DEF VAR01
DEF VAR02
```

```
LOAD
  VAR01="HALLO/WELT"
  VAR02=LEFT(VAR01,4)         ; Ergebnis = "WELT"
END_LOAD
```

Funktion MIDS: String aus der Mitte

Syntax:	**MIDS**(string, start [, länge])	
Beschreibung:	MIDS liefert eine Zeichenkette zurück, welche die angegebene Anzahl Zeichen ab der angegeben Stelle eines Strings enthält.	
Parameter:	string	Zeichenkette oder Variable mit der zu verarbeitenden Zeichenkette
	start	Start, ab dem in der Zeichenkette gelesen wird
	länge	Anzahl der Zeichen, welche ausgelesen werden sollen

Beispiel

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HALLO/WELT"
  VAR02=LEFT(VAR01,4,4)           ; Ergebnis = "LO/W"
END_LOAD
```

Funktion REPLACE: Ersetzen von Zeichen

Syntax:	**REPLACE**(string, FindString, ReplaceString [, start [, count]])	
Beschreibung:	Die Funktion REPLACE ersetzt ein Zeichen/Zeichenkette in einem String durch ein anderes Zeichen/Zeichenkette.	
Parameter:	string	String , in dem der FindString durch den ReplaceString ersetzt werden soll.
	FindString	Zu ersetzender String
	ReplaceString	Ersatz-String (wird anstelle des FindString eingesetzt)
	start	Startposition, ab der gesucht und ersetzt wird
	count	Anzahl der Zeichen, die ab der Startposition nach FindString durchsucht werden sollen
Rückgabewert:		
	string = Leerstring	Kopie von String
	FindString = Leerstring	Kopie von String
	ReplaceString = Leerstring	Kopie von String, in der alle Vorkommen von FindString gelöscht sind
	start > Len(String)	Leerstring
	count = 0	Kopie von String

Funktion STRCMP: Strings vergleichen

Syntax:	**STRCMP**(string1, string2 [, CaseInsensitive])	
Beschreibung:	STRCMP vergleicht eine Zeichenkette mit einem anderen Zeichenkette.	
Parameter:	string1	Zeichenkette, die mit einer zweiten Zeichenkette verglichen werden soll
	string2	Zeichenkette, die mit der ersten Zeichenkette verglichen werden soll
	CaseInsensitive	Vergleich mit/ohne Beachtung der Groß-/Kleinschreibung: ohne Angabe bzw. FALSE = Groß-/Kleinschreibung wird beachtet TRUE = Groß-/Kleinschreibung wird nicht beachtet

Beispiel

```
REG[0]=STRCMP("Hugo", "HUGO")              ; Ergebnis = 32
                                           ; (<> 0, Strings sind nicht identisch)

REG[0]=STRCMP("Hugo", "HUGO", TRUE)        ; Ergebnis = 0
                                           ; (== 0, Stings sind identisch
```

Funktion STRINSERT: String einfügen

Syntax:	**STRINSERT**(string1, string2 , insert position)	
Beschreibung:	STRINSERT fügt eine Zeichenkette an einer bestimmten Position in eine Zeichenkette ein.	
Parameter:	string1	Zeichenkette, in die eine Zeichenkette eingefügt werden soll
	string2	Zeichenkette, die eingefügt werden soll
	insert position	Position, an der die Zeichenkette eingefügt werden soll

Beispiel

```
REG[0]=STRINSERT("Hello!", " world", 5)    ; Ergebnis = "Hello world!")
```

Funktion STRREMOVE: String entfernen

Syntax:	**STRREMOVE**(string1, remove positon , count)
Beschreibung:	STRREMOVE entfernt eine bestimmte Anzahl Zeichen an einer bestimmten Position innerhalb einer Zeichenkette.

Parameter:	string1	Zeichenkette, aus welcher Zeichen entfernt werden sollen
	remove position	Position, ab welcher Zeichen aus der Zeichenkette entfernt werden sollen
	count	Anzahl zu entfernender Zeichen

Beispiel

```
REG[0]=STRREMOVE("Hello world!", 5, 6)     ; Ergebnis = "Hello!")
```

Funktion TRIMLEFT: Leerzeichen links vom String entfernen

Syntax:	**TRIMLEFT**(string1)	
Beschreibung:	TRIMLEFT entfernt Leerzeichen links von einer Zeichenkette.	
Parameter:	string1	Zeichenkette, aus welcher Leerzeichen links der Zeichenkette entfernt werden sollen

Beispiel

```
REG[0]=TRIMLEFT("    Hello!")              ; Ergebnis = "Hello!")
```

Funktion TRIMRIGHT: Leerzeichen rechts vom String entfernen

Syntax:	**TRIMRIGHT**(string1)	
Beschreibung:	TRIMRIGHT entfernt Leerzeichen rechts von einer Zeichenkette.	
Parameter:	string1	Zeichenkette, aus welcher Leerzeichen rechts der Zeichenkette entfernt werden sollen

Beispiel

```
REG[0]=TRIMRIGHT("Hello!    ")             ; Ergebnis = "Hello!")
```

Funktion FORMAT: Werte oder String mit Formatierungskennung einfügen

Syntax:	**FORMAT**(Text mit Formatierungskennung [, Wert1] ... [, Wert28])
Beschreibung:	Die Funktion FORMAT bietet die Möglichkeit, durch Nutzung von Formatierungskennungen, bis zu 28 Werte oder Strings an bestimmten Stellen in einem vorgegebenen Text einzufügen.

Formatierungs-kennungen:	Syntax	%[Flags] [Breite] [.Nachkommastellen] Typ
	Flags	Optionales Zeichen zum Festlegen der Ausgabeformatierung: • rechts- oder linksbündig ("-" für linksbündig) • Vornullen hinzufügen ("0") • default: mit Leerzeichen auffüllen, wenn der auszugebende Wert weniger Stellen hat als mit [Breite] angegeben wurde
	Breite	Das Argument legt die minimale Ausgabebreite einer nicht negativen Zahl fest. Hat der Wert weniger Stellen als durch das Argument festgelegt, werden die fehlenden Stellen mit Leerzeichen aufgefüllt.
	Nachkomma-stellen	Bei einer Gleitkommazahl legt der optionale Parameter die Anzahl von Nachkommastellen fest.
	Typ	Das Typzeichen legt fest, welche Datenformate der Print–Anweisung übergeben werden. Diese Zeichen müssen angegeben werden. • d: Integer Wert • f: Gleitkommazahl • s: String • o: oktal • x: hexadezimal • b: binär

Beispiel

```
DEF VAR1
DEF VAR2
LOAD
VAR1 = 123
VAR2 = FORMAT("Hello %08b %.2f %s!", VAR1 + 1, 987.654321, "world")
                                    ; Ergebnis = "Hello 01111100 987.65 world!"
END_LOAD
```

Siehe auch

Verwendung von Strings (Seite 104)

8.3.40 WHILE-/UNTIL-Schleifen

Beschreibung

Mit den DO-LOOP-Kommandos ist es möglich eine Schleife zu realisieren. Je nach Projektierung wird diese entweder solange durchlaufen wie eine Bedingung erfüllt ist (WHILE) oder bis eine Bedingung zutrifft (UNTIL).

Da Schleifen je nach Projektierung die Systemperformance beeinträchtigen können, sollten Sie diese mit Bedacht einsetzen und auf zeitintensive Aktionen in den Schleifen verzichten.

Es empfiehlt sich als Laufvariable z.B. ein Register (REG[]) einsetzen, da normale Anzeigevariablen (insbesondere solche mit System- oder Anwendervariablen--Anbindung) durch die extrem häufigen Aktualisierungen bzw. Schreibvorgänge ebenfalls die Systemperformance beeinträchtigen können.

Mit Hilfe der Funktion DEBUG (siehe Kapitel DEBUG (Seite 145)) kann die Laufzeit von "Run MyScreens"-Methoden ermittelt werden. Damit lassen sich gegebenenfalls durch Schleifen erzeugte Probleme (hohe CPU-Last, verminderte Reaktionsfähigkeit) identifizieren.

Hinweis

Da jede FOR-Schleife durch eine WHILE-Schleife ersetzt werden kann, wird in EasyScreen die Syntax zur Formulierung einer FOR-Schleife nicht unterstützt.

Programmierung

```
DO
   <Anweisungen>
LOOP_WHILE <Bedingung zum Fortsetzen der Schleife>

DO
   <Anweisungen>
LOOP_UNTIL <Bedingung zur Beendigung der Schleife>

DO_WHILE <Bedingung zum Fortsetzen der Schleife>
   <Anweisungen>
LOOP

DO_UNTIL <Bedingung zur Beendigung der Schleife>
   <Anweisungen>
LOOP
```

Beispiel

```
REG[0] = 5
DO
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
   REG[1] = -5

   DO
      DEBUG("INNER: " << REG[1])
      REG[1] = REG[1] + 1
   LOOP_WHILE REG[1] < 0

LOOP_WHILE REG[0] < 10
```

```
REG[0] = 5
DO
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
   REG[1] = -5

   DO
      DEBUG("INNER: " << REG[1])
      REG[1] = REG[1] + 1
   LOOP_UNTIL 0 <= REG[1]

LOOP_WHILE 10 > REG[0]
```

```
REG[0] = 5
DO_WHILE 10 > REG[0]
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
   REG[1] = -5

   DO_UNTIL 0 <= REG[1]
     DEBUG("INNER: " << REG[1])
     REG[1] = REG[1] + 1
   LOOP

LOOP
```

```
REG[0] = 5
DO_WHILE 10 > REG[0]
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
   REG[1] = -5

   DO
     DEBUG("INNER: " << REG[1])
     REG[1] = REG[1] + 1
   LOOP_UNTIL 0 <= REG[1]

LOOP
```

8.3.41 Zyklische Ausführung von Skripten: START_TIMER, STOP_TIMER

Beschreibung

Mit Hilfe von Timern können SUB-Methoden zyklisch aufgerufen werden. Dafür gibt es die Funktionen START_TIMER() und STOP_TIMER().

Hinweis

Es kann nur ein Timer je SUB-Methode projektiert werden.

Programmierung

Syntax:	**START_TIMER**("SUB-Name", Intervall)	
Beschreibung:	Zyklische Bearbeitung einer SUB-Methode starten	
Parameter:	SUB-Name:	Name der zyklisch aufzurufenden SUB-Methode
	Intervall:	Intervall in Millisekunden

Syntax:	**STOP_TIMER**("SUB-Name")	
Beschreibung:	Zyklische Bearbeitung einer SUB-Methode stoppen	
Parameter:	SUB-Name:	Name der SUB-Methode, deren Timer gestoppt werden soll

Beispiel

```
//M(TimerSample/"My timer")
DEF MyVariable=(I//0/,"Number of cyclic calls:"/WR1)
VS1=("Start%ntimer")
VS2=("Stop%ntimer")

SUB(MyTimerSub)
   MyVariable = MyVariable + 1
END_SUB

PRESS(VS1)
   ;Calls SUB "MyTimerSub" every 1000 milliseconds
   START_TIMER("MyTimerSub", 1000)
END_PRESS

PRESS(VS2)
   STOP_TIMER("MyTimerSub")
END_PRESS
```

Wird START_TIMER erneut für einen bereits einer SUB-Methode zugewiesenen Timer aufgerufen, dann wird das neue Intervall übernommen, wenn er sich unterscheidet. Ansonsten wird dieser zweite Aufruf ignoriert.

Systembedingt, ist das kleinste Intervall 100 Millisekunden.

Wird STOP_TIMER für eine SUB-Methode aufgerufen, für die derzeit kein Timer läuft, dann wird dieser Aufruf ignoriert.

Grafische und logische Elemente 9

9.1 Linie, Trennlinie, Rechteck, Kreis und Ellipse

9.1.1 Grafikelemente definieren

Beschreibung

Linien, Trennlinien, Rechtecke und Ellipsen/Kreise werden in der LOAD-Methode projektiert:

- Transparente Rechtecke werden erreicht, indem die Füllfarbe auf die Systemhintergrundfarbe gesetzt wird.
- Beim Element ELLIPSE entsteht ein Kreis, indem die Höhe und die Breite gleich gesetzt wird.
- Horizontale und vertikale Trennlinien haben immer die exakte Fensterbreite bzw. Fensterhöhe. Durch Trennlinien werden keine Scrollbars verursacht.
  Wenn Sie für die Darstellung von Trennlinien bisher das Element RECT verwendet haben, z. B. RECT(305,0,1,370,0,0,1), wird dieses von der Programmierunterstützung erkannt und automatisch in V_SEPARATOR(305,1,"#87a5cd", 1) konvertiert. Dies gilt sowohl für vertikale als auch für horizontale Trennlinien.

Element LINE

Programmierung:

Syntax:	LINE (x1, y1, x2, y2, color, style, tag, visible)	
Beschreibung:	Linie definieren	
Parameter:	x1	x-Koordinate des Startpunktes
	y1	y-Koordinate des Startpunktes
	x2	x-Koordinate des Endpunktes
	y2	y-Koordinate des Endpunktes
	color	Farbe der Linie
	style	Stil der Linie: 1 = durchgezogen 2 = gestrichelt 3 = gepunktet 4 = strichpunktiert
	tag	Ein Grafikelement bzw. eine Gruppe von Grafikelemente können Sie über dieses tag mit den Funktionen SHOW_GRAPHIC, HIDE_GRAPHIC und DELETE_GRAPHIC ein- bzw. ausblenden und löschen.
	visible	Grafikelement sichtbar (TRUE/FALSE)

Element RECT

Programmierung:

Syntax:	RECT (x, y, w, h, frame color, fill color, style, tag, visible)	
Beschreibung:	Rechteck definieren	
Parameter:	x	x-Koordinate Links-Oben
	y	y-Koordinate Links-Oben
	w	Breite
	h	Höhe
	frame color	Farbe des Rahmens
	fill color	Füllfarbe
	style	Stil des Rahmens: 1 = durchgezogen 2 = gestrichelt 3 = gepunktet 4 = strichpunktiert
	tag	Ein Grafikelement bzw. eine Gruppe von Grafikelemente können Sie über dieses tag mit den Funktionen SHOW_GRAPHIC, HIDE_GRAPHIC und DELETE_GRAPHIC ein- bzw. ausblenden und löschen.
	visible	Grafikelement sichtbar (TRUE/FALSE)

Element ELLIPSE

Programmierung:

Syntax:	ELLIPSE(x, y, w, h, frame color, fill color, style, tag, visible)	
Beschreibung:	Ellipse bzw. Kreis definieren	
Parameter:	x	x-Koordinate Links-Oben
	y	y-Koordinate Links-Oben
	w	Breite
	h	Höhe
	frame color	Farbe des Rahmens
	fill color	Füllfarbe
	style	Stil des Rahmens: 1 = durchgezogen 2 = gestrichelt 3 = gepunktet 4 = strichpunktiert
	tag	Ein Grafikelement bzw. eine Gruppe von Grafikelemente können Sie über dieses tag mit den Funktionen SHOW_GRAPHIC, HIDE_GRAPHIC und DELETE_GRAPHIC ein- bzw. ausblenden und löschen.
	visible	Grafikelement sichtbar (TRUE/FALSE)

Bei gleichem Wert für Höhe und Breite entsteht ein Kreis.

Element V_SEPARATOR

Programmierung:

Syntax:	V_SEPARATOR (x,w,color,pen)	
Beschreibung:	Vertikale Trennlinie definieren	
Parameter:	x	x-Position
	pen width	Linienstärke
	color	Farbe
	style	Stil der Linie: 1 = durchgezogen 2 = gestrichelt 3 = gepunktet 4 = strichpunktiert
	tag	Ein Grafikelement bzw. eine Gruppe von Grafikelemente können Sie über dieses tag mit den Funktionen SHOW_GRAPHIC, HIDE_GRAPHIC und DELETE_GRAPHIC ein- bzw. ausblenden und löschen.
	visible	Grafikelement sichtbar (TRUE/FALSE)

Element H_SEPARATOR

Programmierung:

Syntax:	H_SEPARATOR (y,h,color,pen)	
Beschreibung:	Horizontale Trennlinie definieren	
Parameter:	y	y-Position
	pen width	Linienstärke
	color	Farbe
	style	Stil der Linie: 1 = durchgezogen 2 = gestrichelt 3 = gepunktet 4 = strichpunktiert
	tag	Ein Grafikelement bzw. eine Gruppe von Grafikelemente können Sie über dieses tag mit den Funktionen SHOW_GRAPHIC, HIDE_GRAPHIC und DELETE_GRAPHIC ein- bzw. ausblenden und löschen.
	visible	Grafikelement sichtbar (TRUE/FALSE)

Weitere Informationen

Im Abschnitt Grafikfunktionen (Seite 192) sind die Funktionen SHOW_GRAPHIC, HIDE_GRAPHIC, CLEAR_BACKGROUND und DELETE_GRAPHIC beschrieben. Mit diesen Funktionen können Sie die Anzeige der Grafikelemente steuern.

9.1.2 Grafikfunktionen

Übersicht

Die folgenden Grafikfunktionen stehen zur Verfügung:

- CLEAR_BACKGROUND
- DELETE_GRAPHIC
- HIDE_GRAPHIC
- SHOW_GRAPHIC

Die Funktionen können Sie auf die Grafikelemente LINE, RECT, ELLIPSE, V_SEPARATOR und H_SEPARATOR (Seite 189) anwenden.

Funktion CLEAR_BACKGROUND: Löschen von Grafikelemente

Mit der Funktion CLEAR_BACKGROUND können die Grafikelemente gelöscht werden.

Syntax:	CLEAR_BACKGROUND()	
Beschreibung:	Löschen von Grafikelementen. Der von den Grafikobjekten belegte Speicher wird freigegeben.	
Parameter:	- keine -	

Funktion DELETE_GRAPHIC: Löschen von einem Grafikelement bzw. einer Gruppe von Grafikelementen

Mit der Funktion DELETE_GRAPHIC können Sie ein spezifisches Grafikelement oder eine Gruppe von Grafikelementen löschen. Bei den entsprechenden Grafikelemente müssen Sie einen Wert im Parameter "tag" definieren.

- Gegenüber der Funktion CLEAR_BACKGROUND können Sie spezifische Grafikelemente löschen.
- Gegenüber der Funktion HIDE_GRAPHIC wird der von den Grafikobjekten belegte Speicher freigegeben.

Syntax:	**DELETE_GRAPHIC**(tag)	
Beschreibung:	Löschen von einem Grafikelement bzw. einer Gruppe von Grafikelementen deren Wert im Parameter "tag" identisch ist.	
Parameter:	tag	Der Parameter "tag" kann in den Grafikelementen definiert werden. Sie können auch mehrere Grafikelemente mit dem selben Wert definieren und damit eine Gruppe von Grafikelementen bilden.

Beispiel

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
```

```
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
DELETE_GRAPHIC(5)                                    ;die 3 Grafikelemente
                                                     vom Typ LINE mit tag=5
                                                     werden gelöscht
```

Funktion HIDE_GRAPHIC: Ausblenden von einem Grafikelement bzw. einer Gruppe von Grafikelementen

Mit der Funktion HIDE_GRAPHIC können Sie ein spezifisches Grafikelement oder eine Gruppe von Grafikelementen ausblenden. Bei den entsprechenden Grafikelemente müssen Sie einen Wert im Parameter "tag" definieren.

Syntax:	HIDE_GRAPHIC(tag)	
Beschreibung:	Ausblenden von einem Grafikelement bzw. einer Gruppe von Grafikelementen deren Wert im Parameter "tag" identisch ist.	
Parameter:	tag	Der Parameter "tag" kann in den Grafikelementen definiert werden. Sie können auch mehrere Grafikelemente mit dem selben Wert definieren und damit eine Gruppe von Grafikelementen bilden.

Beispiel

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
HIDE_GRAPHIC(5)                                      ;die 3 Grafikelemente
                                                     vom Typ LINE mit tag=5
                                                     werden ausgeblendet
```

Funktion SHOW_GRAPHIC: Einblenden von einem Grafikelement bzw. einer Gruppe von Grafikelementen

Mit der Funktion SHOW_GRAPHIC können Sie ein spezifisches Grafikelement oder eine Gruppe von Grafikelementen einblenden. Bei den entsprechenden Grafikelemente müssen Sie einen Wert im Parameter "tag" definieren.

Syntax:	SHOW_GRAPHIC(tag)	
Beschreibung:	Einblenden von einem Grafikelement bzw. einer Gruppe von Grafikelementen deren Wert im Parameter "tag" identisch ist.	
Parameter:	tag	Der Parameter "tag" kann in den Grafikelementen definiert werden. Sie können auch mehrere Grafikelemente mit dem selben Wert definieren und damit eine Gruppe von Grafikelementen bilden.

Beispiel

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
SHOW_GRAPHIC(5)                                          ;die 3 Grafikelemente
                                                         vom Typ LINE mit tag=5
                                                         werden eingeblendet
```

9.2 Array definieren

Definition

Mit Hilfe eines Arrays werden Daten eines einheitlichen Datentyps im Speicher geordnet so abgelegt, dass ein Zugriff auf die Daten über einen Index möglich wird.

Beschreibung

Arrays können ein- oder zweidimensional sein. Ein eindimensionales Array wird wie ein zweidimensionales Array mit einer Zeile oder einer Spalte betrachtet.

Arrays werden mit der Kennung //A definiert und durch //END beendet. Die Anzahl der Zeilen und Spalten ist beliebig. Ein Array hat den folgenden Aufbau:

Programmierung

Syntax:	**//A**(Bezeichner) (a/b...) (c/d...) ... //END	
Beschreibung:	Array definieren	
Parameter:	Bezeichner	Name des Arrays
	a, b, c, d	Werte des Arrays Werte vom Typ STRING müssen in Doppelhochkomma angegeben werden.

Beispiel

```
//A(Gewinde)                    ; Größe/Steigung/Kerndurchmesser
```

```
(0.3 / 0.075 / 0.202)
(0.4 / 0.1   / 0.270)
(0.5 / 0.125 / 0.338)
(0.6 / 0.15  / 0.406)
(0.8 / 0.2   / 0.540)
(1.0 / 0.25  / 0.676)
(1.2 / 0.25  / 0.676)
(1.4 / 0.3   / 1.010)
(1.7 / 0.35  / 1.246)
//END
```

9.2.1 Auf den Wert eines Array-Elements zugreifen

Beschreibung

Mit der Eigenschaft Wert (Bezeichner.val) kann der Wert eines Arrrayzugriffs weitergegeben werden.

Der Zeilenindex (Zeilennummer des Arrays) und der Spaltenindex (Spaltennummer des Arrays) beginnen jeweils bei 0. Zeigt ein Zeilenindex oder Spaltenindex außerhalb des Arrays, wird der Wert 0 oder ein Leerstring ausgegeben und die Variable ERR hat den Wert TRUE. Die Variable ERR ist ebenfalls TRUE, wenn ein Suchbegriff nicht gefunden wurde.

Programmierung

Syntax:	Bezeichner [Z,[M[,C]]].val oder Bezeichner [Z,[M[,C]]]
Beschreibung:	Zugriff auf eindimensionalen Array mit nur einer Spalte
Syntax:	Bezeichner [S,[M[,C]]].val] oder Bezeichner [S,[M[,C]]] oder
Beschreibung:	Zugriff auf eindimensionalen Array mit nur einer Zeile
Syntax:	Bezeichner [Z,S,[M[,C]]].val oder Bezeichner [Z,S,[M[,C]]]
Beschreibung:	Zugriff auf zweidimensionalen Array

Parameter:	Bezeichner:	Name des Arrays	
	Z:	Zeilenwert (Zeilenindex oder Suchbegriff)	
	S:	Spaltenwert (Spaltenindex oder Suchbegriff)	
	M:	Zugriffsmodus	
		0	Direkt
		1	Zeile suchend, Spalte direkt
		2	Zeile direkt, Spalte suchend
		3	Suchend
		4	Zeilenindex suchen
		5	Spaltenindex suchen
	C:	Vergleichsmodus	
		0	Suchbegriff muss sich im Wertebereich der Zeile oder Spalte befinden.
		1	Suchbegriff muss exakt gefunden werden.

Beispiel:		
	`VAR1 = MET_G[REG[3],1,0].VAL`	`;Var1 einen Wert aus Array MET_G zuordnen`

Zugriffsmodus

- **Zugriffsmodus "Direkt"**
  Beim Zugriffsmodus "Direkt" (M = 0) erfolgt der Zugriff auf das Array mit dem Zeilenindex in Z und dem Spaltenindex in S. Der Vergleichsmodus C wird nicht ausgewertet.
- **Zugriffsmodus "Suchend"**
  Beim Zugriffsmodus M = 1, 2 oder 3 erfolgt die Suche immer in der Zeile 0 oder Spalte 0.

Modus M	Zeilenwert Z	Spaltenwert S	Ausgabewert
0	Zeilenindex	Spaltenindex	Wert aus Zeile Z und Spalte S
1	Suchbegriff: Suche in Spalte 0	Spaltenindex der Spalte, aus der der Wert gelesen wird	Wert aus gefundener Zeile und Spalte S
2	Zeilenindex der Zeile, aus der der Returnwert gelesen wird	Suchbegriff: Suche in Zeile 0	Wert aus Zeile Z und gefundener Spalte
3	Suchbegriff: Suche in Spalte 0	Suchbegriff: Suche in Zeile 0	Wert aus gefundener Zeile und gefundener Spalte
4	Suchbegriff: Suche in Spalte S	Spaltenindex der Spalte, in der gesucht wird	Zeilenindex
5	Zeilenindex der Zeile, in der gesucht wird.	Suchbegriff: Suche in Zeile Z	Spaltenindex

Vergleichsmodus

Bei Verwendung von Vergleichsmodus C = 0 muss der Inhalt der Suchzeile oder der Suchspalte in aufsteigender Reihenfolge sortiert sein. Ist der Suchbegriff kleiner als das erste Element oder größer als das letzte, wird der Wert 0 oder ein Leerstring ausgegeben und die Errorvariable ERR ist TRUE.

Bei Verwendung von Vergleichsmodus C = 1 muss sich der Suchbegriff in der Suchzeile oder in der Suchspalte finden lassen. Ist der Suchbegriff nicht zu finden, wird der Wert 0 oder ein Leerstring ausgegeben und die Errorvariable ERR ist TRUE.

9.2.2 Beispiel: Zugriff auf ein Array-Element

Voraussetzung

Nachstehend werden zwei Arrays definiert, die Voraussetzung für die folgenden Beispiele sind:

```
//A(Gewinde)
                (0.3 / 0.075 / 0.202)
                (0.4 / 0.1   / 0.270)
                (0.5 / 0.125 / 0.338)
                (0.6 / 0.15  / 0.406)
                (0.8 / 0.2   / 0.540)
                (1.0 / 0.25  / 0.676)
                (1.2 / 0.25  / 0.676)
                (1.4 / 0.3   / 1.010)
                (1.7 / 0.35  / 1.246)
//END


//A(Array2)
                ("BEZ" /     "STG" /     "KDM" )
                (0.3 /       0.075 /     0.202 )
                (0.4 /       0.1 /       0.270 )
                (0.5 /       0.125 /     0.338 )
                (0.6 /       0.15 /      0.406 )
                (0.8 /       0.2 /       0.540 )
                (1.0 /       0.25 /      0.676 )
                (1.2 /       0.25 /      0.676 )
                (1.4 /       0.3 /       1.010 )
                (1.7 /       0.35 /      1.246 )
//END
```

Beispiele

- **Zugriffsmodus Beispiel 1:**
  In Z steht der Suchbegriff. Dieser Begriff wird immer in der Spalte 0 gesucht. Mit dem Zeilenindex des gefundenen Begriffs wird der Wert aus der Spalte S ausgegeben:
  ```
  VAR1 = Gewinde[0.5,1,1]   ;VAR1 hat den Wert 0.125
  ```
  Erklärung:
  Suche in der Spalte 0 von Array "Gewinde" den Wert 0.5 und gib den in der Spalte 1 gefundenen Wert der gleichen Zeile aus.
- **Zugriffsmodus Beispiel 2:**
  In S steht der Suchbegriff. Dieser Begriff wird immer in der Zeile 0 gesucht. Mit dem Spaltenindex des gefundenen Begriffs wird der Wert aus der Zeile Z ausgegeben:
  ```
  VAR1 = ARRAY2[3,"STG",2]   ;VAR1 hat den Wert 0.125
  ```
  Erklärung:
  Suche in der Zeile 0 von Array "Array2" die Spalte mit dem Inhalt "STG". Gib den Wert aus der gefundenen Spalte und der Zeile mit dem Index 3 aus.
- **Zugriffsmodus Beispiel 3:**
  In Z und S steht jeweils ein Suchbegriff. Mit dem Begriff in Z wird in der Spalte 0 der Zeilenindex und mit dem Begriff in S wird in der Zeile 0 der Spaltenindex gesucht. Mit dem gefundenen Zeilenindex und Spaltenindex wird der Wert aus dem Array ausgegeben:
  ```
  VAR1 = ARRAY2[0.6,"STG",3]   ;VAR1 hat den Wert 0.15
  ```
  Erklärung:
  Suche in der Spalte 0 von Array "Array2" die Zeile mit dem Inhalt 0.6, suche in der Zeile 0 von Array2 die Spalte mit dem Inhalt "STG". Gib den Wert aus der gefundenen Zeile und Spalte nach VAR1.
- **Zugriffsmodus Beispiel 4:**
  In Z steht der Suchbegriff. In S steht Spaltenindex der Spalte, in der gesucht wird. Der Zeilenindex des gefundenen Begriffs wird ausgegeben:
  ```
  VAR1 = Gewinde[0.125,1,4]   ;VAR1 hat den Wert 2
  ```
  Erklärung:
  Suche in der Spalte 1 von Array "Gewinde" den Wert 0.125 und gib den Zeilenindex des gefundenen Wertes nach VAR1.
- **Zugriffsmodus Beispiel 5:**
  In Z steht der Zeilenindex der Zeile, in der gesucht wird. Der Suchbegriff steht in S. Der Spaltenindex des gefundenen Begriffs wird ausgegeben:
  ```
  VAR1 = Gewinde[4,0.2,5,1]  ;VAR1 hat den Wert 1
  ```
  Erklärung:
  Suche in der Zeile 4 von Array "Gewinde" den Wert 0.2 und gib den Spaltenindex des gefundenen Wertes nach VAR1. Vergleichsmodus 1 wurde gewählt, da die Werte der Zeile 4 nicht aufsteigend sortiert sind.

9.2.3 Zustand eines Array-Elements abfragen

Beschreibung

Mit der Eigenschaft Zustand kann abgefragt werden, ob ein Arrayzugriff einen gültigen Wert liefert.

Programmierung

Syntax:	Bezeichner [Z, S, [M[,C]]]**.vld**
Beschreibung:	Die Eigenschaft ist nur lesbar.
Parameter:	Bezeichner Name des Arrays
Rückgabewert:	FALSE = ungültiger Wert
	TRUE = gültiger Wert

Beispiel

```
DEF MPIT = (R///"MPIT",,"MPIT",""/wr3)
DEF PIT  = (R///"PIT",,"PIT",""/wr3)
PRESS(VS1)
  MPIT = 0.6
  IF MET_G[MPIT,0,4,1].VLD == TRUE
    PIT    = MET_G[MPIT,1,0].VAL
    REG[4] = PIT
    REG[1] = "OK"
  ELSE
    REG[1] = "ERROR"
  ENDIF
END_PRESS
```

9.3 Tabellenbeschreibung (Grid)

Definition

Die Werte einer Tabelle (Grid) werden im Gegensatz zum Array laufend aktualisiert. Es handelt sich um eine tabellarische Darstellung von Werten, welche z. B. aus der NC- oder PLC stammen können. Die Tabelle ist spaltenorientiert organisiert und enthält somit innerhalb einer Spalte immer gleichartige Variablen.

Zuordnung

Der Verweis auf eine Tabellenbeschreibung wird in der Variablenbeschreibung vorgenommen:

- Die Variablendefinition bestimmt die anzuzeigenden Werte, die Definition der Tabellenelemente das Aussehen und die Anordnung am Bildschirm. Die Eigenschaften der Ein-/Ausgabefelder aus der Definitionszeile der Variablen werden in der Tabelle übernommen.
- Der sichtbare Bereich der Tabelle wird durch die Breite und Höhe des Ein-/Ausgabefelds bestimmt. Sind mehr Zeilen oder Spalten vorhanden als im sichtbaren Bereich Platz finden, kann horizontal und vertikal geblättert werden.

Tabellenbezeichner

Bezeichner einer Tabelle gleichartiger Werte des NCK oder der PLC, die über einen Baustein eines Kanals adressiert werden können. Der Tabellenbezeichner wird durch ein vorangestelltes %-Zeichen von Grenzwerten oder dem Toggle-Feld unterschieden. Dem Tabellenbezeichner kann, getrennt durch Komma, noch ein Dateiname folgen, der die Datei angibt, in der die Tabellenbeschreibung definiert ist.

Bezeichner einer Tabelle gleichartiger Werte z. B. aus NCK oder PLC. Der Tabellenbezeichner wird an der Position angegeben, an der die Angabe für Grenzwerte oder Toggle-Feld projektiert wird. Der Tabellenbezeichner wird durch ein vorangestelltes %-Zeichen eingeleitet. Danach kann, getrennt durch Komma, noch ein Dateiname folgen. Dieser gibt die Datei an, in der die Tabellenbeschreibung definiert ist. Standardmäßig wird die Tabelle in der Projektierungsdatei gesucht, wo die Maske selbst projektiert ist.

Eine Tabellendefinition kann auch dynamisch, z. B. in der LOAD-Methode, durch die Funktion Load Grid (LG) geladen werden.

System- oder Anwendervariable

Dieser Parameter bleibt für Tabellen leer, weil die anzuzeigenden Variablen im Detail in den Spaltendefinitionszeilen angegeben werden. Die Tabellenbeschreibung kann dynamisch bereitgestellt werden.

Beispiel

Definition einer Variable MyGridVar, die in ihrem Ein-/Ausgabefeld (Abstand links: 100, Abstand oben: Standard, Breite: 350, Höhe: 100) ein Grid „MyGrid1“ darstellt.

```
DEF MyGridVar=(V/% MyGrid1///////100,,350,100)
```

Siehe auch

Parameter von Variablen (Seite 91)

Load Grid (LG) (Seite 162)

9.3.1 Tabelle (Grid) definieren

Beschreibung

Der Tabellenblock besteht aus:

- Kopfbeschreibung
- 1 bis n Spaltenbeschreibungen

Programmierung

Syntax:	**//G**(Tabellenbezeichner/Tabellentyp/Zeilenanzahl/ [Attribut feste Zeile],[Attribut feste Spalte])		
Beschreibung:	Tabelle (Grid) definieren		
Parameter:	Tabellenbezeichner	Der Tabellenbezeichner wird hier ohne führendes %-Zeichen verwendet. Er kann nur einmal in einem Dialog verwendet werden.	
	Tabellentyp	0 (Voreinstellung)	Tabelle für PLC- oder Anwenderdaten (NCK- und kanalspezifische Daten)
		1	und weitere reserviert
	Zeilenzahl	Zeilenzahl einschließlich Kopfzeile	
		Die feste Zeile oder feste Spalte wird nicht gerollt. Die Spaltenanzahl ergibt sich aus der Anzahl der projektierten Spalten.	
	Anzeige Spaltenüberschriften-Zeile	-1:	Spaltenüberschriften-Zeile wird nicht angezeigt
		0:	Spaltenüberschriften-Zeile wird angezeigt (default)
	Anzahl feste Spalten	0:	keine feste Spalte
		1 ... n:	Anzahl der Spalten die immer sichtbar sein sollen, d. h. nicht horizontal gescrollt werden

Beispiele

`//G(grid1/0/5/-1,2)`	Spaltenüberschriften-Zeile wird ausgeblendet und 2 feste Spalten
`//G(grid1/0/5/,1)`	Spaltenüberschriften-Zeile wird angezeigt und 1 feste Spalte
`//G(grid1/0/5)`	Spaltenüberschriften-Zeile wird angezeigt und keine feste Spalten

9.3.2 Spalten definieren

Beschreibung

Für Tabellen (Grid) ist es sinnvoll, Variablen mit Index zu benutzen. Die Indexnummer ist bei PLC- oder NC-Variablen mit einem oder mehreren Indizes von Bedeutung.

Die in einer Tabelle angezeigten Werte können Sie, im Rahmen der durch die Attribute festgelegten Rechte und innerhalb der ggf. definierten Grenzen, direkt modifizieren.

Programmierung

Syntax:	(Typ/Grenzwerte/leer/Langtext,Spaltenüberschrift/Attribute/Hilfebild/ System- oder Anwendervariable/Spaltenbreite/Offset1, Offset2, Offset3) Siehe auch Erweiterte Projektierungssyntax (Seite 46).	
Beschreibung:	Spalten definieren	
Parameter:	analog zu Variablen	
	Typ	Datentyp
	Grenzwerte	Grenzwert MIN, Grenzwert MAX
	Langtext, Spaltenüberschrift	
	Attribute	
	Hilfebild	
	System- oder Anwendervariable	Als Variable sind PLC- oder NC-Variable in Doppelhochkomma anzugeben.
	Spaltenbreite	Angabe in Pixel
	Offset	Die Schrittweiten, in denen der jeweilige Index hochgezählt werden soll, um die Spalte zu füllen, werden im zugeordneten Offset-Parameter angegeben: • Offset1: Schrittweite für den 1. Index • Offset2: Schrittweite für den 2. Index • Offset3: Schrittweite für den 3. Index

Spaltenüberschrift aus Textdatei

Die Spaltenüberschrift kann als Text oder Textnummer ($8xxxx) angegeben werden und wird gegebenenfalls nicht gerollt.

Spalteneigenschaften ändern

Die dynamisch änderbaren (schreibbaren) Spalteneigenschaften heißen:

- Grenzwerte (min,max)
- Spaltenüberschrift (st)
- Attribute (wr, ac und li)
- Hilfebild (hlp)

Die Änderung einer Spalteneigenschaft erfolgt über den Bezeichner der Variablen aus der Definitionszeile und über den Index der Spalte (mit 1 beginnend).

Beispiel: `VAR1[1].st="Spalte 1"`

Für Spaltendefinition können die Attribute wr, ac und li angeben werden.

Siehe auch

Load Grid (LG) (Seite 162)

9.3.3 Fokussteuerung in der Tabelle (Grid)

Beschreibung

Mit Eigenschaften Row und Col kann der Fokus innerhalb einer Tabelle gesetzt und ermittelt werden:

- Bezeichner.**Row**
- Bezeichner.**Col**

Programmierung

Jede Zelle einer Tabelle besitzt die Eigenschaften Val und Vld.

Für das Schreiben und Lesen der Zelleneigenschaften ist neben dem Bezeichner der Variablen aus der Definitionszeile ein Zeilen- und Spaltenindex anzugeben.

Syntax:	Bezeichner[Zeilenindex, Spaltenindex].Val oder Bezeichner[Zeilenindex, Spaltenindex]
Beschreibung:	Eigenschaften Val
Syntax:	Bezeichner[Zeilenindex, Spaltenindex].Vld
Beschreibung:	Eigenschaften Vld

Beispiel

```
Var1[2,3].val=1.203
```

Wird kein Zeilen- und Spaltenindex angegeben, gelten die Indizes der fokussierten Zelle, das entspricht:

```
Var1.Row =2
Var1.Col=3
Var1.val=1.203
```

9.4 Custom Widgets

9.4.1 Custom Widgets definieren

Beschreibung

Mit Hilfe eines Custom Widgets werden anwenderspezifische Anzeigeelemente im Dialog projektiert.

Software-Option

Um Custom Widgets in Dialogen zu nutzen benötigen Sie zusätzlich folgende Software-Option:

"SINUMERIK Integrate Run MyHMI /3GL" (6FC5800-0AP60-0YB0)

Programmierung

Definition:	**DEF** (Name)	
Syntax:	**(W///"",**"(Bibliotheksname).(Klassenname)**"/////**a**,**b**,**c**,**d**);**	
Beschreibung:	W	Custom Widget definieren
Parameter:	Name	Custom Widget Name, frei wählbar
	Bibliotheksname	Frei wählbar, Name der dll (Windows) oder so (Linux) Bibliotheksdatei
	Klassenname	Frei wählbar, Name der Klassenfunktion aus der zuvor benannten Bibliothek
	a, b, c, d	Position und Größe der Projektierung

Beispiel

Ein Custom Widget wird in folgender Weise in der Dialogprojektierung definiert:

```
DEF Cus = (W///"","slestestcustomwidget.SlEsTestCustomWidget"/////
20,20,250,100);
```

9.4.2 Aufbau der Custom Widget Bibliothek

Beschreibung

Im Wesentlichen enthält die Custom Widget Bibliothek eine definierte Klasse. Der Name dieser Klasse muss in der Dialogprojektierung neben dem Bibliotheksnamen angegeben werden. Vom Bibliotheksnamen ausgehend greift "Run MyScreens" auf eine gleichnamige dll-Datei zu, z. B.:

```
slestestcustomwidget.dll
```

Programmierung

Die Klassendefinition der dll-Datei sollte so aussehen:

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT


class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
....
public slots:
   bool serialize(const QString& szFilePath, bool bIsStoring);
....
}
```

9.4.3 Aufbau der Custom Widget Schnittstelle

Beschreibung

Um das Custom Widget im Dialog anzeigen zu können, wird die Bibliothek durch eine Schnittstelle ergänzt. Diese enthält Makrodefinitionen mit der "Run MyScreens" das Custom Widget initiiert. Die Schnittstelle liegt in Form einer cpp-Datei vor. Der Datei-Name ist frei wählbar, z. B.:
`sleswidgetfactory.cpp`

Programmierung

Die Schnittstelle wird wie folgt definiert:

```
#include "slestestcustomwidget.h"     ; Die Header-Datei des betreffenden Custom
                                      Widgets wird am Dateianfang eingezogen
....
//Makros                              ; Makrodefinitionen werden nicht geändert
....
WIDGET_CLASS_EXPORT(SlEsTestCustom-   ; Das betreffende Custom Widget wird am Da-
Widget)                               teiende deklariert
```

Beispiel

Inhalt der Datei sleswidgetfactory.cpp für ein Custom Widget mit dem Klassennamen "SlEsTestCustomWidget":

```
#include <Qt/qglobal.h>
```

```
#include "slestestcustomwidget.h"

////////////////////////////////////////////////////////////////////////
// MAKROS FOR PLUGIN DLL-EXPORT - DO NOT CHANGE
////////////////////////////////////////////////////////////////////////

#ifndef Q_EXTERN_C
#ifdef __cplusplus
#define Q_EXTERN_C    extern "C"
#else
#define Q_EXTERN_C    extern
#endif
#endif

#define SL_ES_FCT_NAME(PLUGIN) sl_es_create_ ##PLUGIN
#define SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( IMPLEMENTATION , PARAM) \
    { \
    IMPLEMENTATION *i = new PARAM; \
    return i; \
    }

#ifdef Q_WS_WIN
#   ifdef Q_CC_BOR
#       define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C __declspec(dllexport) void* \
    __stdcall SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#   else
#       define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C __declspec(dllexport) void* SL_ES_FCT_NAME(PLUGIN) \
    (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#   endif
#else
#   define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C void* SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#endif

#define WIDGET_CLASS_EXPORT(CLASSNAME)  \
    EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(CLASSNAME,CLASSNAME(pParent))

////////////////////////////////////////////////////////////////////////
// FOR OEM USER - please declare here your widget classes for export
```

```
////////////////////////////////////////////////////////////////////////

WIDGET_CLASS_EXPORT(SlEsTestCustomWidget)
```

9.4.4 Interaktion zwischen Custom Widget und Dialog - automatischer Datenaustausch

Custom Widgets interagieren mit Dialogen und können Werte anzeigen oder manipulieren.

Bedingungen

Ein automatischer Datenaustausch erfolgt unter folgenden Bedingungen:

Bedingung	Richtung
Beim Start oder Rückübersetzen eines Dialogs	Dialog → Custom Widget
Beim Ausführen des GC-Befehls zur Generierung von Zyklenaufrufen	Custom Widget → Dialog

Programmierung

Folgende Definitionen sind für die Interaktionen notwendig:

Erweiterung der Dialogprojektierung

Definition:	**DEF** (Variable)	
Syntax:	**(**(Typ)**//5/"","**(Variable)**","/wr2/)**	
Variablentyp:	Typ	Standard Eingabefeld (kein Grid oder Toggle) mit beliebigem Datentyp (kein W)
Parameter:	Variable	Beliebige Benennung einer Variable für Datenaustausch
Eingabemodus:	wr2	Lesen und schreiben

Beispiel

```
DEF CUSVAR1 = (R//5/"","CUSVAR1",""/wr2/)
```

Erweiterung der Klassendefinition

In der Klassendefinition des Custom Widgets muss ein QProperty angelegt werden, dessen Name identisch mit der ausgewählten Variable der Dialogprojektierung ist, z. B.:

```
Q_PROPERTY(double CUSVAR1 READ cusVar1 WRITE setCusVar1);
```

Beispiel

Die Klassendefinition der dll-Datei sollte so aussehen:

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT
```

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
 Q_OBJECT
 Q_PROPERTY(double CUSVAR1  READ cusVar1    WRITE setCusVar1);
....
....
}
```

9.4.5 Interaktion zwischen Custom Widget und Dialog - manueller Datenaustausch

Neben dem automatischen Datenaustausch ist auch ein manueller Datenaustausch möglich. Der Austausch erfolgt dynamisch, d. h. zur Laufzeit des Dialogs. Folgende Aktionen sind möglich:

- Properties des Custom Widgets können gelesen und geschrieben werden.
- Methoden des Custom Widgets können von der Run MyScreens Projektierung aus aufgerufen werden.
- Auf ein bestimmtes Signal des Custom Widgets kann reagiert und damit Unterprogramme (SUB) in der Run MyScreens Projektierung aufgerufen werden.

9.4.5.1 Properties lesen und schreiben

Beschreibung

Um Properties des Custom Widgets zu lesen und zu schreiben stehen die Funktionen ReadCWProperties und WriteCWProperties in der Run MyScreens Projektierung zur Verfügung.

Programmierung

Syntax:	**ReadCWProperty**("Variablenname", "Propertyname")	
Beschreibung:	Property eines Custom Widgets lesen	
Parameter:	Variablenname	Name der Dialogvariable, der ein Custom Widget zugeordnet ist
	Propertyname	Name des zu lesenden Properties des Custom Widgets
Rückgabewert:	Aktueller Wert des Custom Widget Properties	

Syntax:	**WriteCWProperty**("Variablenname", "Propertyname", "Wert")	
Beschreibung:	Property eines CustomWidgets schreiben	
Parameter:	Variablenname	Name der Dialogvariable, der ein Custom Widget zugeordnet ist
	Propertyname	Name des zu schreibenden Properties des Custom Widgets
	Wert	Wert, der in das Property des CustomWidgets geschrieben werden soll

Beispiele

Beispiel 1:

Lesen des Properties "MyStringVar" des Custom Widgets, das mit der Dialogvariablen "MyCWVar1" verbunden ist und Zuweisung des Wertes in das Register 7.

CustomWidget Klassendeklaration:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(QString MyStringVar
              READ myStringVar
              WRITE setMyStringVar);
   ...
```

Dialogprojektierung:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   REG[7]=ReadCWProperty("MyCWVar1", "MyStringVar")
END_PRESS
```

Beispiel 2:

Schreiben des Ergebnisses aus der Berechnung "3 + sin(123.456)" in das Property "MyRealVar" des CustomWidgets, das mit der Dialogvariablen "MyCWVar1" verbunden ist.

CustomWidget Klassendeklaration:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(double MyRealVar
              READ myRealVar
              WRITE setMyRealVar);
```

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
   ...
```

Dialogprojektierung:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   WriteCWProperty("MyCWVar1", "MyRealVar", 3 + sin(123.456))
END_PRESS
```

9.4.5.2 Ausführen einer Methode des Custom Widgets

Beschreibung

Um Methoden des Custom Widgets auszuführen, steht die Funktion CallCWMethod in der Run MyScreens Projektierung zur Verfügung.

Dabei darf die aufzurufende Custom Widget-Methode maximal 10 Übergabeparameter haben.

Folgende Datenformate der Übergabeparameter werden unterstützt:

- bool
- uint
- int
- double
- QString
- QByteArray

Programmierung

Syntax:	**CallCWMethod**("Variablenname", "Methodenname[, Argument 0][, Argument 1 ... [,Argument 9]")
Beschreibung:	CustomWidget Methode aufrufen

Parameter:	Variablenname	Name der Dialogvariable, der ein Custom Widget zugeordnet ist
	Methodenname	Name der aufzurufenden Custom Widget-Methode
	Argument 0 - 9	Übergabeparameter für die CustomWidget-Methode Unterstützte Datenformate: siehe oben **Hinweis:** Die Übergabeparameter werden immer "ByVal" übergeben, das heißt es wird immer nur der Wert übergeben und nicht z. B. die Referenz auf eine Variable.
Rückgabewert:	Rückgabewert der Custom Widget-Methode Folgende Datenformate der Übergabeparameter werden unterstützt: • void • bool • uint • int • double • QString • QByteArray **Hinweis:** Selbst wenn das Datenformat des Rückgabewertes der Custom Widget-Methode "void" ist, muss dieser formell z. B. einer Variable zugewiesen werden.	

Beispiel

CustomWidget Klassendeklaration:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   public slots:
       void myFunc1(int nValue, const QString& szString, double dValue);
...
```

Dialogprojektierung:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
DEF MyStringVar1 = (S)
DEF MyRealVar = (R)

PRESS(VS3)
   REG[9] = CallCWMethod("MyCWVar1", "myFunc1", 1+7, MyStringVar1, sin(MyRealVar) –
8)
END_PRESS
```

Hinweis

Das Custom Widget muss die Methode "serialize" implementieren. In dieser haben Sie die Möglichkeit, die internen Daten eines Custom Widgets
in eine vorgegebene Datei zu schreiben oder daraus wieder herzustellen. Dies ist vor allem notwendig, wenn bei geöffneter "Run MyScreens"-Maske in einen anderen Bedienbereich gewechselt und anschließend wieder zurückgekehrt wird. Andernfalls gehen die internen Daten beim Wiedereinblenden verloren.

Syntax:	public slots: bool **serialize**(const QString& szFilePath, bool bIsStoring);	
Beschreibung:	Lesen bzw. schreiben von internen Daten und Zuständen aus bzw. in eine Datei	
Parameter:	szFilePath	Name der Datei mit vollständiger Pfadangabe, in die die internen Daten und Zustände des Custom Widgets geschrieben oder aus der sie gelesen werden sollen. Die Datei muss das Custom Widget ggf. selbst anlegen.
	bIsStoring	TRUE = schreiben FALSE = lesen

Beispiel

```
bool SlEsTestCustomWidget::serialize(const QString& szFilePath, bool bIsStoring)
{
   QFileInfo fi(szFilePath);
   bool bReturn = false;
   QDir dir;
   if (dir.mkpath(fi.canonicalPath()))
   {
      QFile fileData(szFilePath);
      QIODevice::OpenMode mode;

      if (bIsStoring)
      {
         mode = QIODevice::WriteOnly;
      }
      else
      {
         mode = QIODevice::ReadOnly;
      }

      if (fileData.open(mode))
      {
         QDataStream streamData;
         streamData.setDevice(&fileData);

         if (bIsStoring)
         {
            streamData << m_nDataCount << m_dValueX;
```

```
bool SlEsTestCustomWidget::serialize(const QString& szFilePath, bool bIsStoring)
          }
          else
          {
             streamData >> m_nDataCount >> m_dValueX;
          }

          streamData.setDevice(0);
             fileData.flush();
             fileData.close();
          bReturn = true;
       }
    }
    return bReturn;
}
```

9.4.5.3 Reaktion auf ein Custom Widget-Signal

Beschreibung

In Run MyScreens ist es möglich auf ein bestimmtes Signal (invokeSub()) des Custom Widget zu reagieren und damit ein Unterprogramm (SUB) aufzurufen.

Für die Wertübergabe (Custom Widget-Signal -> SUB) gibt es 10 global Variablen, die sogenannten SIGARG, die in der Projektierung vergleichbar mit den Registern (REG) sind. Darin werden die mit dem Custom Widget-Signal übertragenen Werte abgelegt.

Folgende Datenformate der Übergabeparameter werden unterstützt:

- bool
- uint
- int
- double
- QString
- QByteArray

Programmierung

Aufruf des Unterprogramms:

Syntax:	`void invokeSub(const QString& rszSignalName, const QVariantList& rvntList);`
Beschreibung:	Custom Widget-Signal, mit dem ein Run MyScreens Unterprogramm aufgerufen wird.

Parameter:	rszSignalName	Name des aufzurufenden Run MyScreens Unterprogrammes
	rvntList	QVariantList Array zur Übertragung von Parametern, die in dem globalen Parametern SIGARG abgelegt werden und in der Projektierung zur Verfügung stehen. Maximale Größe: 10 Elemente Unterstützte Datenformate: siehe oben **Hinweis:** Die Übergabeparameter werden immer „ByVal" übergeben, d.h. es wird immer nur der Wert übergeben und nicht z. B. die Referenz auf eine Variable.

Aufzurufendes Unterprogramm:

Syntax:	`SUB(on_<Variablenname>_<Signalname>)`
	`…`
	`END_SUB`	

Beschreibung:	Reaktion auf ein Custom Widget-Signal	
Parameter:	Variablenname	Name der Dialogvariable, der ein Custom Widget zugeordnet ist.
	Signalname	Name des Custom Widget-Signals
	SIGARG 0 - 9	Übergabeparameter für die Custom Widget-Methode. Unterstützte Datenformate: siehe oben **Hinweis:** Die Übergabeparameter werden immer „ByVal" übergeben, d.h. es wird immer nur der Wert übergeben und nicht z. B. die Referenz auf eine Variable.

Beispiel

Custom Widget Klassendeklaration:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
   Q_OBJECT
signals:
   void invokeSub(const QString& szSubName, const QVariantList& vntList);
…
}
```

CustomWidget Klasse:

```

QVariantList vntList;
vntList << 123.456;
emit invokeSub("MySub", vntList);
```

Dialogprojektierung:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
SUB(on_MyCWVar1_MySub)
```

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
   DEBUG("SUB(on_MyCWVar1_MySub) was called with parameter: """ << SIGARG[0] <<
""""")
END_SUB
```

Ergebnis "easyscreen_log.txt":

```
[10:22:40.445] DEBUG: SUB(on_MyCWVar1_MySub) was called with parameter: "123.456"
```

9.5 SlEsGraphCustomWidget

9.5.1 SlEsGraphCustomWidget

Allgemein

Mit Hilfe von SlEsGraphCustomWidget können Sie geometrische Objekte (Punkt, Linie, Rechteck, Rechteck mit abgerundeten Ecken, Ellipse, Kreisbogen, Text) und Kurven aus Stützpunkten (z. B. Messwerte, Verlauf) darstellen.

Die Objekte werden in einer oder mehreren sogenannten Konturen organisiert. Diese können dann einzeln oder auch kombiniert zur Anzeige gebracht oder geleert, angewählt und gelöscht werden.

Die Objekte werden mit Hilfe von Funktionen der aktuell angewählten Kontur hinzugefügt und auch in dieser Reihenfolge gezeichnet. Möchten Sie Objekte mit einer bestimmten Farbe zeichnen, so müssen Sie vor dem Hinzufügen die entsprechende Zeichenfarbe setzen. Alle nachfolgend hinzugefügten Objekte werden dann mit dieser Zeichenfarbe gezeichnet. Neben der Zeichenfarbe können Sie auch die Zeichenbreite und den Zeichenstil beeinflussen. Bei den geschlossenen Objekten Rechteck, Rechteck mit abgerundeten Ecken und Ellipse können Sie vor dem Hinzufügen der Objekte eine Füllfarbe setzen.

Jede Kontur kann in verschiedene Modi gebracht werden:

- Normale Anzeige von allen Objekttypen.
- Punkte können zu einer sogenannten Polyline verbunden und so als eine Kurve/Graph dargestellt werden.
- Die Fläche zwischen Punkten, Linien oder Kreisbögen bis zur X-Achse kann mit der jeweils aktuellen Füllfarbe eingefärbt werden. Diese Möglichkeit können Sie zum Beispiel zur Visualisierung von Restmaterial beim Drehen nutzen.
  Werden in diesem Modus Punkte zusätzlich als Polyline angezeigt, wird die gesamte Fläche unterhalb der Kurve bis zur X-Achse ausgefüllt. Die Punkte, Linien und Kreisbögen können sich in allen vier Quadranten befinden.

Mit einem speziellen Cursor-Modi können Sie eine oder mehrere Konturen "ablaufen". Der Cursor wird dafür auf das erste oder letzte Punkt-Objekt einer Kontur gesetzt oder ausgehend von der aktuellen Cursorposition innerhalb der Kontur ein Punkt-Objekt vorwärts oder zurück bewegt.

Nach Bedarf kann eine zweite Y-Achse (rechts) mit eigener Skalierung eingeblendet werden. Diese ist durch einen Offset und einem Faktor an die erste Y-Achse (links) gekoppelt.

Mittels Suchfunktion kann anhand einer vorgegebenen X-Koordinate ein vorher einer Kontur hinzugefügter Punkt gefunden und falls gewünscht der Cursor darauf gesetzt werden.

Konturen können Sie auch als Ringpuffer mit einstellbarer Größe konfigurieren.

Durch die Serialisierung kann der aktuelle Zustand des SlEsGraphCustomWidget in binärer Form in einer Datei abgelegt und auch wiederhergestellt werden.

Das SlEsGraphCustomWidget kann über die Geste "Pan" (verschieben der View) und die Gesten "Pinch"/"Spread" (zoomen in/aus der View) bedient werden.

Per Mausbedienung kann die View verschoben (linke Maustaste + Move) und gezoomt (Mausrad) werden.

Die Anzeige wird aus Performance-Gründen nicht automatisch aktualisiert. Je nach Anwendungsfall können Sie durch Aufruf der entsprechenden Funktion die Aktualisierung anstoßen.

Beispiel

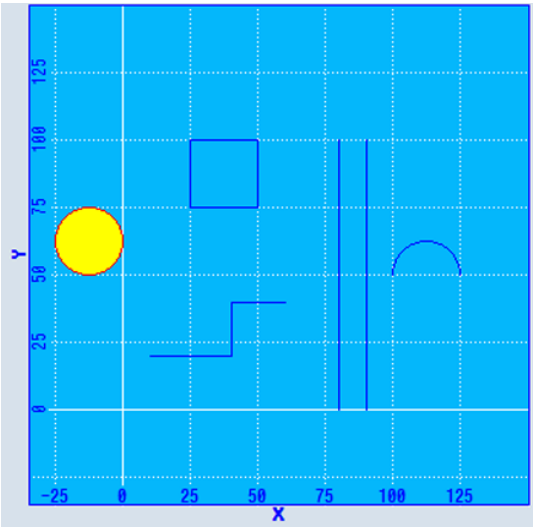


Bild 9-1 SlEsGraphCustomWidget - Beispiel

```
//M(MyGraphSampleMask/"SlEsGraphCustomWidget Sample")
DEF MyGraphVar = (W///,"slesgraphcustomwidget.SlEsGraphCustomWidget"/////10,10,340,340/0,0,0,0)
VS1=("Add objects",,se1)
LOAD
   WRITECWPROPERTY("MyGraphVar", "AxisNameX", "X")
   WRITECWPROPERTY("MyGraphVar", "AxisNameY", "Y")
   WRITECWPROPERTY("MyGraphVar", "ScaleTextOrientationYAxis", 2)
   WRITECWPROPERTY("MyGraphVar", "KeepAspectRatio", TRUE)

   REG[0]= CALLCWMETHOD("MyGraphVar", "addContour", "MyContour", TRUE)
   REG[0]= CALLCWMETHOD("MyGraphVar", "showContour", "MyContour")
```

```
//M(MyGraphSampleMask/"SlEsGraphCustomWidget Sample")
   REG[0]= CALLCWMETHOD("MyGraphVar", "setView", -35, -35, 150, 150)
END_LOAD

PRESS(VS1)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 10, 20, 40, 20)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 40, 20, 40, 40)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 40, 40, 60, 40)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 80, 0, 80, 100)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 90, 0, 90, 100)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addRect", 25, 100, 50, 75)
   REG[0]= CALLCWMETHOD("MyGraphVar", "setPenColor", "#ff0000");red
   REG[0]= CALLCWMETHOD("MyGraphVar", "setFillColor", "#ffff00");yellow
   REG[0]= CALLCWMETHOD("MyGraphVar", "addCircle", -12.5, 62.5, 12.5)
   REG[0]= CALLCWMETHOD("MyGraphVar", "setFillColor");off/default
   REG[0]= CALLCWMETHOD("MyGraphVar", "setPenColor");off/default
   REG[0]= CALLCWMETHOD("MyGraphVar", "addArc", 100, 37.5, 125, 62.5, 0, 180)
   REG[0]= CALLCWMETHOD("MyGraphVar", "update")
END_PRESS
```

9.5.2 Hinweise zur Performance

Abhängig von der eingesetzten Hardware und Grundauslastung des Systems müssen Sie bei Verwendung des SlEsGraphCustomWidgets die folgenden Richtwerte beachten. Neben der eingesetzten Hardware und Grundauslastung variieren die Werte abhängig von der Projektierung der SIEsGraphCustomWidgets.

Beachten Sie diese Richtwerte um die Reaktionsfähigkeit und Stabilität des Gesamtsystems nicht zu beeinträchtigen.

- Anzahl der zu einem Zeitpunkt gleichzeitig projektierten SlEsGraphCustomWidgets (z. B. maximal 1)
- Anzahl der projektierten Konturen (z. B. maximal 6)
- Anzahl der projektierten Grafikobjekte je Kontur (z. B. maximal 1.000)
- Häufigkeit der angeforderten Aktualisierungen (z. B. maximal 500 ms)

Hinweis

Die Anzeige ist nicht echtzeitfähig.

9.5.3 Properties lesen und schreiben

Beschreibung

Die im folgenden Kapitel aufgeführten Properties werden mit der Funktion ReadCWProperty() gelesen und mit WriteCWProperty() gesetzt.

Beispiele

Lesen des Properties "CursorX" des durch die Anzeigevariable "MyGraphVar" verbundenen SlEsGraphCustomWidget. Das Ergebnis wird in das Register 0 geschrieben.

```
REG[0] = ReadCWProperty("MyGraphVar", "CursorX")
```

Schreiben des Wertes "MyFirstContour" in das Property "SelectedContour" des durch die Anzeigevariable "MyGraphVar" verbundenen SlEsGraphCustomWidget.

```
WriteCWProperty("MyGraphVar ", "SelectedContour", "MyFirstContour")
```

9.5.4 Properties

Übersicht

Property	Beschreibung
AxisNameX	Achsbezeichnungen X-Achse
AxisNameY	Achsbezeichnungen Y-Achse
AxisNameY2	Achsbezeichnungen zweite Y-Achse (rechts)
AxisY2Visible	zweite Y-Achse (rechts) ein/ausblenden
AxisY2Offset	Offset zweite Y-Achse (rechts)
AxisY2Factor	Faktor zweite Y-Achse (rechts)
ScaleTextEmbedded	Positionierung der Skalierungstexte
ScaleTextOrientationYAxis	Ausrichtung der Skalierungstexte der Y-Achse
KeepAspectRatio	Seitenverhältnis beibehalten
SelectedContour	Name der aktuell angewählte Kontur
BackColor	Hintergrundfarbe
ChartBackColor	Hintergrundfarbe der Zeichenfläche
ForeColor	Vordergrundfarbe für Texte und Default-Zeichenfarbe
ForeColorY2	Vordergrundfarbe für die Texte der zweiten Y-Achse (rechts)
GridColor	Farbe der Grid-Linien
GridColorY2	Farbe der horizontalen Grid-Linien der zweiten Y-Achse (rechts)
CursorColor	Farbe des Cursor-Fadenkreuzes
ShowCursor	Cursor ein-/ausblenden
CursorX	Cursor X-Position
CursorY	Cursor Y-Position

Property	Beschreibung
CursorY2	Cursor Y-Position bezogen auf zweite Y-Achse (rechts)
CursorStyle	Darstellungsart des Cursors
ViewMoveZoomMode	Verhalten beim Zoomen und Verschieben durch Gesten

AxisNameX – Achsbezeichnungen X-Achse

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisNameX**") WriteCWProperty(GraphVarName, "**AxisNameX**", Value)	
Beschreibung:	Liest bzw. setzt den Bezeichner der X-Achse. Wird kein Bezeichner angegeben, so wird der zur Verfügung stehende Platz für die Zeichenfläche ausgenutzt.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (QString)
	Value	zu setzender Wert (QString)

AxisNameY – Achsbezeichnungen Y-Achse

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisNameY**") WriteCWProperty(GraphVarName, "**AxisNameY**", Value)	
Beschreibung:	Liest bzw. setzt den Bezeichner der Y-Achse. Wird kein Bezeichner angegeben, so wird der zur Verfügung stehende Platz für die Zeichenfläche ausgenutzt.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (QString)
	Value	zu setzender Wert (QString)

AxisY2Visible – zweite Y-Achse (rechts) ein-/ausblenden

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Visible**") WriteCWProperty(GraphVarName, "**AxisY2Visible**", Value)	
Beschreibung:	Zweite Y-Achse (rechts) ein-/ausblenden	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE oder FALSE

AxisY2Offset – Offset zweite Y-Achse (rechts)

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Offset**") WriteCWProperty(GraphVarName, "**AxisY2Offset**", Value)
Beschreibung:	Offset der zweiten Y-Achse (rechts) bezogen auf die erste Y-Achse (links). Siehe Beispiel zu Property AxisY2Factor.

Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (double)
	Value	zu setzender Wert (double)

AxisY2Factor – Faktor zweite Y-Achse (rechts)

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Factor**") WriteCWProperty(GraphVarName, "**AxisY2Factor**", Value)	
Beschreibung:	Faktor der zweiten Y-Achse (rechts) bezogen auf die erste Y-Achse (links).	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (double)
	Value	zu setzender Wert (double)

Zusammen mit dem Property "AxisY2Offset" ist es möglich eine zweite Y-Achse (rechts) mit einer eigenen Skalierung einzublenden. Die Skala wird durch Offset und Faktor an die erste Y-Achse (links) gekoppelt.

Formel für Umrechnung Y2 nach Y:

Y = Y2 / Faktor - Offset

Formel für Umrechnung Y nach Y2:

Y2 = (Y + Offset) * Faktor

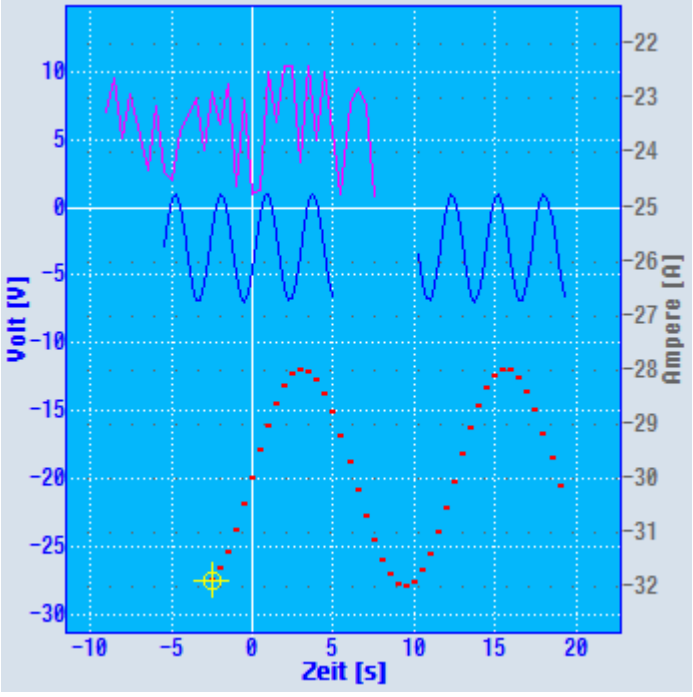


Bild 9-2 Beispiel mit zweiter Y-Achse mit eigener Skalierung

Beispiel

Y: 0 bis 200 soll Y2: -25 bis 25 entsprechen.

- Offset: -100
- Faktor: 0.25

Beispielrechnung:

Y2 = -30

Y = -30 / 0.25 – (-100) = -20

Beispielrechnung:

Y = 0

Y2 = (0 + (-100)) * 0.25 = -25

Die X-Achse ist für beide Koordinatensysteme gleich.

Die Farben können Sie mit setForeColorY2() und setGridColorY2() setzen (siehe Farben).

Die Beschriftung der Achsen wird mit setAxisNameY2() gesetzt (siehe Achsbezeichnungen).

ScaleTextEmbedded - Positionierung der Skalierungstexte

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**ScaleTextEmbedded**") WriteCWProperty(GraphVarName, "**ScaleTextEmbedded**", Value)	
Beschreibung:	Mit dieser Property legen Sie fest, ob Skalierungstexte innerhalb oder außerhalb der Zeichenfläche positioniert werden.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE oder FALSE

Beispiel

Skalierungstexte außerhalb des Zeichenbereiches:

ScaleTextEmbedded = FALSE

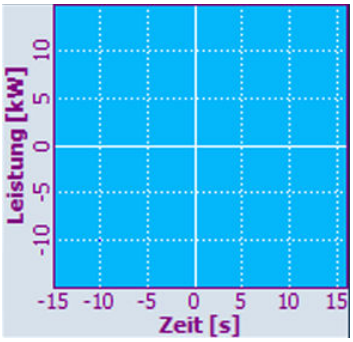


Bild 9-3 Skalierungstexte außerhalb des Zeichenbereiches

Skalierungstexte innerhalb des Zeichenbereiches:

ScaleTextEmbedded = TRUE

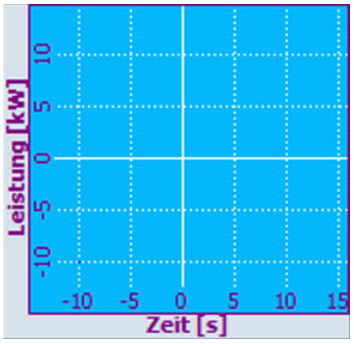


Bild 9-4 Skalierungstexte innerhalb des Zeichenbereiches

ScaleTextOrientationYAxis - Ausrichtung der Skalierungstexte der Y-Achse

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**ScaleTextOrientationYAxis**") WriteCWProperty(GraphVarName, "**ScaleTextOrientationYAxis**", Value)	
Beschreibung:	Mit dieser Funktion werden die Skalierungstexte der Y-Achse vertikal ausgerichtet.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (int)
	Value	zu setzender Wert (int): 1 (= horizontal) oder 2 (= vertikal)

Beispiel

Skalierungstexte innerhalb des Zeichenbereiches:

- ScaleTextEmbedded = TRUE

Textausrichtung horizontal:

- ScaleTextOrientationYAxis = 1

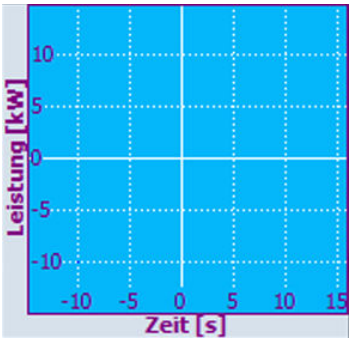


Bild 9-5 Skalierungstexte innerhalb des Zeichenbereiches, horizontale Textausrichtung

Textausrichtung vertikal:

- ScaleTextOrientationYAxis = 2

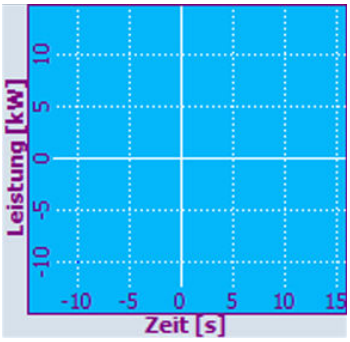


Bild 9-6 Skalierungstexte innerhalb des Zeichenbereiches, vertikale Textausrichtung

Skalierungstexte außerhalb des Zeichenbereiches:

- ScaleTextEmbedded = FALSE

Textausrichtung horizontal:

- ScaleTextOrientationYAxis = 1

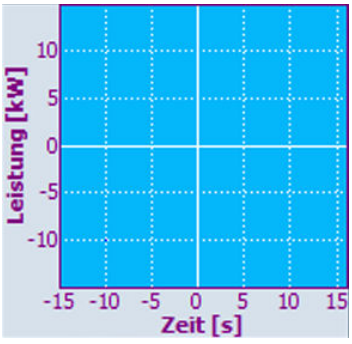


Bild 9-7 Skalierungstexte außerhalb des Zeichenbereiches, horizontale Textausrichtung

Textausrichtung vertikal:

- ScaleTextOrientationYAxis = 2

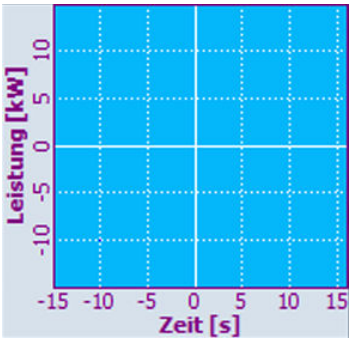


Bild 9-8 Skalierungstexte außerhalb des Zeichenbereiches, vertikale Textausrichtung

KeepAspectRatio – Seitenverhältnis beibehalten

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**KeepAspectRatio**") WriteCWProperty(GraphVarName, "**KeepAspectRatio**", Value)	
Beschreibung:	Mit dieser Property legen Sie fest, ob die mit setView() festgelegte View automatisch so ausgerichtet, angepasst oder erweitert werden soll, dass das Seitenverhältnis von X- zu Y-Achse immer gleich bleibt. Relevant ist diese Eigenschaft insbesondere bei der Anzeige von geometrischen Figuren. Damit wird z. B. ein Kreis oder Quadrat auch als solches und nicht zu einer Ellipse oder Rechteck verzerrt dargestellt.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (bool)
	Return Value	zu setzender Wert (bool): TRUE oder FALSE

Beispiel

```
setView(-15, -15, 15, 15)
setFillColor("#A0A0A4")
addCircle(0, 0, 5)
KeepAspectRatio = TRUE
```

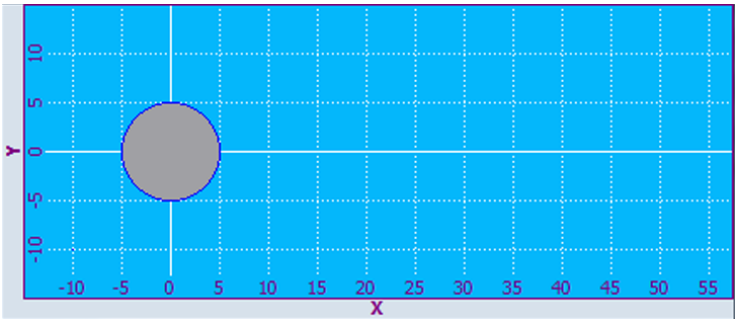


Bild 9-9 Seitenverhältnis von X- zu Y-Achse bleibt gleich

```
KeepAspectRatio = FALSE
```

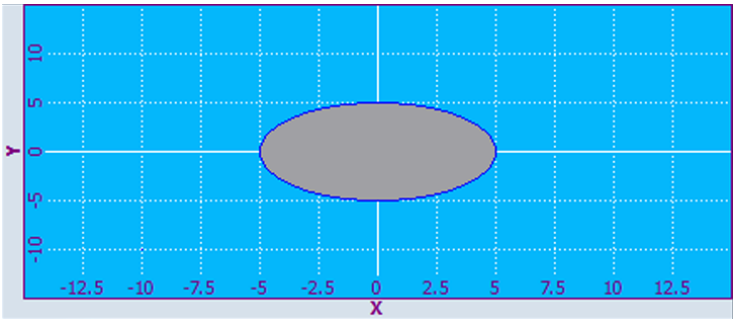


Bild 9-10 Seitenverhältnis von X- zu Y-Achse verzerrt

SelectedContour – Name der aktuell angewählte Kontur

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, " **SelectedContour**") WriteCWProperty(GraphVarName, " **SelectedContour**", Value)	
Beschreibung:	Liest bzw. setzt die Auswahl der aktuellen Kontur.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (QString)
	Value	zu setzender Wert (QString)

Hinweis

Um Operationen auf eine Kontur ausführen zu können, z. B. um Grafikobjekte hinzuzufügen, müssen Sie diese vorher anwählen.

BackColor – Hintergrundfarbe

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, " **BackColor**") WriteCWProperty(GraphVarName, " **BackColor**", Value)	
Beschreibung:	Hintergrundfarbe für Achsbeschriftung, abhängig von der Property ScaleTextInsideChartArea auch der Skalierungstexte.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Farbe des Properties (QString)
	Value	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

ChartBackColor – Hintergrundfarbe der Zeichenfläche

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**ChartBackColor**") WriteCWProperty(GraphVarName, "**ChartBackColor**", Value)
Beschreibung:	Hintergrundfarbe für den Zeichenbereich.

Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesene Farbe des Properties (QString)
	Value	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

ForeColor - Vordergrundfarbe für die Beschriftung und Default-Zeichenfarbe

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**ForeColor**") WriteCWProperty(GraphVarName, "**ForeColor**", Value)	
Beschreibung:	Vordergrundfarbe für Texte und Default-Zeichenfarbe.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Farbe des Properties (QString)
	Value	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

ForeColorY2 - Vordergrundfarbe für die Beschriftung der zweiten Y-Achse (rechts)

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**ForeColorY2**") WriteCWProperty(GraphVarName, "**ForeColorY2**", Value)	
Beschreibung:	Vordergrundfarbe für Texte der zweiten Y-Achse (rechts).	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Farbe des Properties (QString)
	Value	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

GridColor - Farbe der Gridlinien

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**GridColor**") WriteCWProperty(GraphVarName, "**GridColor**", Value)	
Beschreibung:	Farbe der Gridlinien.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Farbe des Properties (QString)
	Value	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

GridColorY2 - Farbe der horizontalen Gridlinien der zweiten Y-Achse (rechts)

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**GridColorY2**") WriteCWProperty(GraphVarName, "**GridColorY2**", Value)
Beschreibung:	Farbe der horizontalen Gridlinien der zweiten Y-Achse (rechts).

Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Farbe des Properties (QString)
	Value	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

CursorColor - Farbe des Cursors

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorColor**") WriteCWProperty(GraphVarName, "**CursorColor**", Value)	
Beschreibung:	Farbe des Cursors.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Farbe des Properties (QString)
	Value	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

ShowCursor – Cursor ein-/ausblenden

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, " **ShowCursor**") WriteCWProperty(GraphVarName, " **ShowCursor**", Value)	
Beschreibung:	Cursor ein-/ausblenden.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE oder FALSE

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

CursorX – Cursor X-Position

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorX**") WriteCWProperty(GraphVarName, "**CursorX**", Value)	
Beschreibung:	X-Position des Cursors.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (double)
	Value	zu setzender Wert (double)

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

CursorY – Cursor Y-Position

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorY**") WriteCWProperty(GraphVarName, "**CursorY**", Value)	
Beschreibung:	Y-Position des Cursors.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (double)
	Value	zu setzender Wert (double)

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

CursorY2 – Cursor Y-Position bezogen auf zweite Y-Achse (rechts)

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorY2**") WriteCWProperty(GraphVarName, "**CursorY2**", Value)	
Beschreibung:	Y-Position des Cursors bezogen auf die zweite Y-Achse (rechts).	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (double)
	Value	zu setzender Wert (double)

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

CursorStyle – Darstellungsart des Cursors

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorStyle**") WriteCWProperty(GraphVarName, "**CursorStyle**", Value)	
Beschreibung:	Darstellungsart des Cursors.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (int)
	Value	zu setzender Wert (int): 0 (= Kreuz) 1 (= Fadenkreuz) 2 (= vertikale Linie) 3 (= horizontale Linie) 4 (= horizontale und vertikale Linie))

ViewMoveZoomMode – Verhalten beim Zoomen und Verschieben durch Gesten

Syntax:	ReturnValue = ReadCWProperty(GraphVarName, "**ViewMoveZoomMode**") WriteCWProperty(GraphVarName, "**ViewMoveZoomMode**", Value)	
Beschreibung:	Durch Gesten kann die angezeigte View des SlEsGraphCustomWidgets verändert werden. Mit dieser Property legen Sie fest, auf welche Art dies erlaubt werden soll. Zum Beispiel kann es in bestimmten Anwendungsfällen sinnvoll sein, dass die View nur horizontal verschoben und gezoomt werden darf, z. B. bei der Anzeige des Verlaufes von Messwerten.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	gelesener Wert des Properties (int)
	Value	zu setzender Wert (int): 0 (= aus) 1 (= nur horizontal) 2 (= nur vertikal) 3 (= horizontal und vertikal)

9.5.5 Funktionen

Die nachfolgend aufgeführten Funktionen können Sie mit der Funktion CallCWMethod() aufrufen.

Beispiel

Setzen der View des durch die Anzeigevariable "MyGraphVar" verbundenen SlEsGraphCustomWidget.

Das Ergebnis wird in das Register 0 geschrieben.

```
REG[0] = CallCWMethod("MyGraphVar", "setView", -8, -5, 11- 20)
```

Übersicht

Funktion	Beschreibung
setView	Koordinatensystem setzen
setMaxContourObjects	Maximale Anzahl Objekte einer Kontur setzen (Ringpuffer)
addContour	Kontur hinzufügen
showContour	Kontur anzeigen
hideContour	Kontur unsichtbar schalten
hideAllContours	Alle Konturen unsichtbar schalten
removeContour	Kontur entfernen
clearContour	Grafikobjekte einer Kontur löschen
fitViewToContours	Automatische Anpassung der View
fitViewToContour	Automatische Anpassung der View
findX	Suchen in einer Kontur

Funktion	Beschreibung
setPolylineMode	Punkte eine Kontur als Polyline/Kurve darstellen
setIntegralFillMode	Punkte eine Kontur als Polyline/Kurve darstellen
repaint, update	Ansicht aktualisieren
addPoint	Punkt einer Kontur hinzufügen
addLine	Linie einer Kontur hinzufügen
addRect	Rechteck einer Kontur hinzufügen
addRoundedRect	Rechteck mit abgerundeten Ecken einer Kontur hinzufügen
addEllipse	Ellipse einer Kontur hinzufügen
addCircle	Kreis einer Kontur hinzufügen
addArc	Kreisbogen einer Kontur hinzufügen
addText	Text einer Kontur hinzufügen
setPenWidth	Zeichenbreite setzen
setPenStyle	Zeichenstil setzen
setPenColor	Zeichenfarbe setzen
setFillColor	Füllfarbe setzen
setCursorPosition	Cursor positionieren
setCursorPositionY2	Cursor bezogen auf die zweite Y-Achse (rechts) positionieren
setCursorOnContour	Cursor auf Kontur positionieren
moveCursorOnContourBegin	Cursor auf das erste Grafikobjekt einer Kontur positionieren
moveCursorOnContourEnd	Cursor auf das letzte Grafikobjekt einer Kontur positionieren
moveCursorOnContourNext	Cursor auf das nächste Grafikobjekt einer Kontur positionieren
serialize	Speichern/Wiederherstellen des aktuellen Zustandes

setView – Koordinatensystem setzen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setView**", x1, y1, x2, y2)	
Beschreibung:	Mit dieser Funktion definieren Sie die Größe des Koordinatensystems, das innerhalb des SlEsGraphCustomWidget angezeigt werden soll. Siehe hierzu auch Property KeepAspectRatio.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x1	linker Rand (double)
	y1	oberer Rand (double)
	x2	rechter Rand (double)
	y2	unterer Rand (double)

Hinweis

Die Funktion aktualisiert automatisch die Anzeige.

Beispiel

```
setView(-8, -5, 11, 20)
AxisNameX = "X"
AxisNameY = "Y"
ScaleTextOrientationYAxis = 1
ScaleTextEmbedded = false
KeepAspectRatio = false
update
```

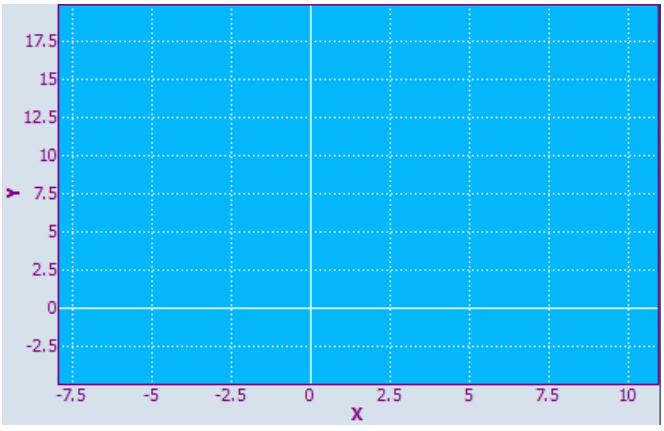


Bild 9-11 Beispiel - setView

setMaxContourObjects – Maximale Anzahl Objekte einer Kontur setzen (Ringpuffer)

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setMaxContourObjects**", Value) ReturnValue = CallCWMethod(GraphVarName, "**setMaxContourObjects**", Value, ContourName)	
Beschreibung:	Mit dieser Funktion legen Sie die Größe des Ringpuffers einer Kontur fest. Überschreitet die Anzahl der der Kontur hinzugefügten Objekte diese Grenze, so wird dafür das älteste Objekt gelöscht.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	Value	Maximale Anzahl an Grafikobjekten (uint)
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

Hinweis

Die Funktion aktualisiert automatisch die Anzeige.

addContour – Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName) ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName, Selected) ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName, Selected, AssignedY2)	
Beschreibung:	Mit dieser Funktion legen Sie eine neue Kontur an. Optional können Sie die Kontur dem Koordinatensystem der zweiten Y-Achse (rechts) zuordnen.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.
	Selected	Kontur soll angewählt werden (bool): TRUE oder FALSE
	AssignedY2	Kontur soll der zweiten Y-Achse (rechts) zugeordnet werden (bool): TRUE oder FALSE

showContour – Kontur anzeigen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**showContour**") ReturnValue = CallCWMethod(GraphVarName, "**showContour**", ContourName)	
Beschreibung:	Diese Funktion setzt eine Kontur sichtbar.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

hideContour – Kontur unsichtbar schalten

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**hideContour**") ReturnValue = CallCWMethod(GraphVarName, "**hideContour**", ContourName)	
Beschreibung:	Diese Funktion setzt eine Kontur unsichtbar.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

showAllContours – Alle Konturen sichtbar schalten

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**showAllContours**")	
Beschreibung:	Diese Funktion setzt alle Konturen sichtbar.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich

hideAllContours – Alle Konturen unsichtbar schalten

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**hideAllContours**")	
Beschreibung:	Diese Funktion setzt alle Konturen unsichtbar.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich

removeContour – Kontur unsichtbar schalten

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**removeContour**") ReturnValue = CallCWMethod(GraphVarName, "**removeContour**", ContourName)	
Beschreibung:	Entfernt eine Kontur.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

clearContour – Grafikobjekte einer Kontur löschen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**clearContour**") ReturnValue = CallCWMethod(GraphVarName, "**clearContour**", ContourName)	
Beschreibung:	Löscht alle enthaltenen Grafikobjekte einer Kontur.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

fitViewToContours – Automatische Anpassung der View

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContours**") ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContours**", OnlyVisible)	
Beschreibung:	Mit dieser Funktion wird die View automatisch an die Konturen angepasst. Abhängig vom Übergabeparameter legen Sie fest, ob entweder nur alle sichtbaren Konturen oder alle Konturen sichtbar sind.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	OnlyVisible	zu setzender Wert (bool): TRUE oder FALSE

fitViewToContour – Automatische Anpassung der View

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContour**", ContourName)	
Beschreibung:	Mit dieser Funktion wird die View automatisch so angepasst, dass nur eine bestimmte Kontur sichtbar ist.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString)

findX – Suchen in einer Kontur

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**findX**", x) ReturnValue = CallCWMethod(GraphVarName, "**findX**", x, ContourName) ReturnValue = CallCWMethod(GraphVarName, "**findX**", x, ContourName, SetCursor)	
Beschreibung:	Mit diesen Funktionen können Sie in einer Kontur nach einem vorher definierten Punkt mit einer bestimmten X-Koordinate suchen. Als Ergebnis erhalten Sie die Y-Koordinate. Optional können Sie auf diesen Punkt auch gleich den Cursor positionieren.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (QString): bei Erfolg wird der zugehörige Y-Wert geliefert
	x	Zu suchender X-Wert (double)
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur
	SetCursor	Cursor setzen (bool): TRUE oder FALSE

Hinweis

Wenn Sie mit dieser Funktion den Cursor setzen, wird die Anzeige automatisch aktualisiert.

setPolylineMode – Punkte einer Kontur als Polyline/Kurve darstellen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setPolylineMode**", SetPoly)	
Beschreibung:	Alle nach diesem Funktionsaufruf hinzugefügten Punkte der aktuellen Kontur werden zu einem Polyline/Kurve verbunden. Diese Funktion ist konturspezifisch und kann abschnittsweise ein- und ausgeschaltet werden.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	SetPoly	PolylineMode (bool): TRUE = ein

Beispiel

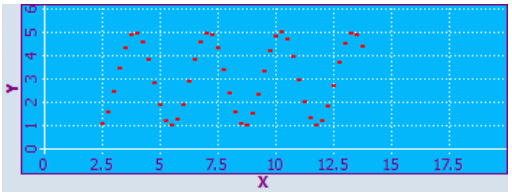


Bild 9-12 Beispiel - PolylineMode: aus

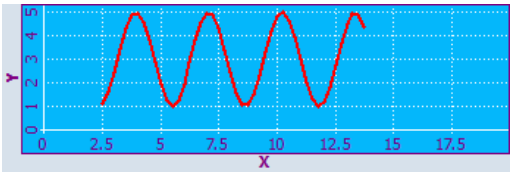


Bild 9-13 Beispiel - PolylineMode: an

setIntegralFillMode – Füllen der Flächen zwischen Punkten, Linien und Kreisbögen und der X-Achse

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setIntegralFillMode**", SetIntFill)	
Beschreibung:	Flächen zwischen Punkten, Linien und Kreisbögen und der X-Achse werden mit der aktuellen Füllfarbe gefüllt (unabhängig davon, ob die Punkte +Y oder –Y sind). Diese Funktion ist konturspezifisch und kann abschnittsweise ein- und ausgeschaltet werden.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	SetIntFill	IntegralFillMode (bool): TRUE = ein

Beispiel

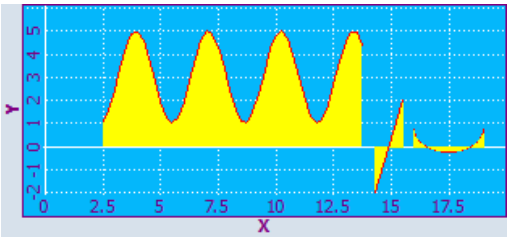


Bild 9-14 Beispiel - setIntegralFillMode: ein

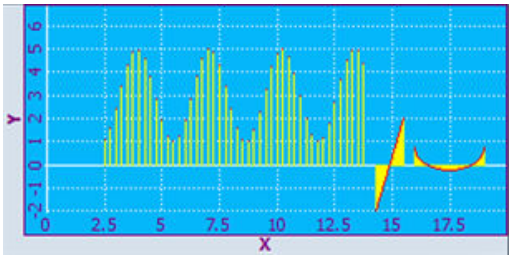


Bild 9-15 Beispiel - setIntegralFillMode: aus

repaint, update – Ansicht aktualisieren

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**repaint**") ReturnValue = CallCWMethod(GraphVarName, "**update**")	
Beschreibung:	Mit diesen Funktionen können Sie manuell die Anzeige aktualisieren.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich

- update()
  Die Funktion aktualisiert die Ansicht, vorausgesetzt dass das Aktualisieren des Widget nicht deaktiviert und das Widget nicht verborgen ist. Die Funktion ist so optimiert, dass die Performance der Anwendung erhalten bleibt und die Ansicht durch zu häufiges Aktualisieren nicht flackert.
- repaint()
  Die Funktion aktualisiert die Ansicht unmittelbar nach dem Funktionsaufruf. Sie sollten die Funktion repaint deshalb nur nutzen, wenn eine sofortige Aktualisierung zwingend notwendig ist, z. B. bei Animationen.

Es wird empfohlen immer mit der Funktion update() zu arbeiten. Intern arbeitet das SlEsGraphCustomWidget immer mit der Funktion update() z.B. bei setView().

addPoint – Punkt einer Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addPoint**", x, y) ReturnValue = CallCWMethod(GraphVarName, "**addPoint**", x, y, DummyPoint)	
Beschreibung:	Einen Punkt der aktuell selektierten Kontur hinzufügen. Darüber hinaus können Sie einen Dummy-Punkt setzen, der zwar nicht gezeichnet wird, aber dafür mit dem Cursor positioniert werden kann.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x	x-Koordinate (double)
	y	y-Koordinate (double)
	DummyPoint	Punkt wird als unsichtbar behandelt (bool): TRUE = ja

addLine – Linie einer Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addLine**", x1, y1, x2, y2)	
Beschreibung:	Eine Linie der aktuell selektierten Kontur hinzufügen	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x1	x-Koordinate des Startpunktes (double)
	y1	y-Koordinate des Startpunktes (double)
	x2	x-Koordinate des Endpunktes (double)
	y2	y-Koordinate des Endpunktes (double)

addRect – Rechteck einer Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addRect**", x1, y1, x2, y2)	
Beschreibung:	Ein Rechteck der aktuell selektierten Kontur hinzufügen.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x1	linker Rand (double)
	y1	linker Rand (double)
	x2	rechter Rand (double)
	y2	unterer Rand (double)

addRoundedRect – Rechteck mit abgerundeten Ecken einer Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addRoundedRect**", x1, y1, x2, y2, r)
Beschreibung:	Ein Rechteck mit abgerundeten Ecken der aktuell selektierten Kontur hinzufügen.

Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x1	linker Rand (double)
	y1	oberer Rand (double)
	x2	oberer Rand (double)
	y2	oberer Rand (double)
	r	Radius (double)

addEllipse – Ellipse einer Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addEllipse**", x1, y1, x2, y2, radius)	
Beschreibung:	Eine Ellipse der aktuell selektierten Kontur hinzufügen.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x1	linker Rand (double)
	y1	oberer Rand (double)
	x2	rechter Rand (double)
	y2	unterer Rand (double)
	radius	Radius (double)

addCircle – Kreis einer Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addCircle**", x, y, radius)	
Beschreibung:	Einen Kreis der aktuell selektierten Kontur hinzufügen.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x	linker Rand (double)
	y	oberer Rand (double)
	radius	Radius (double)

addArc – Kreisbogen einer Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addArc**", x1, y1, x2, y2, StartAngle, SpanAngle
Beschreibung:	Einen Kreisbogen der aktuell selektierten Kontur hinzufügen.

Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x1	linker Rand (double)
	y1	oberer Rand (double)
	x2	rechter Rand (double)
	y2	unterer Rand (double)
	StartAngle	Anfangswinkel in Grad (double)
	SpanAngle	Öffnungswinkel in Grad (double)

addText – Text einer Kontur hinzufügen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**addText**", x, y, Text)	
Beschreibung:	Einen Text der aktuell selektierten Kontur hinzufügen.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x	linker Rand (double)
	y	oberer Rand (double)
	Text	Text (QString)

setPenWidth – Zeichenbreite setzen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setPenWidth**") ReturnValue = CallCWMethod(GraphVarName, "**setPenWidth**", width)	
Beschreibung:	Legt die Zeichenbreite ab dem Zeitpunkt des Funktionsaufrufes für die nachfolgend hinzugefügten Objekte der aktuellen Kontur fest.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	width	Zeichenbreite (double). Wird keine Zeichenbreite vorgegeben, so wird die Default-Zeichenbreite gesetzt.

setPenStyle – Zeichenstil setzen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setPenStyle**") ReturnValue = CallCWMethod(GraphVarName, "**setPenStyle**", style)
Beschreibung:	Legt den Zeichenstil ab dem Zeitpunkt des Funktionsaufrufes für die nachfolgend hinzugefügten Objekte der aktuellen Kontur fest.

Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	style	1 : durchgezogene Linie (___) 2 : unterbrochene Linie (_ _) 3 : gepunktete Linie (...) 4 : Linie-Punkt-Linie (_._) 5 : Linie-Punkt-Punkt (_..)

setPenColor – Zeichenfarbe setzen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setPenColor**") ReturnValue = CallCWMethod(GraphVarName, "**setPenColor**", Color)	
Beschreibung:	Legt die Zeichenfarbe ab dem Zeitpunkt des Funktionsaufrufes für die nachfolgend hinzugefügten Objekte der aktuellen Kontur fest.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	Color	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

setFillColor – Füllfarbe setzen

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setFillColor**") ReturnValue = CallCWMethod(GraphVarName, "**setFillColor**", Color)	
Beschreibung:	Legt die Füllfarbe ab dem Zeitpunkt des Funktionsaufrufes für die nachfolgend hinzugefügten Objekte der aktuellen Kontur fest.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	Color	zu setzender Wert (QString) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

setCursorPosition – Cursor positionieren

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorPosition**", x, y)	
Beschreibung:	Setzt den Cursor an die vorgegebene Position.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x	x-Koordinate (double)
	y	y-Koordinate (double)

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

setCursorPositionY2Cursor – Cursor bezogen auf die zweite Y-Achse (rechts) positionieren

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorPositionY2**", x, y2)	
Beschreibung:	Setzt den Cursor an die vorgegebene Position bezogen auf die zweite Y-Achse (rechts).	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	x	x-Koordinate (double)
	y2	Y-Koordinate bezogen auf die zweite Y-Achse (rechts) (double)

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

setCursorOnContour – Cursor auf Kontur positionieren

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorOnContour**") ReturnValue = CallCWMethod(GraphVarName, "**setCursorOnContour**", ContourName)	
Beschreibung:	Setzt den Cursor auf die letzte Cursorposition innerhalb einer Kontur.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

Beispiel

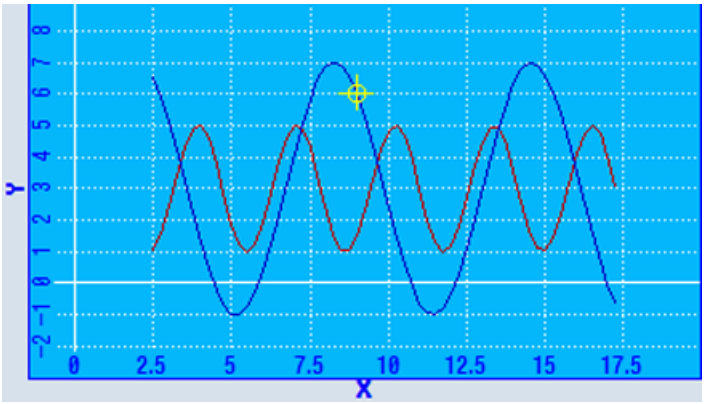


Bild 9-16 Beispiel - setCursorOnContour

moveCursorOnContourBegin – Cursor auf das erste Grafikobjekt einer Kontur positionieren

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBegin**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBegin**", ContourName)	
Beschreibung:	Ausgehend von der aktuellen Cursorposition innerhalb einer Kontur, können Sie mit dieser Funktion den Cursor auf das erste Punkt-Objekt einer Kontur navigieren.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

moveCursorOnContourEnd – Cursor auf das letzte Grafikobjekt einer Kontur positionieren

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourEnd**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourEnd**", ContourName)
Beschreibung:	Ausgehend von der aktuellen Cursorposition innerhalb einer Kontur, können Sie mit dieser Funktion den Cursor auf das letzte Punkt-Objekt einer Kontur navigieren.

Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

moveCursorOnContourNext – Cursor auf das nächste Grafikobjekt einer Kontur positionieren

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourNext**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourNext**", ContourName)	
Beschreibung:	Ausgehend von der aktuellen Cursorposition innerhalb einer Kontur, können Sie mit dieser Funktion den Cursor auf das nächste Punkt-Objekt einer Kontur navigieren.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

moveCursorOnContourBack – Cursor auf das vorherige Grafikobjekt einer Kontur positionieren

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBack**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBack**", ContourName)	
Beschreibung:	Ausgehend von der aktuellen Cursorposition innerhalb einer Kontur, können Sie mit dieser Funktion den Cursor auf das vorherige Punkt-Objekt einer Kontur navigieren.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	ContourName	Name der Kontur (QString). Wird keine Kontur angegeben, bezieht sich der Aufruf automatisch auf die aktuell angewählte Kontur.

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

serialize – Speichern/Wiederherstellen des aktuellen Zustandes

Syntax:	ReturnValue = CallCWMethod(GraphVarName, "**serialize**", FilePath, IsStoring)	
Beschreibung:	Mit dieser Funktion können Sie bei Bedarf den aktuellen Zustand und Inhalt des SlEsGraphWidget binär in einer Datei oder DataStream schreiben und auch wiederherstellen.	
Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	FilePath	Vollständiger Pfad mit Filename (QString)
	IsStoring	Speichern/Wiederherstellen (bool): TRUE = Speichern

Hinweis

Diese Funktion aktualisiert automatisch die Anzeige.

9.5.6 Signale

Das nachfolgend beschriebene Signal ViewChanged können Sie in der Projektierung abfangen und entsprechend darauf reagieren.

Übersicht

Funktion	Beschreibung
ViewChanged	Änderung der View

ViewChanged – Änderung der View

Syntax:	SUB(on_<GraphVarName>**_ViewChanged**) ... END_SUB
Beschreibung:	Ändert sich die View, so wird das Signal "ViewChanged" gesendet. Auf dieses können Sie in der Projektierung in einer bestimmten SUB-Methode reagieren. Die SIGARG-Parameter werden entsprechend gesetzt.

Parameter:	GraphVarName	Name der Anzeigevariable die ein SlEsGraphCustomWidget enthält
	SIGARG[0]	linker Rand (double)
	SIGARG[1]	oberer Rand (double)
	SIGARG[2]	rechter Rand (double)
	SIGARG[3]	unterer Rand (double)

Beispiel

```
DEF MyGraphVar = (W///,"slesgraphcustomwidget.SlEsGraphCustomWidget"/////
10,10,340,340/0,0,0,0)
```

```
SUB(on_MyGraphVar_ViewChanged
     DLGL("Current view: " << SIGARG[0] << ", " << SIGARG[1] << ", " <<   SIGARG[2]
<< ", " << SIGARG[3]
END_SUB
```

9.6 SlEsTouchButton

9.6.1 SlEsTouchButton

Allgemein

Mit Run MyScreens erstellen Sie auf einfache Art und Weise auch funktional anspruchsvolle Applikationen. Insbesondere für die Touch-Bedienung können Sie frei platzierbare Buttons (TouchButtons) projektieren. Für die Projektierung der TouchButtons gelten die folgenden Eigenschaften:

- Die beiden möglichen Zustandsänderungen des Buttons "clickt" und "checkt" können in der Projektierung abgefangen und entsprechende Aktionen ausgelöst werden.
- Die TouchButton können per Single- oder Multi-Touch, Maus und Tastatur bedient werden.
- Der TouchButton wird entsprechend der aktuellen Auflösung dargestellt. Dies gilt unter anderem auch für den Font und die Anzeigebilder.
- Der TouchButton besitzt zwei Darstellungsstile, "Softkey Look&Fee" und "Anwenderspezifisch". Im Darstellungsstil "Softkey Look&Feel" ist die Darstellung der TouchButton entsprechend der Operate Softkeys. Für beide Darstellungsstile stehen Funktionen wie zum Beispiel das Skalieren von Bildern zur Verfügung.
- TouchButton werden als CustomWidget realisiert und zur Verfügung gestellt.

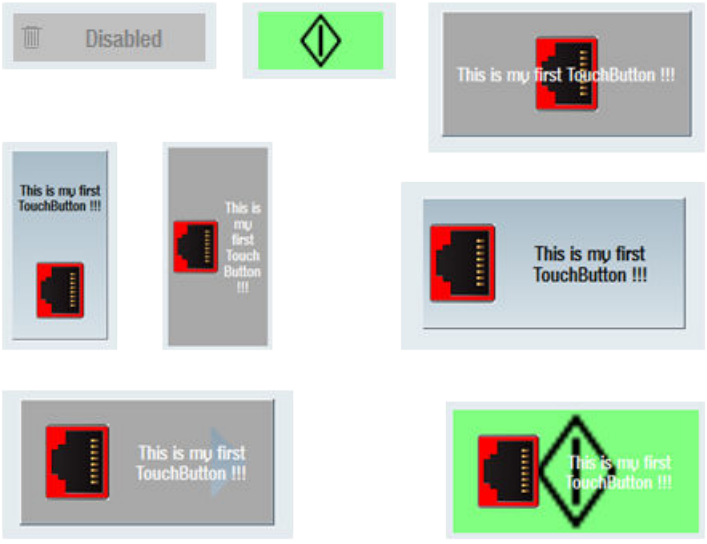


Bild 9-17 Beispiele für TouchButton

Beispiel

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
LOAD
   WRITECWPROPERTY("MyTB1", "text", "This is my first TouchButton !!!")
   WRITECWPROPERTY("MyTB1", "textPressed", "This is my first TouchButton (pressed)!!!")
   WRITECWPROPERTY("MyTB1", "picture", "dsm_remove_trashcan_red.png") WRITECWPROPERTY("MyTB1", "pic-
tureAlignment", "left")
   WRITECWPROPERTY("MyTB1", "scalePicture", FALSE)
   WRITECWPROPERTY("MyTB1", "picturePressed", "slsu_topology_empty_round_slot.png") WRITECWPROPER-
TY("MyTB1", "picture", "slsu_topology_empty_slot_left_error.png")
END_LOAD
```

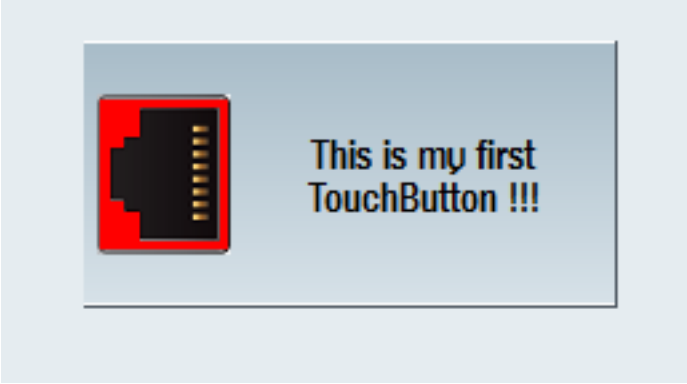


Bild 9-18 Beispiel "This is my first TouchButton !!!"

9.6.2 Properties lesen und schreiben

Beschreibung

Die im folgenden Kapitel aufgeführten Properties werden mit der Funktion ReadCWProperty() gelesen und mit WriteCWProperty() gesetzt.

Beispiele

Lesen des Properties "Text" des durch die Anzeigevariable "MyTouchButton" verbundenen SlEsTouchButton. Das Ergebnis wird in das Register 0 geschrieben.

```
REG[0] = ReadCWProperty("MyTouchButton", "Text")
```

Schreiben des Werts "sk_ok.png" in das Property "Picture" des durch die Anzeigevariable "MyTouchButton" verbundenen SlEsTouchButton.

```
WriteCWProperty("MyTouchButton", "Picture", "sk_ok.png")
```

9.6.3 Properties

Übersicht

Property	Beschreibung
ButtonStyle	Darstellungsstil des TouchButton
Flat	Darstellung flach oder 3D
Enabled	Bedienbarkeit ein/ausschalten
Checkable	Toggle-Funktion ein/ausschalten
Checked	TouchButton ist gerade eingerastet (checked)
ShowFocusRect	Fokus-Rechteck anzeigen, wenn der TouchButton den Eingabefokus hat
Picture	Anzuzeigendes Bild, wenn der TouchButton im normalen nicht gedrückten Zustand ist
PicturePressed	Anzuzeigendes Bild, wenn der TouchButton gedrückt oder eingerastet (checked) dargestellt wird
PictureAlignment PictureAlignmentString	Ausrichtung des Bildes
ScalePicture	Bild wird skaliert (gestreckt oder gestaucht)
PictureKeepAspectRatio	Streckungsverhalten des Bildes
Text	Normaler Anzeigetext
TextPressed	Anzeigetext, wenn der TouchButton gedrückt ist
textAlignment textAlignmentString	Ausrichtung des Texts
TextAlignedToPicture	Text wird relativ zum Bild ausgerichtet - ein/ausschalten

Property	Beschreibung
BackgroundPicture	Anzuzeigendes Hintergrundbild
BackgroundPictureAlignment BackgroundPictureAlignmentString	Ausrichtung des Hintergrundbildes
ScaleBackgroundPicture	Hintergrundbild wird skaliert (gestreckt oder gestaucht)
BackgroundPictureKeepAspectRatio	Streckungsverhalten des Bildes
BackColor	Hintergrundfarbe, wenn der TouchButton im normalen und nichtgedrückten Zustand ist
BackColorChecked	Hintergrundfarbe, wenn der TouchButton eingerastet ist (checked)
BackColorPressed	Hintergrundfarbe, wenn der TouchButton momentan gedrückt wird
BackColorDisabled	Hintergrundfarbe, wenn der TouchButton nicht bedienbar ist
TextColor	Textfarbe, wenn der TouchButton im normalen und nichtgedrückten Zustand ist
TextColorChecked	Textfarbe, wenn der TouchButton eingerastet ist (checked)
TextColorPressed	Textfarbe, wenn der TouchButton momentan gedrückt wird
TextColorDisabled	Textfarbe, wenn der TouchButton nicht bedienbar ist

ButtonStyle – Darstellungsstil

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ButtonStyle**") WriteCWProperty(TouchButtonVarName, "**ButtonStyle**", Value)	
Beschreibung:	Liest/setzt den Datstellungsstil des TouchButtons.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (int)
	Value	zu setzender Wert (int) 0 = Softkey Look&Feel (default) 1 = Anwenderspezifisch

Mit der Einstellung "0 = Softkey Look&Feel" präsentiert und verhält sich der TouchButton genauso wie ein Softkey. Dabei wird auch der aktuell eingestellte Skin berücksichtigt – siehe Anzeigemaschinendatum 9112 = $MM_HMI_SKIN.

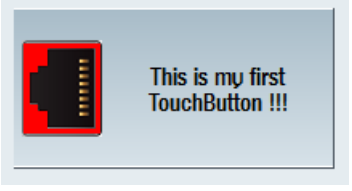


Bild 9-19 ButtonStyle - Softkey Look&Feel

Mit der Einstellung "1 = Anwenderspezifisch" kann die Text- und Hintergrundfarbe je nach Zustand des TouchButton definiert werden. Darüber hinaus ist ein 3D-Effekt, das automatische Skalieren von Bildern und die Beeinflussung der Abstände von den Rändern möglich.

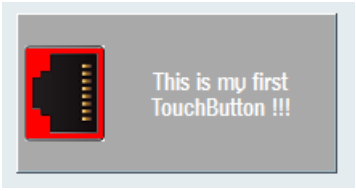


Bild 9-20 ButtonStyle - Anwenderspezifisch

Flat – Darstellungsart

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Flat**") WriteCWProperty(TouchButtonVarName, "**Flat**", Value)	
Beschreibung:	Liest/setzt, ob der TouchButton flach (default) oder 3D dargestellt wird **Hinweis:** Diese Property steht nur mit aktivem Darstellungsstil (ButtonStyle) "1= Anwenderspezifisch" zur Verfügung.	
	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchBut-ton enthält
Parameter:	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE (default) oder FALSE

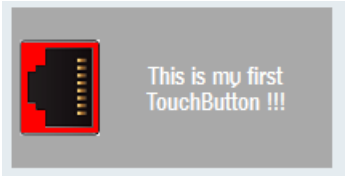


Bild 9-21 Darstellungsart flach

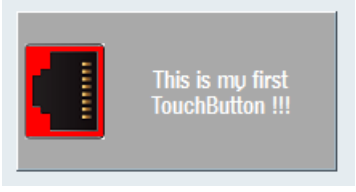


Bild 9-22 Darstellungsart 3D

Enabled – Bedienbarkeit des TouchButtons

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Enabled**") WriteCWProperty(TouchButtonVarName, "**Enabled**", Value)	
Beschreibung:	Liest/setzt, ob der TouchButton bedienbar (= TRUE) oder nicht bedienbar (FALSE) sein soll.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE (default) oder FALSE

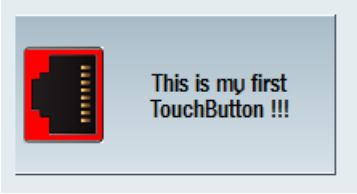


Bild 9-23 TouchButton bedienbar

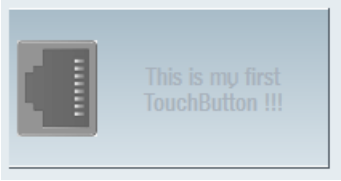


Bild 9-24 TouchButton nicht bedienbar

Hinweis

Wird auf einen nicht bedienbaren TouchButtons geklickt, so wird das Signal "clickedDisabled" gesendet. Das Signal kann ausgewertet werden und z. B. eine entsprechenden Meldung, weshalb der TouchButton und die damit verbundene Funktion im Moment nicht bedienbar ist, ausgegeben werden. Das Anzeigebild oder Hintergrundbild wird im deaktivierten Zustand automatisch in Graustufen dargestellt.

Checkable – Toggle-Funktion ein/ausschalten

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Checkable**") WriteCWProperty(TouchButtonVarName, "**Checkable**", Value)	
Beschreibung:	Liest/setzt die Toggle-Funktionalität des TouchButtons.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE oder FALSE (default)

Hinweis

Standardmäßig wechselt der TouchButton nach dem Loslassen wieder in seinen Normalzustand (vgl. Taster). Ist hingegen die Toggle-Funktionalität des TouchButtons eingeschaltet (TRUE), dann bleibt dieser nach dem Drücken zunächst in der gedrückten Position. Wird er erneut betätigt, dann wechselt er wieder in seinen Normalzustand (vgl. Schalter).

Siehe auch Property "Checked".

Checked – Aktueller Toggle-Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Checked**") WriteCWProperty(TouchButtonVarName, "**Checked**", Value)	
Beschreibung:	Liest/setzt den aktuellen Toggle-Zustand des TouchButtons.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE oder FALSE (default)

Ist die Toggle-Funktion durch das Property "Checkable" auf "TRUE" aktiviert, so kann mit dem Property "Checked" der aktuelle Toggle-Zustand gelesen oder gesetzt werden.

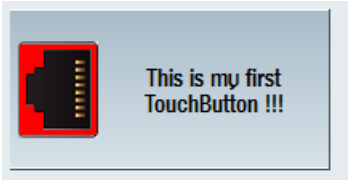


Bild 9-25 Unchecked

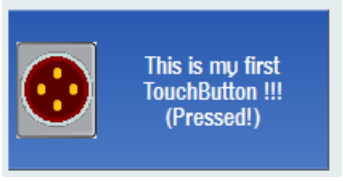


Bild 9-26 Checked

Für beide Zustände können Text und Bild spezifisch angezeigt werden.

Hinweis

Siehe auch Property "Checkable".

ShowFocusRect – Fokus-Rechteck Darstellung

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ShowFocusRect**") WriteCWProperty(TouchButtonVarName, "**ShowFocusRect**", Value)	
Beschreibung:	Liest/setzt, ob der TouchButton ein Fokus-Rechteck darstellen soll, sobald er den Eingabefokus erhält. .	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE oder FALSE (default)

Die Farbe des Fokusrahmens wird automatisch mit der Operate-Farbeinstellung gesetzt, im folgenden Beispiel "orange".

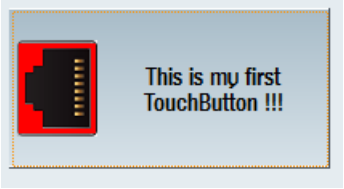


Bild 9-27 ShowFocusRect

Picture – Anzeigebild

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Picture**") WriteCWProperty(TouchButtonVarName, "**Picture**", Value)	
Beschreibung:	Liest/setzt das anzuzeigende Bild, wenn der TouchButton in seiner Ruhestellung (nicht gedrückt) ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (String)
	Value	zu setzender Wert (String)

Als Wert wird der Dateiname angegeben, zum Beispiel "sk_ok.png". Der TouchButton ermittelt automatisch die korrekte Bilddatei aus dem aktuellen Auflösungsverzeichnis (siehe Kapitel Bilder/Grafik verwenden (Seite 63)).

Das Anzeigebild wird im deaktivierten Zustand automatisch in Graustufen dargestellt.

Hinweis

Siehe auch Properties "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PicturePressed – Anzeigebild im gedrückten Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PicturePressed**") WriteCWProperty(TouchButtonVarName, "**PicturePressed**", Value)	
Beschreibung:	Liest/setzt das anzuzeigende Bild, wenn der TouchButton im gedrücktem oder eingerastetem Zustand ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (String)
	Value	zu setzender Wert (String)

Wird kein Wert angegeben (Leerstring " "), so zeigt der TouchButton in diesem Zustand das mit dem Property "Picture" vorgegebene Bild an.

Als Wert wird der Dateiname angegeben, zum Beispiel "sk_ok.png". Der TouchButton ermittelt automatisch die korrekte Bilddatei aus dem aktuellen Auflösungsverzeichnis (siehe Kapitel Bilder/Grafik verwenden (Seite 63)).

Das Anzeigebild wird im deaktivierten Zustand automatisch in Graustufen dargestellt.

Hinweis

Siehe auch Properties "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PictureAlignment – Ausrichtung des Bildes

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureAlignment**") WriteCWProperty(TouchButtonVarName, "**PictureAlignment**", Value)	
Beschreibung:	Liest/setzt die Ausrichtung des Anzeigebildes auf dem TouchButton.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (int)
	Value	zu setzender Wert (int): 1 = Links (default) 2 = Rechts 32 = Oben 64 = Unten 128 = Zentriert

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).
Siehe auch Properties "Text", "TextAlignedToPicture", "PictureAlignmentString", "ScalePicture", "PictureKeepAspectRatio".

PictureAlignmentString – Ausrichtung des Bildes

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureAlignmentString**") WriteCWProperty(TouchButtonVarName, "**PictureAlignmentString**", Value)	
Beschreibung:	Liest/setzt die Ausrichtung des Anzeigebildes. Dises Property ist gleichbedeutend mit dem Property "PictureAlignment". Der Wert wird hier jedoch nicht als Zahl (int), sondern als String übergeben.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (String)
	Value	zu setzender Wert (String): "Left" = Links (default) "Right" = Rechts "Top" = Oben "Bottom" = Unten "Center" = Zentriert

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

Siehe auch Properties "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

ScalePicture – Anzeigebild skalieren

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ScalePicture**") WriteCWProperty(TouchButtonVarName, "**ScalePicture**", Value)	
Beschreibung:	Liest/setzt, ob der TouchButton je nach Ausrichtung das anzuzeigende Bild skalieren soll.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE oder FALSE (default)

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

PictureKeepAspectRatio – Anzeigebild im Seitenverhältnis skalieren

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureKeepAspectRatio**") WriteCWProperty(TouchButtonVarName, "**PictureKeepAspectRatio**", Value)	
Beschreibung:	Liest/setzt, ob beim Skalieren (Property "ScalePicture" = "TRUE") das Seitenverhältnis des Anzeigebild beibehalten werden soll oder nicht.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE (default) oder FALSE

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

Text – Anzeigetext

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Text**") WriteCWProperty(TouchButtonVarName, "**Text**", Value)
Beschreibung:	Liest/setzt den Anzeigetext, wenn der TouchButton in seiner Ruhestellung (nicht gedrückt) ist.

Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (String)
	Value	zu setzender Wert (String)

Die Texte werden automatisch an Wortgrenzen im zur Verfügung stehenden Rechteck umgebrochen.

Zeilenumbrüche können systembedingt nur erzwungen werden, wenn der Anzeigetext aus einer Sprachdatei kommt.

Hinweis

Siehe Kapitel Sprachabhängige Texte (Seite 270).

TextPressed – Anzeigetext im gedrückten Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextPressed**") WriteCWProperty(TouchButtonVarName, "**TextPressed**", Value)	
Beschreibung:	Liest/setzt den Anzeigetext, wenn der TouchButton im gedrücktem oder eingerastetem Zustand ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (String)
	Value	zu setzender Wert (String)

Die Texte werden automatisch an Wortgrenzen im zur Verfügung stehenden Rechteck umgebrochen.

Zeilenumbrüche können systembedingt nur erzwungen werden, wenn der Anzeigetext aus einer Sprachdatei kommt.

Hinweis

Siehe Kapitel Sprachabhängige Texte (Seite 270).

TextAlignment – Ausrichtung des Texts

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextAlignment**") WriteCWProperty(TouchButtonVarName, "**TextAlignment**", Value)
Beschreibung:	Liest/setzt die Ausrichtung des Texts auf dem TouchButton.

Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (int)
	Value	zu setzender Wert (int): 1 = Links 2 = Rechts 32 = Oben 64 = Unten 128 = Zentriert (default) (siehe Beispiele unten)

Bild 9-28 TextAlignment - links

Bild 9-29 TextAlignment - rechts

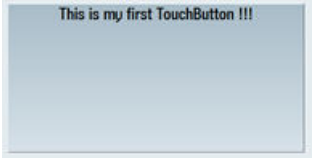


Bild 9-30 TextAlignment - oben

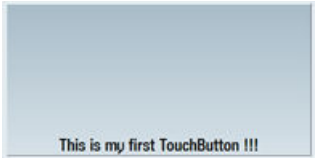


Bild 9-31 TextAlignment - unten

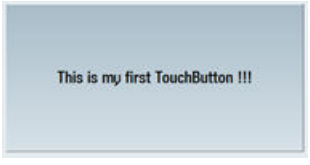


Bild 9-32 TextAlignment - zentriert

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

Siehe auch Properties "Text", "TextAlignmentString", "TextAlignedToPicture".

TextAlignmentString – Ausrichtung des Texts

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextAlignmentString**") WriteCWProperty(TouchButtonVarName, "**TextAlignmentString**", Value)	
Beschreibung:	Liest/setzt die Ausrichtung des Texts. Dises Property ist gleichbedeutend mit dem Property "TextAlignment". Der Wert wird nicht als Zahl (int), sondern als String übergeben.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (String)
	Value	zu setzender Wert (String): "Left" = Links "Right" = Rechts "Top" = Oben "Bottom" = Unten "Center" = Zentriert (default)

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

Siehe auch Properties "Text", "TextAlignment", "TextAlignedToPicture".

TextAlignedToPicture – Textausrichtung relativ zum Bild

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "TextAlignedToPicture") WriteCWProperty(TouchButtonVarName, "**TextAlignedToPicture**", Value)	
Beschreibung:	Liest/setzt, ob der Anzeigetext relativ zum Bild positioniert werden soll. Wird hier FALSE gesetzt, dann wird der Text auf dem TouchButton zentriert dargestellt.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE (default) oder FALSE

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).
Siehe auch Properties "Text", "TextPressed", "Picture", "PicturePressed".

BackColor – Hintergrundfarbe

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, " **BackColor**") WriteCWProperty(TouchButtonVarName, " **BackColor**", Value)	
Beschreibung:	Liest/setzt die Hintergrundfarbe, wenn der TouchButton in seiner Ruhestellung (nicht gedrückt) ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Farbe des Properties (String)
	Value	zu setzender Wert (String) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

Hinweis

Diese Property steht nur mit aktivem Darstellungsstil "1 – Anwenderspezifisch" zur Verfügung.

BackColorChecked – Hintergrundfarbe im eingerasteten Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorChecked**") WriteCWProperty(TouchButtonVarName, " **BackColorChecked**", Value)	
Beschreibung:	Liest/setzt die Hintergrundfarbe, wenn der TouchButton im eingerasteten Zustand (Toggle-Funktion) ist.Siehe Properties "Checked", "Checkable"	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Farbe des Properties (String)
	Value	zu setzender Wert (String) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

Hinweis

Diese Property steht nur mit aktivem Darstellungsstil "1 – Anwenderspezifisch" zur Verfügung.

Hinweis

Siehe auch Properties "Checked", "Checkable".

BackColorPressed – Hintergrundfarbe im gedrückten Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorPressed**") WriteCWProperty(TouchButtonVarName, " **BackColorPressed**", Value)
Beschreibung:	Liest/setzt die Hintergrundfarbe, wenn der TouchButton im gedrückten Zustand ist.

Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Farbe des Properties (String)
	Value	zu setzender Wert (String) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

Hinweis

Diese Property steht nur mit aktivem Darstellungsstil "1 – Anwenderspezifisch" zur Verfügung.

BackColorDisabled – Hintergrundfarbe im deaktivierten Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorDisabled**") WriteCWProperty(TouchButtonVarName, " **BackColorDisabled**", Value)	
Beschreibung:	Liest/setzt die Hintergrundfarbe, wenn der TouchButton im deaktiviertem Zustand ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Farbe des Properties (String)
	Value	zu setzender Wert (String) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

Hinweis

Diese Property steht nur mit aktivem Darstellungsstil "1 – Anwenderspezifisch" zur Verfügung.

Hinweis

Siehe auch Property "Enabled".

TextColor – Textfarbe

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColor**") WriteCWProperty(TouchButtonVarName, " **TextColor**", Value)	
Beschreibung:	Liest/setzt die Textfarbe, wenn der TouchButton in seiner Ruhestellung (nicht gedrückt) ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Farbe des Properties (String)
	Value	zu setzender Wert (String) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

Hinweis

Diese Property steht nur mit aktivem Darstellungsstil "1 – Anwenderspezifisch" zur Verfügung.

TextColorChecked – Textfarbe im eingerasteten Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorChecked**") WriteCWProperty(TouchButtonVarName, " **TextColorChecked**", Value)	
Beschreibung:	Liest/setzt die Textfarbe, wenn der TouchButton im eingerasteten Zustand (Toggle-Funktion) ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Farbe des Properties (String)
	Value	zu setzender Wert (String) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

Hinweis

Diese Property steht nur mit aktivem Darstellungsstil "1 – Anwenderspezifisch" zur Verfügung.

Hinweis

Siehe auch Properties "Checked", "Checkable".

TextColorPressed – Textfarbe im gedrückten Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorPressed**") WriteCWProperty(TouchButtonVarName, "**TextColorPressed**", Value)	
Beschreibung:	Liest/setzt die Textfarbe, wenn der TouchButton im gedrückten Zustand ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Farbe des Properties (String)
	Value	zu setzender Wert (String) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

Hinweis

Diese Property steht nur mit aktivem Darstellungsstil "1 – Anwenderspezifisch" zur Verfügung.

TextColorDisabled – Textfarbe im deaktivierten Zustand

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorDisabled**") WriteCWProperty(TouchButtonVarName, "**TextColorDisabled**", Value)	
Beschreibung:	Liest/setzt die Textfarbe, wenn der TouchButton im deaktiviertem Zustand ist.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Farbe des Properties (String)
	Value	zu setzender Wert (String) als RGB-Wert der Form "#RRGGBB", z. B. "#04B7FB"

Hinweis

Diese Property steht nur mit aktivem Darstellungsstil "1 – Anwenderspezifisch" zur Verfügung.

Hinweis

Siehe auch Property "Enabled".

BackgroundPicture – Hintergrundbild

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPicture**") WriteCWProperty(TouchButtonVarName, "**BackgroundPicture**", Value)	
Beschreibung:	Liest/setzt das anzuzeigende permanente, d.h. zustandsunabhängige Hintergrundbild.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (String)
	Value	zu setzender Wert (String)

Als Wert wird der Dateiname angegeben, zum Beispiel "sk_ok.png". Der TouchButton ermittelt automatisch die korrekte Bilddatei aus dem aktuellen Auflösungsverzeichnis (siehe Kapitel Bilder/Grafik verwenden (Seite 63).).

Das Hintergrundbild wird im deaktivierten Zustand automatisch in Graustufen dargestellt.

Hinweis

Siehe Kapitel Bilder/Grafik verwenden (Seite 63).

Siehe auch Properties "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureAlignment – Ausrichtung des Hintergrundbildes

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureAlignment**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureAlignment**", Value)	
Beschreibung:	Liest/setzt die Ausrichtung des Hintergrundbildes.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (int)
	Value	zu setzender Wert (int): 1 = links 2 = rechts 32 = oben 64 = unten 128 = zentriert (default)

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

Siehe auch Properties "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureAlignmentString – Ausrichtung des Hintergrundbildes

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureAlignmentString**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureAlignmentString**", Value)	
Beschreibung:	Liest/setzt die Ausrichtung des Hintergrundbildes. Dises Property ist gleichbedeutend mit dem Property "BackgroundPictureAlignment". Der Wert wird hier jedoch nicht als Zahl (int), sondern als String übergeben.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (int)
	Value	zu setzender Wert (int): "Left" = links "Right" = rechts "Top" = oben "Bottom" = unten "Center" = zentriert (default)

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

Siehe auch Properties "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

ScaleBackgroundPicture – Hintergrundbild skalieren

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ScalePicture**") WriteCWProperty(TouchButtonVarName, "**ScalePicture**", Value)	
Beschreibung:	Liest/setzt, ob der TouchButton je nach Ausrichtung das anzuzeigende Bild skalieren soll.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE oder FALSE (default)

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

Siehe auch Properties "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureKeepAspectRatio – Hintergrundbild im Seitenverhältnis skalieren

Syntax:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureKeepAspectRatio**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureKeepAspectRatio**", Value)	
Beschreibung:	Liest/setzt, ob beim Skalieren (Property "ScaleBackgroundPicture" = "TRUE") das Seitenverhältnis des Hintergrundbild beibehalten werden soll oder nicht.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	gelesener Wert des Properties (bool)
	Value	zu setzender Wert (bool): TRUE (default) oder FALSE

Hinweis

Siehe Kapitel Positionierung und Ausrichtung von Bild und Text (Seite 268).

9.6.4 Funktionen

Die nachfolgend aufgeführten Funktionen können Sie mit der Funktion CallCWMethod() aufrufen.

Beispiel

Setzen der Abstände des durch die Anzeigevariable "MyTouchButton" verbundenen SlEsTouchButton.

Das Ergebnis wird in das Register 0 geschrieben.

```
REG[0] = CallCWMethod("MyTouchButton", "setMargins", 20, 20, 20, 20,
20)
```

Übersicht

Funktion	Beschreibung
setMargins	Abstände einstellen
serialize	Speichern/Wiederherstellen des aktuellen Zustandes

setMargins – Abstände einstellen

Syntax:	ReturnValue = CallCWMethod(TouchButtonVarName, "**setMargins**", left, top, right, bottom, center) ReturnValue = CallCWMethod(TouchButtonVarName, "**setMargins**", left, top, right, bottom, center, marginAlignment)	
Beschreibung:	Mit dieser Funktion werden die Rahmenabstände und der Abstand zwischen Bild und Text definiert. Wird ein Wert mit "-1" angegeben, so gilt für diesen Wert der System-Default-Abstand. Ist der Wert größer oder gleich "0", so wird der Wert als Randabstand gesetzt.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	left	linker Rand (int)
	top	oberer Rand (int)
	right	rechter Rand (int)
	bottom	unterer Rand (int)
	center	Abstand zwischen Bild und Text (int), wenn Alignment nicht "center" ist
	marginAlignment	optional: bestimmt, ob die Margin neben dem Anzeigebild auch für die Positionierung des Anzeigetexts wirken soll (bool), TRUE (default)

serialize – Speichern/Wiederherstellen des aktuellen Zustandes

Syntax:	ReturnValue = CallCWMethod(TouchButtonVarName, "**serialize**", FilePath, IsStoring)	
Beschreibung:	Mit dieser Funktion kann bei Bedarf der aktuelle Zustand und Inhalt des SlEsTouchButton binär in eine Datei oder DataStream geschrieben und wiederhergestellt werden. Diese Funktion aktualisiert automatisch die Anzeige.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	Return Value	Errorcode (bool): TRUE = erfolgreich
	FilePath	Vollständiger Pfad mit Filename (QString)
	IsStoring	Speichern/Wiederherstellen (bool): TRUE = Speichern

Hinweis

Die Funktion wird nicht direkt durch den Projekteur verwendet!

9.6.5 Signale

Die nachfolgend beschriebenen Signale können Sie in der Projektierung abfangen und entsprechend darauf reagieren.

Übersicht

Funktion	Beschreibung
clickedDisabled	Betätigung eines deaktivierten TouchButtons
clicked	Es wurde ein Click oder Tap ausgeführt
checked	Es wurde getoggelt

- Ist ein TouchButton bedienbar und nicht togglebarer, wird bei Betätigung das Signalen "clicked" gesendet.
- Ist ein TouchButton bedienbar und togglebarer, wird bei Betätigung folgende Sequenz an Signalen gesendet:
  "checked" → "clicked"
- Ist ein Touchbutton nicht bedienbar, wird bei Betätigung das Signal "clickedDisabled" gesendet.

Alle oben genannten Signale werden immer erst nach einer Bedienhandlung gesendet, d. h. erst wenn die Maus oder die Leertaste losgelassen wird oder nachdem ein Tap bei Multitouch ausgeführt wurde.

Dadurch hat der SlEsTouchButton eine reine Schalterfunktionalität, d. h. es ist dadurch keine Tasterfunktionalität realisierbar.

Hinweis

Beachten Sie bei der Multitouch-Bedienung, dass ein Click erst dann gesendet wird, wenn eine Tap-Geste (drücken und loslassen innerhalb von ca. 0,7 Sekunden) vollständig erkannt wurde. Würde bereits beim einfachen Drücken eine Aktion ausgeführt werden, wäre z. B. ein Scrollen/ Verschieben der dahinterliegenden ScrollArea nicht möglich, beziehungsweise könnte es leicht zu ungewollter Fehlbedienungen führen.

clicked – Es wurde ein Click auf einen TouchButton ausgeführt

Syntax:	SUB(on_<TouchButtonVarName>_clicked) ... END_SUB	
Beschreibung:	Eine vollständige Sequenz aus drücken und loslassen eines bedienbaren Touch-Buttons ergibt ein Signal "clicked". Es wird empfohlen vorwiegend mit diesem Signal zu arbeiten.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	SIGARG[0]	liefert den Toggle-Zustand des TouchButtons (bool)

Beispiel

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_clicked)
   DLGL("checked: " << SIGARG[0])
END_SUB
```

checked – Es wurde ein TouchButton getoggelt

Syntax:	SUB(on_<TouchButtonVarName>_checked) ... END_SUB	
Beschreibung:	Ein toggelbarer TouchButton wurde gedrückt, wodurch sich sein Zustand geändert hat.	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	SIGARG[0]	liefert den Toggle-Zustand des TouchButtons (bool)

Beispiel

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_toggled)
   DLGL("toggled: " << SIGARG[0])
END_SUB
```

clickedDisabled – Es wurde ein Click auf einen nicht bedienbaren TouchButton ausgeführt

Syntax:	SUB(on_<TouchButtonVarName>_clickedDisabled) ... END_SUB	
Beschreibung:	Eine vollständige Sequenz aus drücken und loslassen eines nicht bedienbaren TouchButtons ergibt ein Signal "clickedDisabled".	
Parameter:	TouchButtonVarName	Name der Anzeigevariable die einen SlEsTouchButton enthält
	SIGARG[0]	liefert den Toggle-Zustand des TouchButtons (bool)

Beispiel

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_clickedDisabled)
   DLGL("checkedDisabled: " << SIGARG[0])
END_SUB
```

9.6.6 Positionierung und Ausrichtung von Bild und Text

Positionierung von Bildern

Mit der vorgegebenen Ausrichtung (Alignment) wird ein Bild folgendermaßen positioniert:

- Im ersten Schritt wird das zur aktuellen Auflösung passende Bild aus dem zugehörigen Auflösungsverzeichnis ermittelt.
- Das Rechteck der Oberfläche (ClientArea) wird dann um die vorgegebenen Abstände zu den Rändern verkleinert (MarginArea).
  Die MarginArea kann mit der Funktion "setMargins" beeinflußt werden:
  - setMargins(-1, -1, -1, -1, -1)

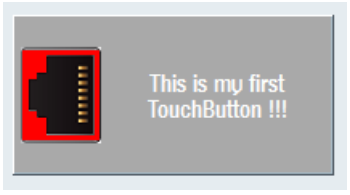


Bild 9-33 setMargins(-1, -1, -1, -1, -1)

  - setMargins(0, 0, 0, 0, 0)

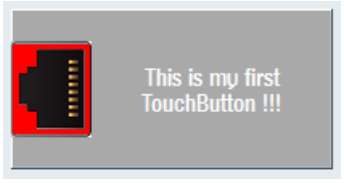


Bild 9-34 setMargins(0, 0, 0, 0, 0)

  - setMargins(20, 20, 20, 20, 20)

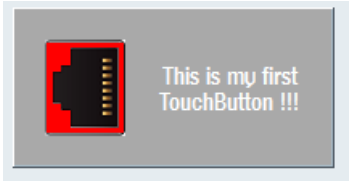


Bild 9-35 setMargins(20, 20, 20, 20, 20)

Beispiel für Ausrichtung "links"

Die Fläche wird horizontal halbiert, so dass das anzuzeigende Bild maximal die Hälfte der Fläche einnehmen kann.

Das Bild wird dann wie folgt gezeichnet:

- Property "scalePicture": FALSE
  Das Bild wird ohne jegliche Skalierung an diese Stelle gezeichnet. Dabei ist darauf zu achten, dass das zugrundeliegende Bild nicht zu groß ist und auch wirklich innerhalb des TouchButtons vollständig dargestellt werden kann.

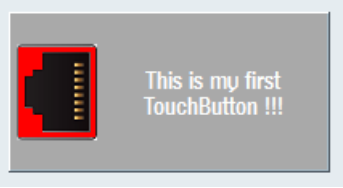


Bild 9-36 scalePicture FALSE

- Property "scalePicture": TRUE
  Das Bild wird soweit skaliert (gestreckt oder gestaucht), dass es in die linke Hälfte des TouchButtons passt. Dabei wird das Property "pictureKeepAspectRatio" wie folgt berücksichtigt:
  - Property "pictureKeepAspectRatio": FALSE
    Das Bild wird horizontal und vertikal soweit skaliert, dass es genau in die linke Hälfte des TouchButtons passt. Dabei wird das Seitenverhältnis des ursprünglichen Bildes nicht weiter beachtet.

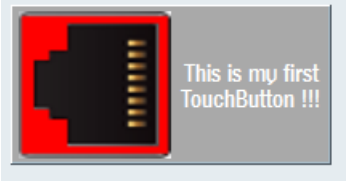


Bild 9-37 scalePicture - pictureKeepAspectRatio FALSE

  - Property "pictureKeepAspectRatio": TRUE
    Das Bild wird unter Berücksichtigung des Seitenverhältnisses des ursprünglichen Bilds so groß wie möglich in die linke Hälfte des TouchButtons skaliert.

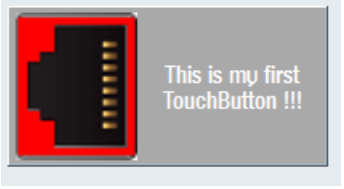


Bild 9-38 scalePicture - pictureKeepAspectRatio TRUE

Hinweis

Für die Ausrichtungen "rechts", "oben" und "unten" gilt das gleiche Prinzip.

Bei der Ausrichtung "zentriert" wird versucht, das Bild abhängig von den Properties "scalePicture" und "pictureKeepAspectRatio" in den kompletten Bereich der MarginArea einzupassen.

Positionierung des Texts

Abhängig vom Property "textAlignedToPicture" wird der Text folgendermaßen positioniert:

- Property "textAlignedToPicture": FALSE
  Der Text wird in der MarginArea vertikal und horizontal zentriert angezeigt.

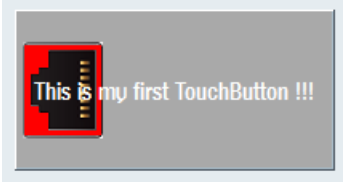


Bild 9-39 textAlignedToPicture FALSE

- Property "textAlignedToPicture": TRUE
  Der Text wird im verbleibenden Bereich der MarginArea abzüglich des Bereichs des gezeichneten Bilds horizontal und vertikal zentriert ausgegeben.

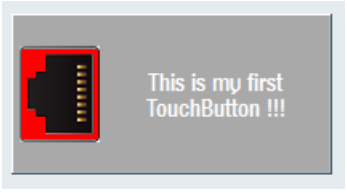


Bild 9-40 textAlignedToPicture TRUE

9.6.7 Sprachabhängige Texte

Der SlEsTouchButton besitzt keine eigentständige Unterstützung für Fremdsprachen. Wie Sie dennoch eine Sprachabhängigkeit realisieren, wird im folgenden Beispiel gezeigt:

Beispiel

easyscreen.ini:

```
[LANGUAGEFILES]LngFile03 = user.txt
```

user_eng.txt:

```
85000 0 0 "This is my first Touchbutton !!!"
```

user_deu.txt:

```
85000 0 0 "Das ist mein erster Touchbutton !!!"
```

Projektierungsdatei:

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
LOAD
```

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
   WRITECWPROPERTY("MyTB1", "text", $85000)
END_LOAD
LANGUAGE
   WRITECWPROPERTY("MyTB1", "text", $85000)
END_LANGUAGE
```

Die Texte werden automatisch an Wortgrenzen im zur Verfügung stehenden Rechteck umgebrochen. Ein erzwungener Zeilenumbruch ist nur möglich, indem der anzuzeigende Text aus einer Sprachdatei stammt. Dabei muss an der entsprechenden Stelle im Text ein "%n" eingefügt werden (siehe folgendes Beispiel).

user_eng.txt:

```
85001 0 0 "This is%nmy%nfirst%nTouchButton !!!"
```

Projektierungsdatei:

```
WRITECWPROPERTY("MyTB1", "text", $85001)
```

Ergebnis

Bild 9-41 Beispiel für Zeilenumbruch in sprachabhängigem Text

Bedienbereich "Custom" 10

10.1 So aktivieren Sie den Bedienbereich "Custom"

Bedienbereich "Custom" aktivieren

Der Bedienbereich "Custom" ist bei Auslieferung nicht aktiviert.

1. Kopieren Sie zuerst die Datei "slamconfig.ini". aus dem Verzeichnis [System siemens-Verzeichnis]/cfg in das Verzeichnis [System oem-Verzeichnis]/cfg
   oder entsprechend in [System addon-Verzeichnis]/cfg oder [System user-Verzeichnis]/cfg.
2. Um den Bedienbereich "Custom" zu aktivieren, ist folgender Eintrag notwendig:
   ```
   [Custom]
   Visible=True
   ```

Ergebnis

Nach dem Aktivieren befindet sich der Softkey für den Bedienbereich "Custom" im Hauptmenü (F10) auf der Menüfortschaltleiste auf dem HSK4 (= Voreinstellung).

Der Bedienbereich "Custom" zeigt ein leeres Fenster über den gesamten Bedienbereich mit einer projektierbaren Überschrift. Alle horizontalen und vertikalen Softkeys sind projektierbar.

10.2 So projektieren Sie den Softkey für "Custom"

Softkey für den Bedienbereich "Custom" projektieren

In der Datei "slamconfig.ini" wird die Beschriftung und die Position des Softkeys für den Bedienbereich "Custom" projektiert.

Für die Projektierung des Einstiegssoftkeys gibt es folgende Möglichkeiten:

1. Um die Beschriftung durch einen **sprachabhängigen Text** auf dem Softkey zu ersetzen, sind im Abschnitt [Custom] folgende Einträge nötig:
   ```
   TextId=MY_TEXT_ID
   TextFile=mytextfile
   TextContext=mycontext
   ```
   In diesem Beispiel zeigt der Softkey den sprachabhängigen Text an, der mit der Text-ID "MY_TEXT_ID" in der Textdatei mytextfile_xxx.qm unter "MyContext" abgelegt wurde (xxx steht für das Sprachkürzel).
2. Um die Beschriftung durch einen **sprachunabhängigen Text** auf dem Softkey zu ersetzen, sind im Abschnitt [Custom] folgende Einträge nötig:
   ```
   TextId=HELLO
   TextFile=<empty>
   TextContext=<empty>
   ```
   In diesem Beispiel zeigt der Softkey für den Bedienbereich "Custom" in jeder Sprache den Text "HELLO" an.
3. Zusätzlich zum Text kann auf dem Softkey auch **ein Piktogramm** angezeigt werden. Dazu ist im Abschnitt [Custom] folgender Eintrag nötig:
   ```
   Picture=mypicture.png
   ```
   Dann zeigt der Softkey die das Piktogramm der Datei mypicture.png an. Grafiken und Bitmaps werden unter folgendem Pfad abgelegt: [System oem-Verzeichnis]/ico/ico<Auflösung>. Entsprechend der Auflösung des Displays ist das jeweilige Verzeichnis zu verwenden.
4. Außerdem kann **die Position** des Softkeys eingestellt werden. Dazu ist im Abschnitt [Custom] folgender Eintrag vorhanden:
   ```
   SoftkeyPosition=12
   ```
   Die Position 12 ist die Voreinstellung. Dies entspricht dem HSK4 auf der Menüfortschaltleiste des Bedienbereichsmenüs. Position 1-8 entspricht HSK1 bis HSK8 auf der Menüleiste, Position 9-16 entspricht HSK1 bis HSK8 auf der Menüfortschaltleiste.

10.3 So projektieren Sie den Bedienbereich "Custom"

Softkey für Bedienbereich "Custom" projektieren

Um den Bedienbereich zu projektieren, benötigen Sie die Dateien "easyscreen.ini" und "custom.ini". Vorlagen für beide Dateien liegen im Verzeichnis [System siemens-Verzeichnis]/templates/cfg.

1. Kopieren Sie zuerst die Dateien in das Verzeichnis [System oem-Verzeichnis]/cfg und nehmen Sie die Änderungen dort vor.
2. In der Datei "easyscreen.ini" ist bereits eine Definitionszeile für den Bedienbereich "Custom" enthalten:
   ```
   ;StartFile02 = area := Custom, dialog := SlEsCustomDialog,
   startfile := custom.com
   ```
   Das ";" am Beginn der Zeile stellt das Kommentarzeichen dar. Die Zeile ist also auskommentiert und damit nicht aktiv. Um das zu ändern, muss das ";" gelöscht werden. Durch das Attribut "startfile" in dieser Zeile wird definiert, dass der Eintrag bei Anwahl des Bedienbereichs "Custom" auf die Projektdatei "custom.com" verweist.

3. Die **Projektdatei "custom.com"** legen Sie im Verzeichnis [System oem-Verzeichnis]/proj an. Darin ist die Projektierung enthalten; die analog zur Datei "aeditor.com" des Bedienbereichs "Programm" erstellt wird. Die projektierten Einstiegssoftkeys kommen daraufhin im Bedienbereich "Custom" zur Anzeige.
4. In der Datei "custom.ini" projektieren Sie **sprachunabhängigen Text** für die Titelzeile des Dialogs.
   Dazu ist in der Vorlage folgender Eintrag vorhanden:
   ```
   [Header]
   Text=Custom
   ```
   Diesen Text können Sie durch einen anwenderspezifischen Text ersetzen.
5. Um ein **Startbild** des Bedienbereichs "Custom" zu projektieren, ist in der Vorlage folgender Eintrag vorhanden:
   ```
   [Picture]
   Picture=logo.png
   ```
   Logo.png ist der Name des Startbildes, das im Startdialog des Bedienbereichs "Custom" angezeigt wird. Hier können Sie z. B. ein Firmen-Logo oder ein anderes Bild anzeigen. Die Datei ist im Verzeichnis der entsprechenden Auflösung abzulegen unter: [System oem-Verzeichnis]/ico/ ...
6. Um beim ersten Aufblenden des Bedienbereichs „Custom“ direkt eine bestimmte "Run MyScreens"-Maske aufzublenden, ist in der Vorlage folgender Eintrag vorhanden:
   ```
   ; Mask shown with startup of area "custom"
   [Startup]
   ;Startup = Mask:=MyCustomStartupMask, File:=mycustommasks.com
   ```
   Der typische Anwendungsfall ist z. B., dass beim Start von SINUMERIK Operate direkt mit einer bestimmten "Run MyScreens"-Maske gestartet wird.
   Hierzu muss in der Konfigurationsdatei "systemconfiguration.ini" in der Sektion „[miscellaneous]“ der Schlüssel „startuparea“ folgendermaßen eingestellt werden:
   ```
   [miscellaneous]
   ;name of the area to be shown at system startup
   startuparea = Custom
   ```

10.4 Programmierbeispiel für den Bereich "Custom"

Beispiel

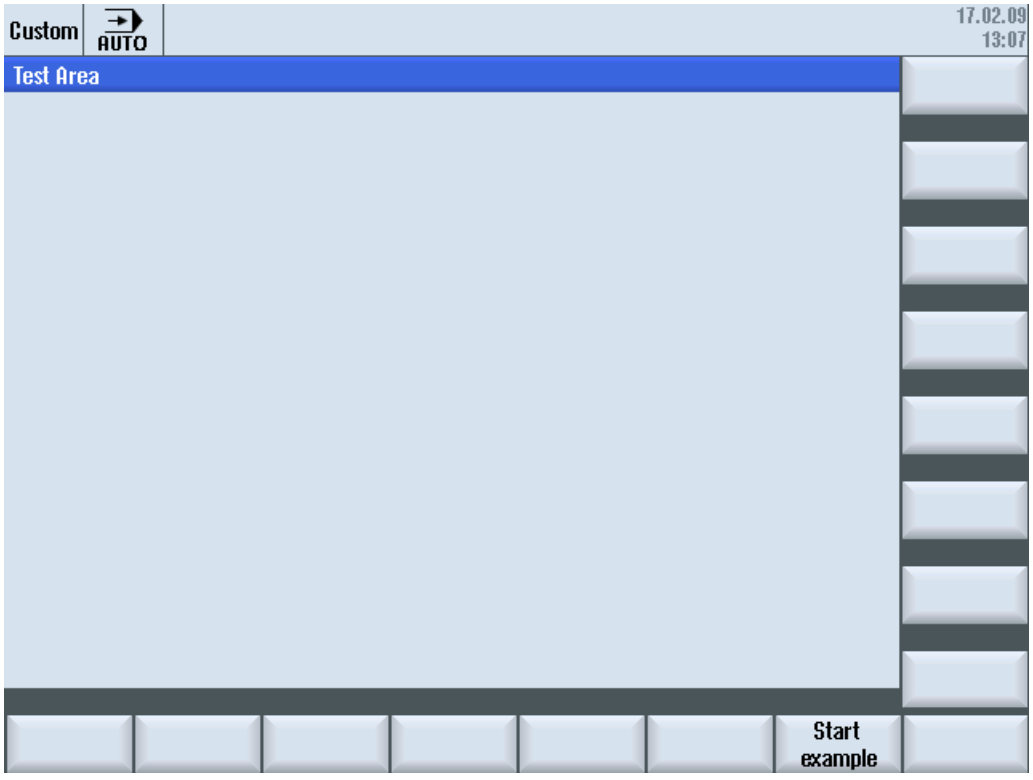


Bild 10-1 Beispiel mit Softkey "Start example"

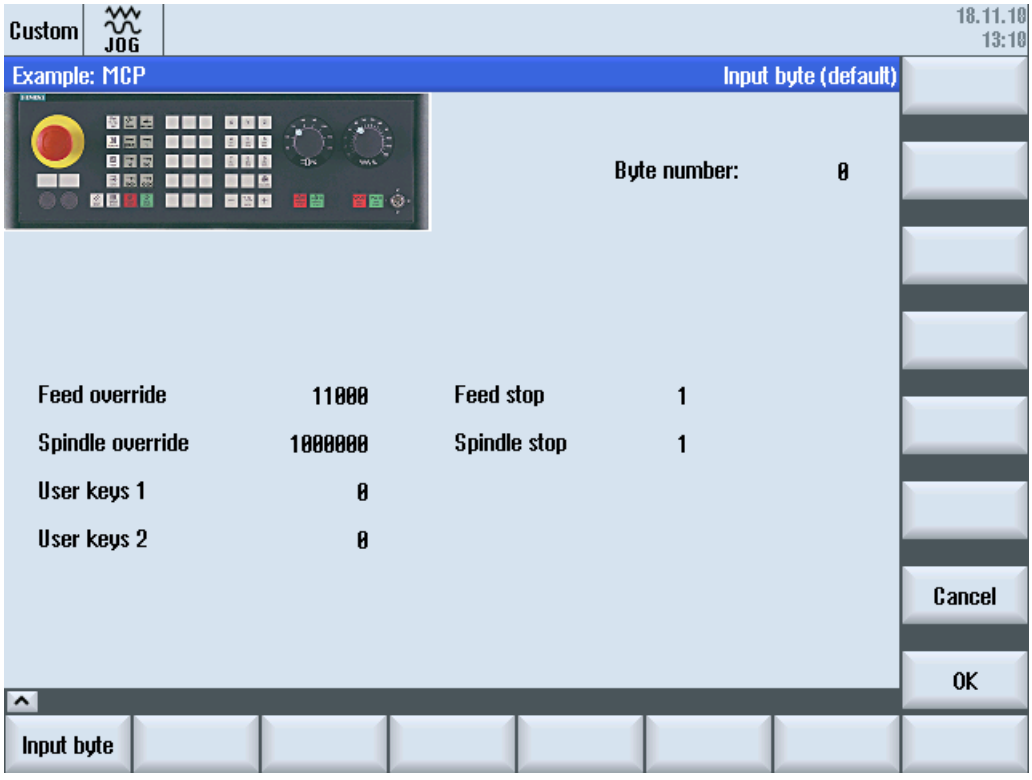


Bild 10-2 Beispiel mit Bitmap und Text-Feldern

Dateiübersicht

Folgende Dateien werden benötigt:

- "custom.com"
- "easyscreen.ini"

Programmierung

Inhalt der Datei "custom.com":

Hinweis

Die im Beispiel eingebundene Grafikdatei "mcp.png" ist exemplarisch. Falls Sie dieses Beispiel nachprogrammieren möchten, müssen Sie die Grafik durch eine Ihnen verfügbare Grafik ersetzen.

```
//S(Start)
   HS7=("Start example", se1, ac7)
   PRESS(HS7)
      LM("Maske4")
```

```
   END_PRESS
//END
//M(Maske4/"Example: MCP"/"mcp.png")
   DEF byte=(I/0/0/"Input byte=0 (default)","Byte number:",""/wr1,li1///380,40,100/480,40,50)
   DEF Feed=(IBB//0/"","Feed override",""/wr1//"EB3"/20,180,100/130,180,100), Axistop=(B//0/"","Feed
stop",""/wr1//"E2.2"/280,180,100/380,180,50/100)
   DEF Spin=(IBB//0/"","Spindle override",""/wr1//"EB0"/20,210,100/130,210,100), spinstop=(B//
0/"","Spindle stop",""/wr1//"E2.4"/280,210,100/380,210,50/100)
   DEF custom1=(IBB//0/"," User keys 1",""/wr1//"EB7.7"/20,240,100/130,240,100)
   DEF custom2=(IBB//0/"","User keys 2",""/wr1//"EB7.5"/20,270,100/130,270,100)
   DEF By1
   DEF By2
   DEF By3
   DEF By6
   DEF By7

   HS1=("Input byte", SE1, AC4)
   HS2=("")
   HS3=("")
   HS4=("")
   HS5=("")
   HS6=("")
   HS7=("")
   HS8=("")
   VS1=("")
   VS2=("")
   VS3=("")
   VS4=("")
   VS5=("")
   VS6=("")
   VS7=("Cancel", SE1, AC7)
   VS8=("OK", SE1, AC7)

   PRESS(VS7)
      EXIT
   END_PRESS

   PRESS(VS8)
      EXIT
   END_PRESS

   LOAD
      By1=1
      By2=2
      By3=3
```

```
      By6=6
      By7=7
   END_LOAD

   PRESS(HS1)
      Byte.wr=2
   END_PRESS

   CHANGE(Byte)
      By1=byte+1
      By2=byte+2
      By3=byte+3
      By6=byte+6
      By7=byte+7
      Feed.VAR="EB"<<By3
      Spin.VAR="EB"<<Byte
      Custom1.VAR="EB"<<By6
      Custom2.VAR="EB"<<By7
      Axisstop.VAR="E"<<By2<<".2"
      Spinstop.VAR="E"<<By2<<".4"
      Byte.wr=1
   END_CHANGE

   CHANGE(Axis stop)
      IF Axistop==0
         Axistop.BC=9
      ELSE
         Axistop.BC=11
      ENDIF
   END_CHANGE

   CHANGE(Spin stop)
      IF Spinstop==0
         Spinstop.BC=9
      ELSE
         Spinstop.BC=11
      ENDIF
   END_CHANGE
//END
```

Dialoganwahl 11

11.1 Dialoganwahl über PLC Softkeys

Projektierung

Beschreibung zur Vorgehensweise:

- in der "systemconfiguration.ini" existiert ein Abschnitt [keyconfiguration]. Der Eintrag spezifiziert eine Aktion für einen speziellen PLC-Softkey.
- Als Aktion wird eine Nummer angegeben, ist sie größer oder gleich 100 handelt es sich um einen "Run MyScreens"-Aufruf.
- In der Datei "easyscreen.ini" muss zur Definition der auszuführenden Aktion ein Abschnitt angelegt werden, dessen Name sich aus dem Bedienbereichsnamen und dem Dialognamen ergibt (siehe Eintrag unter [keyconfiguration] → Area:=..., Dialog:=...) → [<Area>_<Dialog>] → z. B. [AreaParameter_SlPaDialog]
- In diesem Abschnitt werden die Aktionsnummern (die in der "systemconfiguration.ini" angegeben wurden → siehe Action:=...) definiert. Hierbei handelt es sich um zwei Befehle:
  1. LS("Sofkeyleiste1","param.com") ... Laden einer Softkey-Leiste
  2. LM("Maske1","param.com") ... Laden einer Maske

Anwahl von Softkey-Leisten über PLC-Softkeys

Bei "Run MyScreens" ist die Anwahl von "Run MyScreens"-Softkey-Leisten und "Run MyScreens"-Dialogen über PLC-Softkeys möglich. Dazu muss das bei der Projektierung der betreffenden PLC-Softkeys anzugebende Attribut "action" einen Wert größer oder gleich 100 haben.

Die Projektierung der PLC-Softkeys erfolgt in der Datei "systemconfiguration.ini" im Abschnitt [keyconfiguration]:

```
[keyconfiguration]
KEY75.1 = Area:=area, Dialog:=dialog, Screen:=screen, Action:= 100,
```

Die Projektierung der Kommandos LM und LS, die beim Eintreffen entsprechender PLC-Softkeys ausgeführt werden sollen, erfolgt in der Datei "easyscreen.ini" und zwar in Abschnitten, deren Name nach folgendem Schema aufgebaut ist:

`[areaname_dialogname]`	Der erste Teil des Namens "areaname" bezeichnet den Bedienbereich, der zweite Teil "dialogname" bezeichnet den Dialog, für den die in diesem Abschnitt projektierten Kommandos gelten.

`[AreaParameter_SlPaDialog]`
`100.screen1 = LS("Softkey1","param.com")`
`101.screen3 = LM("Maske1","param.com")`
	Es sind die in der Datei "systemconfiguration.ini" für den Bedienbereich und Dialog vergebenen Namen zu verwenden. Die Angabe des Dialogs ist optional.
	Sie kann insbesondere bei Bedienbereichen, die nur durch einen einzigen Dialog implementiert werden, entfallen, siehe nebenstehendes Beispiel.
	Wenn im Bedienbereich AreaParameter, der durch den Dialog SlPaDialog implementiert wird, "screen1" angezeigt wird, dann wird bei Auftreten der "action" mit dem Wert 100 das Kommando "LS("Softkey1","param.com")" ausgeführt.

`action.screen=Kommando`	Die beiden Attribute "action" und "screen" geben eindeutig an, wann das angegebene Kommando ausgeführt wird.
	Die Angabe von "screen" ist optional.
	Zulässige Kommandos sind:
	`LM (LoadMask)`
	`LS (LoadSoftkeys)`

11.1.1 Aufruf von Run MyScreens-Zyklenmasken im Programmeditor

Einsatzbereich

Über PLC Keys wird die Möglichkeit geschaffen, Run MyScreens-Zyklenmasken im Programmeditor direkt zu öffnen.

Randbedingung

Ein Technologiezyklus kann nur bei geöffnetem Programmeditor ausgewählt werden. Der über die Maske parametrierte Technologiezyklus wird an der aktuellen Cursorposition im Programmeditor eingefügt.

Vorgehensweise

1. Erstellen Sie den Run MyScreens-Zyklenaufruf über PLC Hardkeys (Seite 284).
2. Damit die Run MyScreens-Maske ordnungsgemäß in den Programmeditor integriert werden kann, müssen Sie die PLC Key-Projektierung wie im folgenden Beispiel beschrieben erweitern:
   Beispiel für eine vollständige KeyProjektierung:
   systemconfiguration.ini
   ```
   [keyconfiguration]
   KEY101.0 = action:=209, runmyscreenmode:=HandledByDialog
   ```
   easyscreen.ini
   ```
   [AreaProgramEdit_SlProgramEdit]
   209 = LM("MASK_E_DR_O1_MAIN","e_dr_o1.com")
   ```
3. Die Rückmeldung der geöffneten Run MyScreens-Zykelnemaske können Sie über PLC-Nahtstelle "DB19.DBW24" überprüfen.
   Beispiel:
   In diesem Beispiel wird der Wert „9876" auf die PLC Nahtstelle geschrieben, wenn die Run MyScreens-Maske geöffnet wird:
   Projektierung mit PLC-ID=9876:
   //M(MASK_E_DR_O1_MAIN/$85305/////XG1,FA2,TA7//"drilling.txt"/**9876**)
   oder
   //M{ MASK_E_DR_O1_MAIN, HD=$85305, LANGFILELIST=" drilling.txt", **PLC_ID=9876**}
   Siehe auch: Funktionshandbuch SINUMERIK 840D sl Grundfunktionen

Status des Programmeditors auf der OA-Schnittstelle

Den Status des Programmeditors können Sie über die CAP-Lokale-Variable "/Hmi/StepEditorInfo" abfragen.

Name der Variable:	/Hmi/StepEditorInfo
Beschreibung der Variable:	Informationen des Editors zum Programm, das den Fokus hat.
Der String enthält folgende Informationen	
[fileName]	Pfadangabe des Programms
[channel]	Kanalnummer
[technlogy]	Technologie
[appearance]	Programm-Kennung (ShopMill/ShopTurn/G-Code)
[state]	Status zum StepEditor: • EditorClosed: Editor ist geschlossen (z. B. HMI nicht in Bedienbereich Programm) • EditEnabled: Editieren erlaubt • EditDisabled: Editieren nicht erlaubt (z. B. readOnly oder ungültige Cursorposition)

11.2 Dialoganwahl über PLC Hardkeys

Einsatzbereich

Von der PLC können in der Bediensoftware folgende Funktionen initiiert werden:

- Anwahl von Bedienbereich
- Anwahl bestimmter Szenarien innerhalb von Bedienbereichen
- Ausführung auf Softkeys projektierte Funktionen

Hardkeys

Alle Tasten - auch die PLC-Keys - werden nachfolgend als Hardkeys bezeichnet. Es können maximal 254 Hardkeys definiert werden. Dabei gilt folgende Aufteilung:

Tastennummer	Verwendung
Key 1 – Key 9	Tasten an der Bedientafelfront
Key 10 – Key 49	reserviert
Key 50 – Key 254 Key 50 – Key 81	PLC-Keys: für OEM reserviert
Key 255	Durch Steuerinformation vorbelegt.

Die Hardkeys 1 - 9 sind folgendermaßen vorbelegt:

Tastenbezeichnung		Aktion / Auswirkung
HK1	MACHINE	Anwahl Bedienbereich "Maschine", letzter Dialog
HK2	PROGRAM	Anwahl Bedienbereich "Programm", letzter Dialog oder letztes Programm
HK3	OFFSET	Anwahl Bedienbereich "Parameter", letzter Dialog
HK4	PROGRAM MANAGER	Anwahl Bedienbereich "Programm", Grundbild "Programm-Manager"
HK5	ALARM	Anwahl Bedienbereich "Diagnose", Dialog "Alarmliste"
HK6	CUSTOM	Anwahl Bedienberich "Custom"
HK7 [1)]	MENU SELECT	Anwahl "Grundmenü"
HK8 [1)]	MENU FUNCTION	Anwahl Bedienbereich "Function"
HK9 [1)]	MENU USER	Anwahl Bedienbereich "User"

1) nur bei 828D

Projektierung

Die Projektierung erfolgt in der Konfigurationsdatei "systemconfiguration.ini" im Abschnitt [keyconfiguration]. Jede Zeile definiert ein so genanntes Hardkey-Ereignis. Unter einem Hardkey-Ereignis wird die n-te Betätigung eines bestimmten Hardkeys verstanden. Zum Beispiel können die zweite und dritte Betätigung eines bestimmten Hardkeys unterschiedliche Reaktionen zur Folge haben.

Die Einträge in der Konfigurationsdatei "systemconfiguration.ini" können mit anwenderspezifischen Einstellungen überladen werden. Dazu stehen die Verzeichnisse [System user-Verzeichnis]/cfg und [System oem-Verzeichnis]/cfg zur Verfügung.

Die Zeilen zur Projektierung der Hardkey-Ereignisse haben folgenden Aufbau:

```
KEYx.n = area:=area, dialog:=dialog, screen:=screen, forms:=form,
menus:=menu,
action:=menu.action, cmdline:=cmdline
KEYx.n = area:=area, dialog:=dialog, cmdline:=cmdline, action:=
action
```

x: Nummer des Hardkeys, Wertebereich: 1 – 254

n: Ereignisnummer – entspricht der n-ten Betätigung des Hardkeys, Wertbereich: 0 – 9

Voraussetzung

Das PLC-Anwenderprogramm muss folgende Voraussetzung erfüllen:

Es wird immer nur ein Hardkey abgearbeitet. Deshalb darf eine neue Anforderung nur dann gesetzt werden, wenn die Bediensoftware die vorhergehende Anforderung quittiert hat. Wenn das PLC-Anwenderprogramm den Hardkey aus einer MCP-Taste ableitet, dann muss es für eine ausreichende Zwischenpufferung der Taste(n) sorgen, damit auch bei schneller Bedienung kein Tastendruck verlorengeht.

PLC-Nahtstelle

In der PLC-Nahtstelle wird ein Bereich für die Anwahl eines Hardkeys vorgesehen. Der Bereich befindet sich im DB19.DBB10. Hier kann die PLC direkt einen Tastenwert zwischen 50 und 254 vorgeben.

Die Quittung durch die Bediensoftware erfolgt in zwei Schritten. Diese Vorgehensweise ist erforderlich, damit der gleiche Tasten-Code zweimal hintereinander von der Bediensoftware korrekt als zwei separate Ereignisse erkannt werden kann. Im ersten Schritt wird die Steuerinformation 255 in das Byte DB19.DBB10 geschrieben. Durch diesen definierten virtuellen Tastendruck kann jede Tastensequenz der PLC eindeutig erkannt werden. Die Steuerinformation hat für das PLC-Anwenderprogramm keine Bedeutung und darf nicht verändert werden. Im zweiten Schritt erfolgt dann die eigentliche Quittung gegenüber der PLC, indem DB19.DBB10 gelöscht wird. Ab diesem Zeitpunkt kann das PLC-Anwenderprogramm einen neuen Hardkey vorgeben. Parallel dazu wird die Anforderung des aktuellen Hardkeys in der Bediensoftware bearbeitet.

Beispiel

Konfigurationsdatei:

```
; PLC hardkeys (KEY50-KEY254)
[keyconfiguration]
KEY50.0 = name := AreaMachine, dialog := SlMachine
KEY51.0 = name := AreaParameter, dialog := SlParameter
KEY52.0 = name := AreaProgramEdit, dialog := SlProgramEdit
KEY53.0 = name := AreaProgramManager, dialog := SlPmDialog
KEY54.0 = name := AreaDiagnosis, dialog := SlDgDialog
```

```
KEY55.0 = name := AreaStartup, dialog := SlSuDialog
KEY56.0 = name := Custom, dialog := SlEsCustomDialog
```

Die Area- und Dialogbezeichner sind der "systemconfiguration.ini" aus [System siemens-Verzeichnis]/cfg zu entnehmen.

11.3 Dialoganwahl über NC

MMC-Befehl in HMI Operate

MMC-Befehle können Sie wie im Folgenden beschrieben anwenden:

1. **Definition von MMC-Kommandos**
   In der Standarddatei "systemconfiguration.ini" gibt es folgende Kombinationen:
   address:=MCYCLES --> command:=LM
   address:=CYCLES --> command:=PICTURE_ON
   Diese Unterscheidung ist notwendig um zwischen Messzyklen und anderen Zyklen zu unterscheiden. Das heißt:
   - LM gilt immer für Messzyklen, PICTURE_ON immer für nicht-Messzyklen
   - Neu definierte MMC-Kommandos dürfen nicht PICTURE_ON und LM bezeichnet werden
2. **"Run MyScreens"-Lizenz**
   Sämtliche von "Run MyScreen" geöffneten Dialoge, außer die Messzyklen, fallen unter die "Run MyScreens"-Lizenz und können somit ohne Lizenz nur in begrenzter Dialoganzahl benutzt werden.
   Beispielaufruf mit "test.com" (Run MyScreens):

```
g0 f50
MMC("CYCLES,PICTURE_ON,test.com,maske1","A")
m0
MMC("CYCLES,PICTURE_OFF","N")
M30
```

Beispiele für Zyklenmasken 12

12.1 Beispiele für Zyklenmasken

Mit der Installation von SINUMERIK Operate werden auch Beispiele für mit Run MyScreens erstellte Zyklenmasken abgelegt.

Die Run MyScreens Beispiele finden Sie unter folgenden Ablagepfaden:

[System siemens-Verzeichnis]		
...\siemens\sinumerik\hmi\template\easyscreen\		
...		
	standard\	Hilfebilder, PNG
	x3d\	Hilfebilder, Animationen
...		
	Milling\	Fräsbearbeitung
	Turning\	Drehbearbeitung
...		
	deu\	deutsche Beschreibung
	eng\	englische Beschreibung

Auf der HMI-Oberfläche sind die Beispiele im Bedienbereich Inbetriebnahme zu finden:

Horizontaler Softkey Systemdaten → HMI-Daten → Vorlagen → Beispiele → Easyscreen → ... (siehe oben).

Kurzbeschreibung der Beispiele

- Fräsbearbeitung
  - Beispiel Bohren: hier ist das Anhängen einer Position möglich
  - Beispiel Messen Werkstück: hier kann bei geöffnetem Dialog mittels NC-Start ein Zyklenaufruf generiert und abgearbeitet werden
  - Beispiel Werkstückzähler: an diesem Beispiel wird die Handhabung mit GUD's gezeigt
- Drehbearbeitung
  - Vorrichtungen: diese enthält einen Reitstock, Stangenlader, Teilefänger
  - Beispiel Messen Werkstück: hier kann bei geöffnetem Dialog mittels NC-Start ein Zyklenaufruf generiert und abgearbeitet werden
  - Beispiel Bohren: hier ist das Anhängen einer Position möglich

Projektierung im Sidescreen 13

Für die Anzeige von Run MyScreens-Projektierungen im Sidescreen stehen Ihnen folgende Möglichkeiten zur Verfügung:

1. Anzeige einer Run MyScreens-Projektierung in einer (separaten) Sidescreen-Page:
   Die Run MyScreens-Projektierung nimmt dabei den gesamten Platz der Sidescreen-Page ein.
2. Anzeige mehrerer Run MyScreens-Projektierungen in einer (separaten) Sidescreen-Page:
   Die einzelnen Run MyScreens-Projektierungen werden innerhalb der Sidescreen-Page, in so genannte Sidescreen-Elementen, übereinander angezeigt. Die Sidescreen-Elemente, und damit auch die Run MyScreens-Projektierungen, können vertikal gescrollt werden.
3. Hinzufügen einer bzw. mehrerer Run MyScreens-Projektierungen zu einer vorhandenen Sidescreen-Page:
   D. h. die Run MyScreens-Projektierung wird zu einer im SIEMENS-Lieferumfang enthaltenen Sidescreen-Page hinzugefügt. Dieser Fall stimmt bis auf die betroffene Sidescreen-Page mit Punkt 2 überein.
4. Hinzufügen einer bzw. mehrerer Run MyScreens-Projektierungen zu einem Sidescreen-Element:
   Die einzelnen Run MyScreens-Projektierungen werden in so genannte Sidescreen-Widgets nebeneinander angezeigt und können horizontal gescrollt werden.

13.1 Anzeige einer Run MyScreens-Projektierung in einer Sidescreen-Page

Für die Anzeige einer Run MyScreens-Projektierung in einer (separaten) Sidescreen-Page müssen Sie zur bestehenden Sidescreen-Konfiguration ("slsidescreen.ini") eine neue Sidescreen-Page hinzufügen. In dieser Sidescreen-Page wird dann die Run MyScreens-Projektierung angezeigt bzw. ausgeführt.

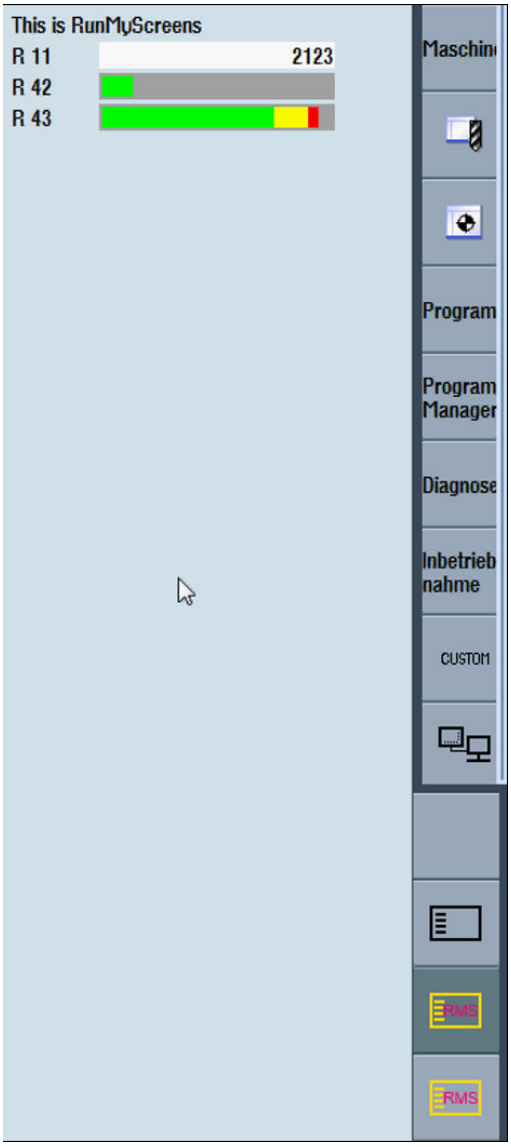


Bild 13-1 **Anzeige einer Run MyScreens-Projektierung in einer Sidescreen-Page**

Vorgehensweise

Erweitern Sie die "slsidescreen.ini" um folgenden Eintrag:

```
[Sidescreen]
PAGE100= name:=RMSPage,
implementation:=sleseasyscreen.SlEsSideScreenPage
```

- Die Nummer der Page, hier "100", müssen Sie entsprechend der vorhandenen Sidescreen-Pages hochzählen. Die Nummer der Page muss ≥ 100 sein. Damit vermeiden Sie Konflikte mit den SIEMENS-Pages.
- Den Namen der Page, hier "RMSPage", können Sie frei wählen.
- Die Angabe für die Implementierung der Page, hier "sleseasyscreen.SlEsSideScreenPage", dürfen Sie nicht verändern.

Konfigurieren Sie im nächsten Schritt die auszuführende Run MyScreens-Projektierung. Legen Sie dazu eine neue Sektion an, deren Name den Namen der neu angelegten Page enthält:

```
[Page_RMSPage]
Icon=rmspage.png
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
PROPERTY003= name:=focusable, type:=bool, value:="true"
```

- Der Name der Sektion wird gebildet, indem Sie an das Präfix "Page_" den Name der Page anhängen, hier "RMSPage".
- Unter Icon geben Sie den Dateinamen an, hier "rmspage.png". Die Datei enthält das Symbol für den Button zur Anwahl der Page. Legen Sie die Datei im Verzeichnis /user/sinumerik/hmi/ico/ico640 bzw. /oem/sinumerik/hmi/ico/ico640 ab.
- Im Property maskPath geben Sie den Namen der Datei an, die die RunMyScreens-Projektierung enthält, hier "sidescreenmask.com".
- Im Property maskName geben Sie den Namen der RunMyScreens-Maske an, die angezeigt werden soll, hier "Mask".
- Im Property focusable legen Sie den Eingabefokus fest. Nur wenn dieses Attribut den Wert "true" hat, kann die Page den Eingabefokus bekommen. Dieses Attribut wird für Pages benötigt, die Eingabeelemente haben.

Legen Sie die "slsidescreen.ini" im Verzeichnis /oem/sinumerik/hmi/cfg bzw. /user/sinumerik/hmi/cfg ab.

Nach dem nächsten HMI-Hochlauf ist die Run MyScreens-Projektierung verfügbar.

Beispiel

Beispiel für eine vollständige Konfiguration in der "slsidescreen.ini":

```
[Sidescreen]
PAGE005= name:=RMSPage,
implementation:=sleseasyscreen.SlEsSideScreenPage
[Page_RMSPage]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

13.2 Anzeige mehrerer Run MyScreens-Projektierungen in einer Sidescreen-Page

Für die Anzeige mehrerer Run MyScreens-Projektierungen in einer (separaten) Sidescreen-Page müssen Sie, zusätzlich zur Sidescreen-Page, sogenannte Sidescreen-Elemente konfigurieren. Die Sidesceen-Elemente dienen zur Anzeige der Run MyScreens-Projektierungen.

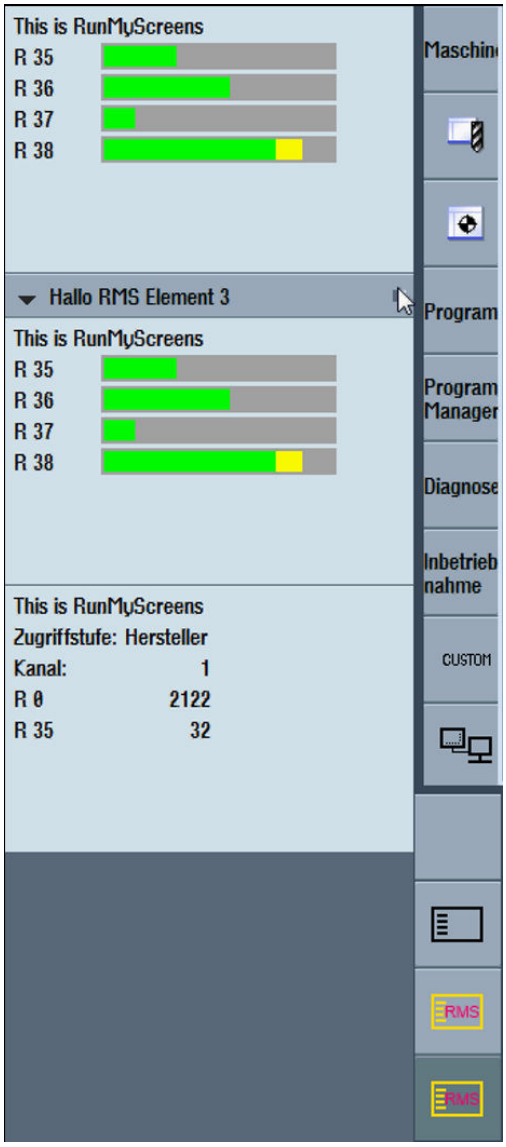


Bild 13-2 **Anzeige mehrerer Run MyScreens-Projektierungen in einer Sidescreen-Page**

Vorgehensweise

Erweitern Sie die "slsidescreen.ini" um folgende Einträge:

```
[Sidescreen]
PAGE100= name:=RMSPage, implementation:=SlSideScreenPage
```

- Die Nummer der Page, hier "100", müssen Sie entsprechend der vorhandenen Sidescreen-Pages hochzählen. Die Nummer der Page muss ≥ 100 sein. Damit vermeiden Sie Konflikte mit den SIEMENS-Pages.
- Den Namen der Page, hier "RMSPage", können Sie frei wählen.
- Die Angabe für die Implementierung der Page, hier "SlSideScreenPage", dürfen Sie nicht verändern.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement001,
implementation:=sleseasyscreen.SlEsSideScreenElement
ELEMENT002= name:=RMSElement002,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Der Page sind zwei Sidescreen-Elemente zugeordnet – RMSElement001 und RMSElement002. Die Namen der Elemente können Sie frei wählen.
- Die Angabe für die Implementierung der Elemente, hier "sleseasyscreen.SlEsSideScreenElement", dürfen Sie nicht verändern.

Schließlich müssen Sie den Sidescreen-Elementen noch die in diesen Elementen anzuzeigenden Run MyScreens-Projektierungen zuordnen:

```
[Element_RMSElement001]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask001"
…
[Element_RMSElement002]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask002"
```

Legen Sie die "slsidescreen.ini" im Verzeichnis /oem/sinumerik/hmi/cfg bzw. /user/sinumerik/hmi/cfg ab.

Nach dem nächsten HMI-Hochlauf ist die Run MyScreens-Projektierung verfügbar.

13.3 Hinzufügen von Run MyScreens-Projektierungen zu einer Sidescreen-Page

Die Vorgehensweise zur Integration einer bzw. mehrerer Run MyScreens-Projektierungen zu einer vorhandenen Sidescreen-Page ist identisch zur im Kapitel Anzeige mehrerer Run MyScreens-Projektierungen in einer Sidescreen-Page (Seite 292) beschriebenen Vorgehensweise. Einziger Unterschied ist, dass die Sidescreen-Elemente zur Anzeige der Run MyScreens-Projektierungen zu einer bereits vorhandenen Sidescreen-Page hinzugefügt werden.

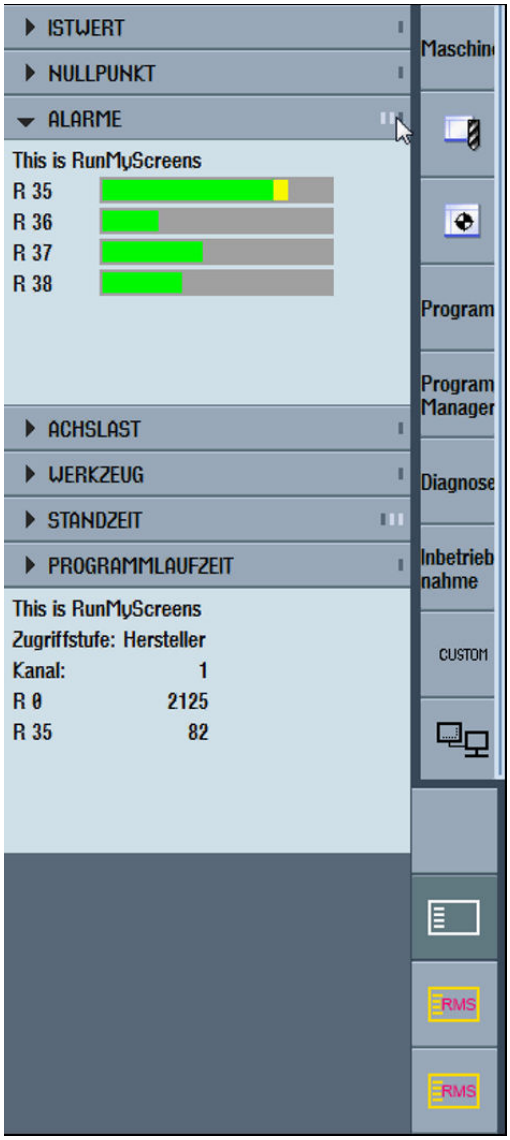


Bild 13-3 **Hinzufügen von Run MyScreens-Projektierungen zu einer Sidescreen-Page**

Hinweis

Von den im SIEMENS-Lieferumfang enthaltenen Sidescreen-Pages können Sie nur die WidgetsPage (siehe Datei "slsidescreen.ini") mit Run MyScreens-Projektierungen erweitern.

Vorgehensweise

Erweitern Sie die "slsidescreen.ini" um folgende Einträge:

Fügen Sie im Abschnitt [Page_WidgetsPage] ein entsprechendes Element zur Aufnahme der Run MyScreen-Projektierung hinzu.

```
[Page_WidgetsPage]
ELEMENT100= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Verwenden Sie für neue Elemente einen Nummernbereich ≥ 100. Damit vermeiden Sie Konflikte mit den SIEMENS-Anzeigeelementen. Das RMSElement wird in der WidgetsPage entsprechend dieser Nummerierung unterhalb der SIEMENS-Elemente eingefügt.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask001"
```

13.4 Hinzufügen von Run MyScreens-Projektierungen zu einem Sidescreen-Element

Basis der im Folgenden exemplarisch beschriebenen Vorgehensweise ist eine Sidescreen-Page mit einem Sidescreen-Element. In dem Sidescreen-Element werden zwei Run MyScreens-Projektierungen nebeneinander angezeigt.

Konfigurieren Sie wie im Folgenden beschrieben eine Sidescreen-Page mit einem Sidescreen-Element. Ordnen Sie dem Sidescreen-Element zwei Sidescreen-Widgets mit den jeweiligen Run MyScreens-Projektierugen zu.

Vorgehensweise

Erweitern Sie die "slsidescreen.ini" um folgende Einträge:

```
[Sidescreen]
PAGE100= name:=RMSPage, implementation:=SlSideScreenPage
```

- Die Angabe der Implementierung, hier "SlSideScreenPage", dürfen Sie nicht verändern.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement, implementation:= SlSideScreenElement
```

- Die Angabe der Implementierung, hier "SlSideScreenElement", dürfen Sie nicht verändern.

```
[Element_RMSElement]
TextId=RMS_ELEMENT_TITLE
TextFile=rmstexts
TextContext=rmstexts
TextContext=rmstexts
```

```
WIDGET001= name:=RMSWidget001,
implementation:=sleseasyscreen.SlEsSideScreenWidget
WIDGET002= name:=RMSWidget002,
implementation:=sleseasyscreen.SlEsSideScreenWidget
```

- Bei Verwendung der Implementierung SlSideScreenElement für das Sidescreen-Element (siehe Sektion [Page_RMSPage]), besitzt das Sidescreen-Element eine Titelzeile, in der ein Text angezeigt werden kann.
  Dieser Text wird mit den Einträgen TextId, TextFile und TextContext projektiert:
  - TextId: Text-Id des anzuzeigenden Textes
  - TextFile: Datei, in der der Text abgelegt ist
  - TextContext: Textkontext, in dem sich der Text innerhalb dieser Datei befindet

  Für jede Sprache ist eine separate Datei bereitzustellen.
  Der Name dieser Dateien wird nach folgendem Schema gebildet:
  <TextFile>_<Sprachsuffix>.ts, z. B. rmstexts_deu.ts für obiges Beispiel.
  Wenn Sie keine Datei und keinen Kontext angegeben (TextFile=<empty> und TextContext=<empty>), wird die Text-Id als Text angezeigt. Als Text-Id können Sie einen beliebigen String angeben.
  Die Textdatei (z. B. rmstexts_deu.ts) ist wie folgt aufgebaut:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TS>
<TS>
 <context>
 <name>rmstexts</name>
 <message>
  <source>Meine Applikation</source>
 <translation>My Application</translation>
 </message>
 </context>
</TS>
```

  Legen Sie die Datei im Verzeichnis /oem/sinumerik/hmi/lng bzw. /user/sinumerik/hmi/lng ab. Beim nächsten HMI-Hochlauf wird die Datei in das zur Laufzeit notwendige Dateiformat konvertiert.
- Die Namen der dem Element zugeordneten Widgets, hier "RMSWidget001" und "RMSWidget002", können Sie frei wählen.
- Die Widget-Implementierung "sleseasysscreen.SlEsSideScreenWidget" dürfen Sie nicht verändern.

```
[Widget_RMSWidget001]
PROPERTY001= name:=maskPath, type:=QString, value:="mymask001.com"
PROPERTY002= name:=maskName, type:=QString, value:="MyMask001"

[Widget_RMSWidget002]
PROPERTY001= name:=maskPath, type:=QString, value:="mymask002.com"
PROPERTY002= name:=maskName, type:=QString, value:="MyMask002"
```

- Die Namen der Sektionen zur Konfiguration der Sidescreen-Widgets werden gebildet, indem Sie an das Präfix "Widget_" den Namen des Widgets anhängen, das in dieser Sektion konfiguriert werden soll.
- Im Property maskPath geben Sie den Namen der Datei an, die die RunMyScreens-Projektierung enthält, hier "mymask001.com" bzw. "mymask002.com".
- Im Property maskName geben Sie den Namen der RunMyScreens-Maske an, die angezeigt werden soll, hier "MyMask001" bzw. "MyMask002".

13.5 Hinweise zur Ausführung von Run MyScreens-Projektierungen im Sidescreen

Beachten Sie folgende Hinweise:

- Die Funktionen LS, LM, DLGL, EXIT, EXITLS stehen bei Ausführung von Run MyScreens-Projektierungen im Sidescreen nicht zur Verfügung.
- Die Abfrage der Maskengröße im LOAD-Block ist nicht möglich.
- Texte werden stets mit dem Font angezeigt, der in SINUMERIK Operate bei einer Bildschirmauflösung von 640x480 Pixel verwendet wird.
- Die projektierten Positionen von Bedienelementen werden ohne Änderung (z. B. Streckung) umgesetzt.
- Die Dialoggröße wird automatisch angepasst.
- Werden Run MyScreens-Projektierungen in einer Sidescreen-Page ausgeführt, steht der Projektierung zur Anzeige die gesamte Fläche der Page zur Verfügung. Bei der Ausführung in Sidescreen-Elementen oder -Widgets wird die zur Anzeige verfügbare Fläche vom System kontextabhängig vorgegeben.
- Header und Softkeys sind nicht projektierbar .
- Alle Debug- und Fehlerausgaben werden sequenziell in die Textdatei "easyscreen_log.txt" geschrieben. Eine spezifische Kennung, aus welcher Projektierung eine Meldung kommt, ist nicht vorhanden.
- Der Zustand einer im Sidescreen angezeigten Run MyScreens-Projektierung wird beim Abblenden der Projektierung verworfen.

Hinweis

Die Integration von Run MyScreens-Projektierungen in den Sidescreen führt ggf. dazu, dass mehr als eine Run MyScreens-Projektierung gleichzeitig ausgeführt wird. Um die Funktionsfähigkeit des Gesamtsystems aufrecht zu erhalten, achten Sie, je nach eingesetzter Hardware, auf die dadurch zusätzlich erzeugte Systemlast.

Projektierung im Display-Manager 14

14.1 Run MyScreens-Projektierungen im Display-Manager anzeigen

Für die Anzeige von Run MyScreens-Projektierungen im Display-Manager steht Ihnen der Dialog SlSideScreenDialog zur Verfügung. Dieser Dialog kann je nach verfügbarem Platz eine oder mehrere Sidescreen-Pages (siehe IM9 Kapitel 3.13) anzeigen. Mehrere Pages werden in Spaltenform nebeneinander angezeigt. Der verfügbare Platz (Breite) wird dabei gleichmäßig auf die einzelnen Spalten aufgeteilt. Die Anzahl der anzuzeigenden Pages und deren Inhalt werden konfiguriert. Jede Page hat eine eigene Konfigurationsdatei. Die Namen dieser Konfigurationsdateien werden bei der Dialog-Projektierung in der Datei "systemconfiguration.ini" angegeben, im Kommandozeilenparameter "cmdline". Hinter dem Parameterbezeichner "–sidescreen1" steht der Name der Konfigurationsdatei für die erste Spalte, hinter dem Parameterbezeichner "–sidescreen2" der Name der Konfigurationsdatei für die zweite Spalte, usw.

Beispiel

```
[dialogs]
DLG500= name:=RMSApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false, cmdline:="-sidescreen1
rmspage1.ini –sidescreen2 rmspage2.ini –spacing 3”
```

Die in diesem Beispiel angelegte Instanz des Dialogs SlSideScreenDialog hat den Namen RMSApp. Sie hat 2 Pages/Spalten. Die beiden Pages/Spalten werden in den Dateien "rmspage1.ini" und "rmspage2.ini" konfiguriert. Zwischen den beiden Spalten ist ein Abstand von 3 Pixeln vorhanden.

Hinweis

Die Namen der Dateien mit den Page-Konfigurationen, hier "rmspage1.ini" und "rmspage2.ini", können Sie frei wählen.

Die Verwendung von Run MyScreens-Projektierungen im Display-Manager erfolgt wie in Kapitel Projektierung im Sidescreen (Seite 289) beschrieben. Der einzige Unterschied besteht darin, dass die Integration der Projektierungen nicht in der Datei "slsidescreen.ini" vorgenommen wird, sondern in den für die Konfiguration der vorhandenen Pages jeweils vorgesehenen Dateien.

Für die Integration von Run MyScreens-Projektierungen in den Display-Manager von SINUMERIK Operate stehen Ihnen 2 Varianten zur Verfügung. Diese werden in den folgenden Kapiteln beschriebenen.

14.2 Integration in SISideScreenDialog

Für die Integration der Run MyScreens-Projektierung müssen Sie die Dialog-Instanz von SlSideScreenDialog in der Datei "systemconfiguration.ini" anlegen und die Datei zur Integration der Run MyScreens-Projektierung erstellen.

Nachfolgendes Beispiel zeigt die Integration einer Run MyScreens-Projektierung. Die Run MyScreens-Projektierung belegt das komplette Fenster des Dialogs SlSideScreenDialog, d. h. es ist nur eine Page/Spalte vorhanden.

Vorgehensweise

Zunächst legen Sie eine Instanz des Dialogs SlSideScreenDialog in der Datei "systemconfiguration.ini" im Abschnitt `[dialogs]` an. Die Instanz erhält den Namen "RMSApp".

```
[dialogs]
DLG500= name:=RMSApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false,
cmdline:="-sidescreen1 rmsapp.ini"
```

Im nächsten Schritt konfigurieren bzw. integrieren Sie die RunMyScreens-Projektierung in der Datei "rmsapp.ini":

```
[Sidescreen]
PAGE001= name:=RMSPage, implementation:=SlSideScreenPage
```

- Legen Sie eine Page mit dem Namen "RMSPage" an.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Auf der Page RMSPage legen Sie ein Element mit dem Namen RMSElement an. Dieses Element dient zur Anzeige der Run MyScreens-Projektierung.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

- Im Property maskPath geben Sie den Namen der Datei an, die die RunMyScreens-Projektierung enthält, hier "sidescreenmask.com".
- Im Property maskName geben Sie den Namen der RunMyScreens-Maske an, die angezeigt werden soll, hier "Mask".

Legen Sie die Dateien "rmsapp.ini" und "systemconfiguration.ini" (siehe oben) im Verzeichnis /oem/sinumerik/hmi/cfg bzw. /user/sinumerik/hmi/cfg ab.

Nach dem nächsten HMI-Hochlauf ist die Run MyScreens-Projektierung verfügbar.

14.3 Integration in SIWidgetsApp

Im Folgenden wird die Integration einer Run MyScreens-Projektierung in die zum SIEMENS-Lieferumfang gehörende Applikation SIWidgetsApp beschrieben.

Abhängig davon, in welche Page/Spalte der Applikation SIWidgetsApp die Run MyScreens-Projektierung integriert werden soll, müssen Sie entweder die Datei "sldmwidgets1.ini" oder die Datei "sldmwidgets2.ini" entsprechend erweitern.

Nachfolgendes Beispiel zeigt die Integration einer Run MyScreens-Projektierung in die linke Page/Spalte der Applikation SlWidgetsApp.

Vorgehensweise

Erweitern Sie die Datei "sldmwidgets1.ini" wie folgt:

```
[Sidescreen]
PAGE001= name:=WidgetsPage, implementation:=SlSideScreenPage
[Page_WidgetsPage]
ELEMENT100= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Legen Sie ein Element mit dem Namen "RMSElement" zur Anzeige der Run MyScreens-Projektierung an.

Hinweis

Verwenden Sie für neue Elemente einen Nummernbereich ≥ 100. Damit vermeiden Sie Konflikte mit den SIEMENS-Anzeigeelementen. Das RMSElement wird in der WidgetsPage entsprechend dieser Nummerierung unterhalb der SIEMENS-Elemente eingefügt.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

- Im Property maskPath geben Sie den Namen der Datei an, die die RunMyScreens-Projektierung enthält, hier "sidescreenmask.com".
- Im Property maskName geben Sie den Namen der Run MyScreens-Maske an, die angezeigt werden soll, hier "Mask".

Legen Sie die Datei "sldmwidgets1.ini" ist im Verzeichnis /oem/sinumerik/hmi/cfg bzw. /user/sinumerik/hmi/cfg ab.

Nach dem nächsten HMI-Hochlauf ist die Run MyScreens-Projektierung verfügbar.

14.4 Hinweise zur Ausführung von Run MyScreens-Projektierungen im Display-Manager

Beachten Sie folgende Hinweise:

- Die Funktionen LS, LM, DLGL, EXIT, EXITLS stehen bei Ausführung von Run MyScreens-Projektierungen im Display-Manager nicht zur Verfügung.
- Die Abfrage der Maskengröße im LOAD-Block ist nicht möglich.
- Texte werden stets mit dem Font angezeigt, der in SINUMERIK Operate bei einer Bildschirmauflösung von 640x480 Pixel verwendet wird.
- Die projektieren Positionen von Bedienelementen werden ohne Änderung (z. B. Streckung) umgesetzt.

- Werden Run MyScreens-Projektierungen in einer Sidescreen-Page ausgeführt steht der Projektierung zur Anzeige die gesamte Fläche der Page zur Verfügung. Bei der Ausführung in Sidescreen-Elementen oder –Widgets wird die zur Anzeige verfügbare Fläche vom System kontextabhängig vorgegeben.
- Alle Debug- und Fehlerausgaben werden sequentiell in die Textdatei easyscreen_log.txt geschrieben. Eine spezifische Kennung, aus welcher Projektierung eine Meldung kommt, ist nicht vorhanden.
- Der Zustand einer im Display-Manager angezeigten Run MyScreens-Projektierung wird beim Abblenden der Projektierung verworfen.

Hinweis

Die Integration von Run MyScreens-Projektierungen in den Display-Manager führt ggf. dazu, dass mehr als eine Run MyScreens-Projektierung gleichzeitig ausgeführt wird. Um die Funktionsfähigkeit des Gesamtsystems aufrecht zu erhalten, achten Sie, je nach eingesetzter Hardware, auf die dadurch zusätzlich erzeugte Systemlast.

Referenz - Listen

A

A.1 Listen der Einstiegssoftkeys

A.1.1 Liste der Einstiegssoftkeys für Drehen

Bedienbereich Programm bei Drehen

HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK7	HSK8
Edit	**Bohren**	**Drehen**	**Konturdrehen**	**Fräsen**	**Diverses**	**Simulation**	**NC Anwahl**
HSK9	HSK10	HSK11	HSK12	HSK13	HSK14	HSK15	HSK16
--	Gerade Kreis (nur bei ShopTurn)	--	--	Werkstück messen	Werkzeug messen	**OEM**	--

Drehen

In den folgenden Tabellen werden die möglichen Einstiegssoftkeys der Technologie Drehen dargestellt. Zuordnungen einzelner Einstiegssoftkeys können systembedingt abweichen. Die angegebenen OEM-Softkeys sind für "Run MyScreens" zulässig.

programGUIDE (G-Code) Einstiegssoftkeys:

	Bohren	**Drehen**	**Konturdrehen**		**Fräsen**		**Diverses**	
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**	
VSK1	Zentrieren	Abspanen	Kontur	--	Planfräsen	Kontur	Einstellungen	High Speed Settings
VSK2	Bohren Reiben	Einstich	Abspanen	--	Tasche	Bahn	Schwenken Ebene	Parallele Achsen
VSK3	Tieflochbohren	Freistich	Abspanen Rest	--	Zapfen Mehrkant	Vorbohren	Schwenken Werkzeug	--
VSK4	Ausdrehen	Gewinde	Stechen	--	Nut	Tasche	--	--
VSK5	Gewinde	Abstich	Stechen Rest	--	Gewindefräsen	Tasche Restmat.	--	--
VSK6	**OEM**	--	Stechdrehen	--	Gravur	Zapfen	Unterprogramm	--
VSK7	Positionen	**OEM**	Stechdrehen Rest	**OEM**	**OEM**	Zapfen Restmat.	--	**OEM**
VSK8	Position wiederh.	--	>>	<<	Konturfräsen	<<	>>	<<

ShopTurn Einstiegssoftkeys:

	Bohren	**Drehen**	**Konturdrehen**		**Fräsen**		**Diverses**		
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**		**HSK10**
VSK1	Bohren mittig	Abspanen	Neue Kontur	--	Planfräsen	Neue Kontur	Einstellungen	High Speed Settings	Werkzeug
VSK2	Zentrieren	Einstich	Abspanen	--	Tasche	Bahn	Schwenken Ebene	Parallele Achsen	Gerade
VSK3	Bohren Reiben	Freistich	Abspanen Rest	--	Zapfen Mehrkant	Vorbohren	Schwenken Werkzeug	Programm wiederhol.	Kreis Mittelp.
VSK4	Tiefloch-bohren	Gewinde	Stechen	--	Nut	Tasche	Gegenspindel	--	Kreis Radius
VSK5	Gewinde	Abstich	Stechen Rest	--	Gewinde-fräsen	Tasche Restmat.	Transformationen	--	Polar
VSK6	**OEM**	--	Stechdrehen	--	Gravur	Zapfen	Unterprogramm	--	Abfahren/-Anfahren
VSK7	Positionen	**OEM**	Stechdrehen Rest	**OEM**	**OEM**	Zapfen Restmat.	--	**OEM**	--
VSK8	Position wiederh.	--	>>	<<	Konturfräsen	<<	>>	<<	--

A.1.2 Liste der Einstiegssoftkeys für Fräsen

Bedienbereich Programm bei Fräsen

HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK7	HSK8
Edit	**Bohren**	**Fräsen**	**Konturfräsen**	**Drehen** (nur bei G-Code)	**Diverses**	**Simulation**	**NC Anwahl**
HSK9	HSK10	HSK11	HSK12	HSK13	HSK14	HSK15	HSK16
--	Gerade Kreis (nur bei ShopMill)	--	--	Werkstück messen	Werkzeug messen	**OEM**	--

Fräsen

In den folgenden Tabellen werden die möglichen Einstiegssoftkeys der Technologie Fräsen dargestellt. Zuordnungen einzelner Einstiegssoftkeys können systembedingt abweichen. Die angegebenen OEM-Softkeys sind für "Run MyScreens" zulässig.

programGUIDE (G-Code) Einstiegssoftkeys:

	Bohren	**Fräsen**	**Konturfräsen**		**Drehen**		**Diverses**	
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**	
VSK1	Zentrieren	Planfräsen	Kontur	--	Abspanen	Kontur	Einstellungen	--

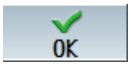
VSK2	Bohren Reiben	Tasche	Bahn	--	Einstich	Abspanen	Schwenken Ebene	Parallele Achsen
VSK3	Tieflochbohren	Zapfen Mehrkant	Vorbohren	--	Freistich	Abspanen Rest	Schwenken Werkzeug	--
VSK4	Ausdrehen	Nut	Tasche	--	Gewinde	Stechen	High Speed Settings	--
VSK5	Gewinde	Gewindefräsen	Tasche Restmat.	--	Abstich	Stechen Rest	--	--
VSK6	**OEM**	Gravur	Zapfen	--	--	Stechdrehen	Unterprogramm	--
VSK7	Positionen	**OEM**	Zapfen Restmat.	**OEM**	**OEM**	Stechdrehen Rest	--	**OEM**
VSK8	Position wiederh.	--	>>	<<	Konturdrehen	<<	>>	<<

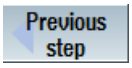
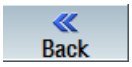
ShopMill Einstiegssoftkeys:

	Bohren	**Fräsen**	**Konturfräsen**		**Diverses**		**Gerade Kreis**
	HSK2	**HSK3**	**HSK4**		**HSK6**		**HSK10**
VSK1	Zentrieren	Planfräsen	Neue Kontur	--	Einstellungen	--	Werkzeug
VSK2	Bohren Reiben	Tasche	Bahn	--	Schwenken Ebene	Parallele Achsen	Gerade
VSK3	Tieflochbohren	Zapfen Mehrkant	Vorbohren	--	Schwenken Werkzeug	Programm wiederhol.	Kreis Mittelp.
VSK4	Ausdrehen	Nut	Tasche	--	High Speed Settings	--	Kreis Radius
VSK5	Gewinde	Gewindefräsen	Tasche Restmat.	--	Transformationen	--	Helix
VSK6	**OEM**	Gravur	Zapfen	--	Unterprogramm	--	Polar
VSK7	Positionen	**OEM**	Zapfen Restmat.	**OEM**	--	**OEM**	--
VSK8	Position wiederh.	--	>>	<<	>>	<<	--

A.2 Liste der vordefinierten Softkeys

Tabelle A-1 Vordefinierte Softkeys

Name	Softkey
SOFTKEY_OK	
SOFTKEY_CANCEL	
SOFTKEY_APPLY	
SOFTKEY_MORE	

Name	Softkey
SOFTKEY_BACK	
SOFTKEY_ASSISTANT_NEXT	
SOFTKEY_ASSISTANT_PREVIOUS	
SOFTKEY_NAV_BACK	

A.3 Liste der Zugriffsstufen

Die verschiedenen Zugriffsstufen haben folgende Bedeutung::

Schutzstufe	Verriegelt durch	Bereich
ac0	reserviert für Siemens	
ac1	Kennwort	Maschinenhersteller
ac2	Kennwort	Service
ac3	Kennwort	Anwender
ac4	Schlüsselschalter Stellung 3	Programmierer, Einrichter
ac5	Schlüsselschalter Stellung 2	qualifizierter Bediener
ac6	Schlüsselschalter Stellung 1	ausgebildeter Bediener
ac7	Schlüsselschalter Stellung 0	angelernter Bediener

A.4 Liste der Farben

Systemfarben

Für die Dialog-Projektierung steht eine einheitliche Farbtabelle zur Verfügung (Teilmenge der jeweiligen Standard-Farben). Für ein Element (Text, Eingabefeld, Hintergrund, etc.) kann eine der folgenden Farben ausgewählt werden, die zwischen 0 und 133 zu finden sind.

Alternativ zu den vordefinierten Farben können Sie Farben auch als RGB-Werte ("#RRGGBB") angeben.

Index	Piktogramm	Farbe	Farbbeschreibung
1		schwarz	
2		orange	
3		dunkelgrün	
4		hellgrau	

Index	Piktogramm	Farbe	Farbbeschreibung
5		dunkelgrau	
6		blau	
7		rot	
8		braun	
9		gelb	
10		weiß	
126		schwarz	Textfarbe eines Ein-/Ausgabe-Feldes, welches aktuell den Fokus hat
127		hellorange	Hintergrundfarbe eines Ein-/Ausgabe-Feldes, welches aktuell den Fokus hat
128		orange	Systemfarbe Focus
129		hellgrau	Hintergrundfarbe
130		blau	Header-Farbe (aktiv)
131		schwarz	Header-Schriftfarbe (aktiv)
132		türkis	Hintergrundfarbe eines Toggle-Feldes
133		hellblau	Hintergrundfarbe Listenbox

A.5 Liste der Sprachkennzeichen im Dateinamen

Unterstützte Sprachen

Standard-Sprachen:

Sprache	Abkürzung im Dateinamen
Chinesisch vereinfacht	chs
Deutsch	deu
Englisch	eng
Französisch	fra
Italienisch	ita
Spanisch	esp

Weitere Sprachen:

Sprache	Abkürzung im Dateinamen
Chinesisch traditionell	cht
Dänisch	dan
Finnisch	fin
Indonesisch	ind
Japanisch	jpn
Koreanisch	kor
Malaiisch	msl
Niederländisch	nld
Polnisch	plk
Portugiesisch (Brasilien)	ptb
Rumänisch	rom
Russisch	rus
Schwedisch	sve
Slowakisch	sky
Slowenisch	slv
Thailändisch	tha
Tschechisch	csy
Türkisch	trk
Ungarisch	hun
Vietnamesisch	vit

A.6 Liste der zugänglichen Systemvariablen

Literatur

Listenhandbuch Systemvariablen /PGAsl/

A.7 Verhalten beim Öffnen des Dialogs (Attribut CB)

Die folgende Tabelle gibt Ihnen eine Übersicht, unter welchen Bedingungen die CHANGE-Methode aufgerufen wird.

Für das Attribut CB gilt:

CB0

Die CHANGE-Methode wird mit dem Aufblenden der Maske angestoßen, wenn die Variable zu dem Zeitpunkt einen gültigen Wert hat (z.B. durch Vorbelegung oder NC/PLC-Variable).

CB1 (default)

Der CHANGE-Methode wird mit dem Aufblenden der Maske nicht explizit angestoßen. Hat die Variable eine projektierte NC/PLC-Variable, dann wird natürlich trotzdem die CHANGE-Methode aufgerufen.

Bedingung				**Reaktion**
Typ	**System- oder Anwendervariable**	**Vorbelegung**	**CHANGE-Methode abarbeiten**	
E-/A-Feld	ja	ja	ja	Durch die projektierte NC/PLC-Variable ergibt sich immer mindestens ein automatischer Aufruf der CHANGE-Methode mit dem aktuellen Wert der NC/PLC-Variable. Die Vorbelegung von CB hat keine Auswirkung.
			nein	
		nein	ja	
			nein	
	nein	ja	ja	Die CHANGE-Methode wird mit dem Wert der Vorbelegung aufgerufen.
			nein	Die CHANGE-Methode wird nicht aufgerufen.
		nein	ja	Kein Aufruf, da kein gültiger Wert vorhanden ist um eine CHANGE-Methode aufzurufen.
			nein	
Toggle	ja	ja	ja	Durch die projektierte NC/PLC-Variable ergibt sich immer mindestens ein automatischer Aufruf der CHANGE-Methode mit dem aktuellen Wert der NC/PLC-Variable. Die Vorbelegung von CB hat keine Auswirkung.
			nein	
		nein	ja	
			nein	
	nein	ja	ja	Die CHANGE-Methode wird mit dem Wert der Vorbelegung aufgerufen.
			nein	Die CHANGE-Methode wird nicht aufgerufen.
		nein (per default wird mit dem ersten Wert aus der Toggle-Liste vorbelegt)	ja	Die CHANGE-Methode wird mit dem ersten Wert der Toggle-Liste als Vorbelegung aufgerufen.
			nein	Die CHANGE-Methode wird nicht aufgerufen.

Tipps und Tricks

B

B.1 Allgemeine Tipps

- Nach Möglichkeit die BTSS-Variante bei Systemvariablen ($-Variablen) verwenden.
  Grund:
  Damit vermeiden Sie die unter Umständen aufwändige Namensauflösung.
  Beispiel:
  Anstatt "$R[10]“ besser "/Channel/Parameter/R[u1,10]“ verwenden.
- Zyklisches (gleichartiges) Setzen z. B. der Masken- und Variablen-Eigenschaft HLP vermeiden. Der Vorher zu setzende Wert mit dem aktuellen Wert vergleichen und erst bei einer Abweichung diesen wirklich setzen.
  Grund:
  Aufwändiges Suchen der auflösungsabhängigen Hilfebilder vermeiden.
  Beispiel:

```
DEF MyVar=(R3///,"X1",,"mm"/WR1//"$AA_IM[0]")
CHANGE(MyVar)
   IF MyVar.VAL < 100
      HLP="mypic1.png"
   ELSE
      HLP="mypic2.png"
   ENDIF
END_CHANGE
```

Empfehlung:

```
CHANGE(MyVar)
   IF MyVar.VAL < 100
      IF HLP <> "mypic1.png"
         HLP="mypic1.png"
      ENDIF
   ELSE
      IF HLP <> "smypic2.png"
         HLP="mypic2.png"
      ENDIF
   ENDIF
END_CHANGE
```

- Mehrere aufeinanderfolgende RNP()-Funktionen durch eine MRNP()-Funktion ersetzen.
  Mehrere aufeinanderfolgende RDOP()-Funktionen durch eine MRDOP()-Funktion ersetzen.
  Grund:
  Reduktion der Kommunikationslast und Steigerung der Performance.
- Antriebsparameter nicht schneller als im 1-Sekunden-Takt lesen, besser langsamer.
  Grund: Die Kommunikation zu den Antrieben kann sonst gegebenenfalls extrem gestört werden oder gar Ausfälle verursachen.
- Aufeinanderfolgende Rechenoperationen mit System- oder Anwendervariablen-angebundenen Dialogvariablen vermeiden. Dafür z. B. Register (REG[x]) oder (unsichtbare) Hilfsvariablen verwenden.
  Grund: Jede Zuweisung eines Wertes bewirkt auch ein Schreiben in die angebundene System- oder Anwendervariable.
- Gleichartiger Code, der in unterschiedlichen Blöcken verwendet wird, sollte aus Gründen der Übersichtlichkeit, Wartbarkeit und Performance (beim Aufblenden der Maske) in einen SUB-Block zusammengefasst werden. Dieser kann dann an den entsprechenden Stellen mit der Funktion CALL() aufgerufen werden.
- Durch Beobachtung der CPU-Auslastung in der Dialogzeile (Einstellung in slguiconfig.xml) kann untersucht werden, welche Änderungen an der Maske sich wie stark auf die Performance auswirken.

B.2 Tipps zum Debuggen

- Verwenden der Funktion DEBUG() zu Diagnosezwecken. Mit Aufruf der Funktion DEBUG() wird der übergebene String in die "easyscreen_log.txt" geschrieben. Auch die Ausgabe von Informationen in die Dialogzeile durch die Funktion DLGL() kann hilfreich sein.
  Nach Beendigung der Masken-Entwicklung sollten diese Funktion aus Performance-Gründen jedoch wieder entfernt oder auskommentiert werden.
- Die Logdatei "easyscreen_log.txt" sollte nach Fertigstellung der Entwicklung einer Maske immer ohne Eintrag sein.

B.3 Tipps zur CHANGE-Methode

- CHANGE-Methoden immer nur sehr kurz und klein halten, insbesondere bei solchen, deren Variablen an eine System- oder Anwendervariable angebunden sind und sich hochfrequent ändert.
  Grund:
  Steigerung der Performance des Maske.
- Nach Möglichkeit keine RNP()-Funktionen in CHANGE-Methoden projektieren. Stattdessen besser parallel eine unsichtbare Variable mit der zu lesenden System- oder Anwendervariable anlegen und diese dann verwenden.
  Grund:
  Mit jedem Aufruf würde zwangsweise eine RNP()-Funktion abgesetzt werden. Im anderen Fall würde einfach nur auf den ohnehin schon vorhandenen aktuellen Wert zugegriffen werden.

Beispiel:

Hier wird mit jeder Änderung der Achsbewegung per RNP() eine Namensauflösung gemacht, um ein kanalspezifisches Maschinendatum zu lesen:

```
DEF AXIS_POSITION_X =  (R///,""///"$AA_IM[X]")

CHANGE(AXIS_POSITION_X)
      DLGL("Axis """ << RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]") << """
has moved: "
<< AXIS_POSITION_X)
END_CHANGE
```

Mit Hilfe einer unsichtbaren Variable das kanalspezifische Maschinendatum aktuell halten, jede Wertänderung in eine temporäre Variable z. B. Register umkopieren.

Diese temporäre Variable kann dann in der CHANGE-Methode der Wertänderung der Achsposition verwendet werden, ohne jedes Mal eine Namensauflösung des Maschinendatums und den anschließenden Lesezugriff zu machen:

```
DEF AXIS_POSITION_X =  (R///,""///"$AA_IM[X]")
DEF AXIS_NAME_X = (S///,""/WR0//"$MC_AXCONF_GEOAX_NAME_TAB[0]")

CHANGE(AXIS_NAME_X)
      REG[0] = AXIS_NAME_X
END_CHANGE

CHANGE(AXIS_POSITION_X1)
   DLGL("Axis """ << REG[0] << """ has moved " << AXIS_POSITION_X)
END_CHANGE
```

- Die Aktualisierungsrate, und damit die Abarbeitung der zugehörigen CHANGE-Methode von Variablen, die an System- oder Anwendervariable mit einer hochfrequenten Wertänderung angebunden sind, kann unter Nutzung der Variablen-Eigenschaft UR reduziert werden, z. B. Variable, die an die Achsistwerte gekoppelt ist.
  Grund:
  Dadurch wird bei Wertänderung die zugehörige CHANGE-Methode in einem festen vorgegebenen Raster abgearbeitet.

B.4 Tipps zu DO-LOOP Schleifen

Da Schleifen, je nach Projektierung, die Performance von SINUMERIK Operate beeinträchtigen können, sollten Sie diese mit Bedacht einsetzen und nach Möglichkeit auf zeitintensive Aktionen verzichten.

Ebenso sollte als Laufvariable z. B. ein Register (REG[]) eingesetzt werden, da normale Anzeigevariablen (insbesondere solche mit BTSS-Anbindung) durch die extrem häufige Aktualisierungen bzw. Schreibvorgänge die Performance beeinträchtigen können.

Beachten Sie, dass die Anzahl der "am Stück" bearbeiteten Skriptzeilen auf derzeit 10.000 beschränkt ist. Wird diese Anzahl erreicht, dann wird die Skript-Bearbeitung mit einem entsprechenden Eintrag in der "easyscreen_log.txt" abgebrochen.

Gründe:

- Verhindern von Endlosschleifen
- "Run MyScreens" muss bedienbar bleiben

Animierte Elemente

C

C.1 Einleitung

Einleitung

Das Handbuch beschreibt das Arbeiten mit dem X3D-Viewer, mit dessen Hilfe Sie bewegte und unbewegte grafische Szenen (Animierte Elemente) – im folgenden als Hilfebilder bezeichnet – in die grafische Oberfläche des SINUMERIK Operate ab der Version V4.7 SP1 integrieren können.

Sie haben die Möglichkeit, in kontextbezogenen Hilfebildern gezielt Bewegungsabläufe und Parameter darzustellen. Damit können Sie die Bedienbarkeit von Anwendungen verbessern und die Benutzerschnittstelle ansprechender gestalten.

Die Umsetzung von der Idee bis zum fertigen Hilfebild umfasst folgende Arbeitsschritte:

- Erstellung von grafischen Elementen und 3D-Modellen für die Hilfebilder, die letztlich im X3D-Viewer zu sehen sein werden (siehe Kapitel Modellierung (Seite 316)).
- Erstellung der Szenenbeschreibungsdatei, in der die Modelldaten der Grafikdatei den gezielt darzustellende Szenen und Animationen zuordnet werden (siehe Kapitel Aufbau der Szenenbeschreibungsdatei (Seite 323)).
- Konvertierung der X3D- und XML-Dateien in HMI-Dateien (siehe Kapitel Konvertierung in hmi-Datei (Seite 328)).
- C++-Integration des X3D-Viewers in die eigene Anwendung (siehe Kapitel Implementierungsbeispiel (Seite 330)).
- Anwendung in Run MyScreens (siehe Kapitel Anzeige in Run MyScreens (Seite 331)).

Übersicht der Dateiformate

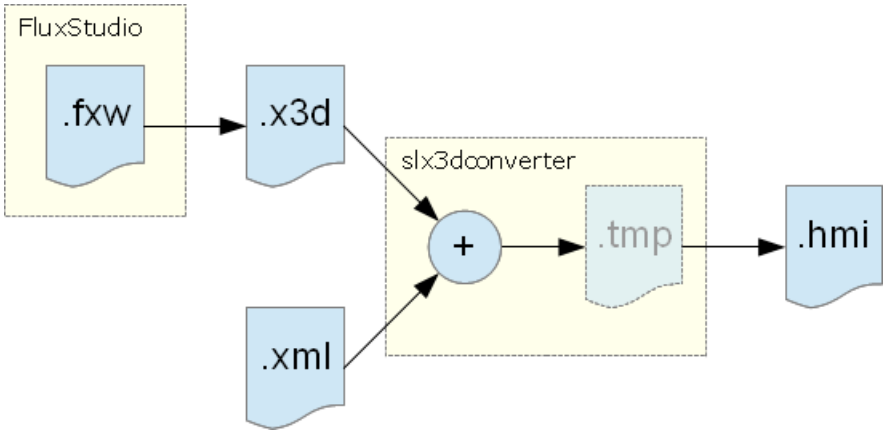


Bild C-1 Dateiformate

Dateiformat	Beschreibung
.fxw	Grafikdatei von Vivaty Studio
.x3d	Modelldatei im Austauschformat
.xml	Definitionsdatei für Animationen und Szenen
.hmi	Ausgabedatei für den X3D-Viewer

C.2 Modellierung

C.2.1 Voraussetzungen

Das Ziel der Modellierung ist, eine Datei mit den erzeugten 3D-Modellen in dem Dateiformat .x3d, einem offiziellen Standard für 3D-Inhalte, zu erstellen.

Die Modellierung erfolgt im **Vivaty Studio**. Das Vivaty Studio ist ein frei verfügbares 3D Modellierungswerkzeug mit vielseitigen Export- und Importmöglichkeit.

Voraussetzungen

- Sie haben Vivaty Studio, Version 1.0 installiert.
- Sie sind mit der Modellierung und dem Umgang mit dem Vivaty Studio vertraut.
- Auf das Dateiformat .x3d wird in dieser Beschreibung nicht weiter eingegangen, entsprechendes Wissen wird vorausgesetzt.

Hinweis

Sie können das Vivaty Studio im Internet unter dem Link (https://www.web3d.org/projects/vivaty-studio) herunterladen.

C.2.2 Regeln für die Modellierung

Beachten Sie bei der Modellierung bitte folgende Regeln. Die Einhaltung dieser Regeln ist für die anschließende Aufbereitung der Hilfebilddatei erforderlich.

Regeln für die Modellierung

1. Der TimeSensor muss den Namen „Animation“ tragen.
2. Alle zusammengehörigen Elemente müssen einer Gruppe zugeordnet werden.
3. Alle Gruppen müssen auf folgende Position gesetzt werden:
   x = 0, y = 0, z = 0

4. Um die Gruppen unsichtbar zu machen, werden die Translationswerte wie folgt gesetzt: x = 1000000, y = 1000000 und z = 1000000
5. Folgende Elemente müssen in den Vivaty Studio-Grafikmodellen und in den XML-Szenendefinitionen (vergleich hierzu Kapitel Überblick (Seite 323)) die selben Namen tragen:
   - Werkzeug
   - Kameras
6. Um eine .x3d-Datei zu erstellen, müssen Sie im Dialog „Export Options" folgende Einstellung vornehmen:

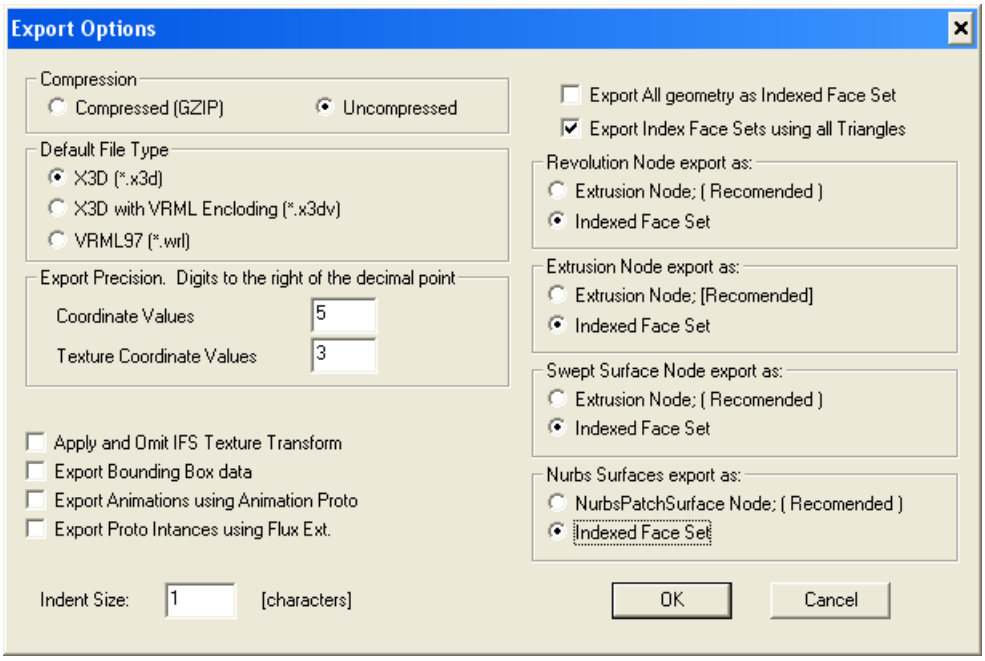


Bild C-2 Einstellungen im Dialog Export Options

7. Für Texte empfehlen wir folgende Einstellungen (siehe auch folgenden Dialog):
   - Texte müssen immer mittig ausgerichtet werden.
   - Die Größe des Textes sollte 0.2 sein. Hierdurch lässt sich der Text gut positionieren. Die Textausgabe durch den X3D-Viewer erfolgt unabhängig von diesem Wert in der Schriftgröße der Bedienoberfläche.

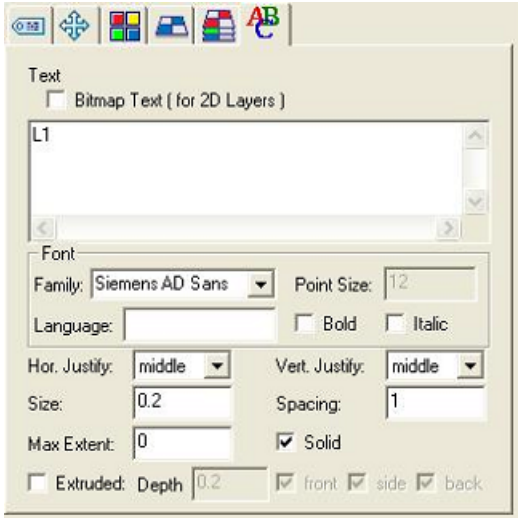


Bild C-3 Einstellungen für Text

8. Um eine Schraffur zur erstellen, verwenden Sie die Datei schnitt.png als Textur. Die Linien gehen immer von links unten nach rechts oben in einem Winkel von 45°. Die Skalierung sollten Sie in beiden Dimensionen auf 3 einstellen. Die Rotation ist von der Bauform abhängig und muss manuell angepasst werden.

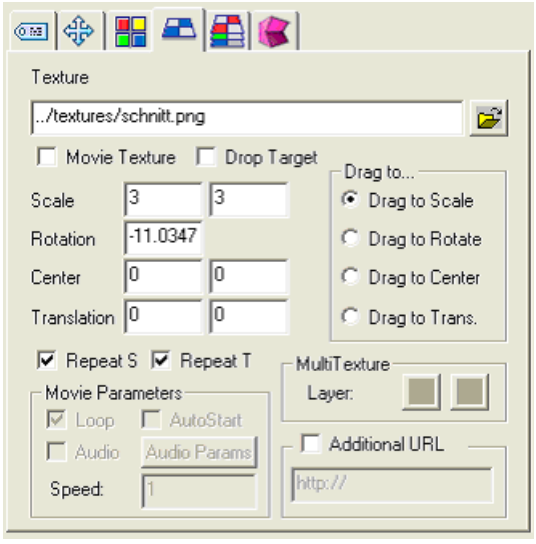


Bild C-4 Einstellungen für Schraffur

9. Die folgenden Größen/Maße empfehlen wir für die Rohteile:
   - Zylinder für Drehbearbeitung:
     Länge = 5,5 ; Radius = 1,4 (vgl. Vorlage turning_blank.fxw)
   - Quader für Fräsbearbeitung:
     x = 4,75 ; y = 3 ; z = 3 (vgl. Vorlage milling_blank.fxw)

Siehe auch

XML Befehle (Seite 323)

C.2.3 Importieren von Grafiken (Modellen)

Im Flux Studio können Sie externe Grafiken (Modelle) importieren. Häufig gebrauchte Modelle, wie zum Beispiel Werkzeuge, können Sie als .x3d- oder .hmi-Dateien zentral ablegen und als fertige Elemente in anderen Hilfebildern wiederverwenden. Eine Liste der mitgelieferten Modellierungsvorlagen finden Sie in Kapitel Vorlagen zur Modellierung (Seite 321).

Komplexe Objekte können auch mit anderen Modellierungswerkzeugen, wie zum Beispiel Cinema4D, erstellt und anschließend importiert werden.

Im Folgenden ist beschrieben, wie Sie diese Modelle wiederverwenden können.

Import als Inline

Für den Import von .x3d-Dateien bietet das Flux Studio das Objekt "Inline". Damit können externe Elemente eingebunden werden, ohne die 3D-Daten dieser Modelle zu vervielfachen.

Über den Menüpunkt **Create** -> **Create Inline** wird das Objekt eingefügt. In den Objekt-Eigenschaften geben Sie dann den Dateiname an.

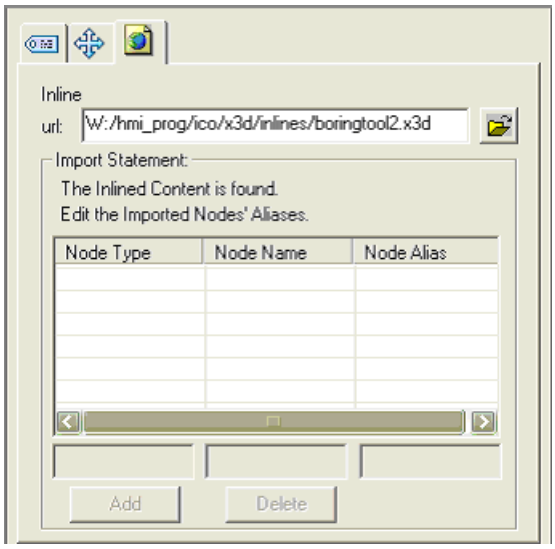


Bild C-5 Import über Objekt "Inline"

Import von 3D-Modelldaten

Für den Import von Modelldaten aus einer Datei gehen Sie wie folgt vor.

1. Über den Menüpunkt **File** -> **Import Other Format** öffnen Sie den Dialog "Flux Studio Accutrans3D Translation Utility".
2. Über das Auswahlfeld "File Types" wählen Sie das entsprechende Format aus. Um zum Beispiel eine Cinema4D–Datei zu importieren, müssen Sie den Dateityp "3D Studio" auswählen.
3. Wählen Sie anschließend über **Select Files** die gewünsche Datei aus und starten Sie den Import.

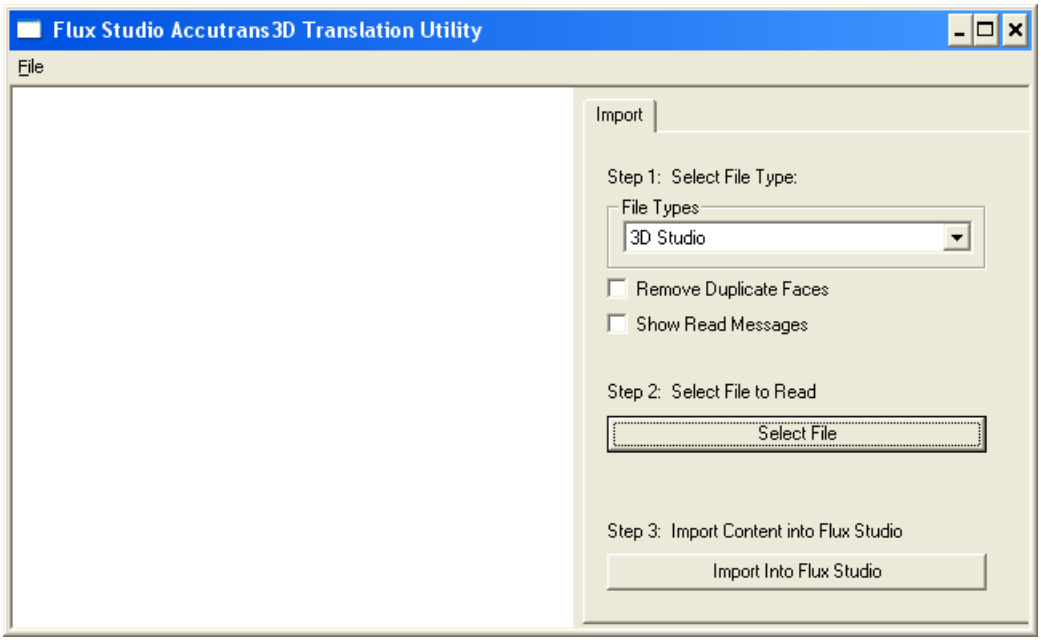


Bild C-6 Import 3D-Modelldaten

Die Modelldaten aus der Cinema4D-Datei werden als komplette Gruppe eingefügt. Alle Kameras und weitere Informationen, die mit importiert werden, sollten gelöscht werden. Nur die grafischen Elemente der Cinema4D-Datei werden benötigt.

C.2.4 Vorlagen zur Modellierung

In diesem Kapitel finden Sie eine Auflistung von grafischen Elementen als Grundlage für die Gestaltung, Farben und Größen. Für ein einheitliches "look and feel" in der Bedienoberfläche empfehlen wir, für die Erstellung von eigenen Grafiken, den Stil der Vorlagen beziehungsweise die Vorlagen selbst zu verwenden.

Allgemein

Vorlage	Beschreibung
intersection_texture.png	Textur für eine Schnittfläche
rapid_traverse_line_hori.fxw	Horizontale Linie zur Verdeutlichung des Werkzeugverfahrweges im Eilgang
rapid_traverse_line_vert.fxw	Vertikale Linie zur Verdeutlichung des Werkzeugverfahrweges im Eilgang
feed_traverse_line_hori.fxw	Horizontale Linie zur Verdeutlichung des Werkzeugweges im Vorschub
feed_traverse_line_vert.fxw	Vertikale Linie zur Verdeutlichung des Werkzeugweges im Vorschub
dimensioning_lines_hori.fxw	Horizontale Maßhilfslinien
dimensioning_lines_vert.fxw	Vertikale Maßhilfslinien
z1_inc.fxw	Horizontale Maßlinie mit dem Bezeichner Z1
x1_inc.fxw	Vertikale Maßlinie mit dem Bezeichner X1
3d_coordinate_origin.fxw	3D Koordinatenkreuz
3d_zero_point.fxw	3D Nullpunkt

Drehen

Vorlage	Beschreibung
turning_blank.fxw	Das Rohteil für die Technologie Drehen: ein einfacher Zylinder
turning_centerline.fxw	Mittellinie
turning_centerpoint.fxw	Koordinatenkreuz zur Darstellung des Nullpunkts im WKS
turning_refpoint.fxw	Bezugspunkt für z. B. eine Bearbeitung
turning_machining_area.fxw	Begrenzungslinien für die Bearbeitungsfläche

Fräsen

Vorlage	Beschreibung
milling_blank.fxw	Das Rohteil für die Technologie Fräsen: ein einfacher Quader
milling_centerline.fxw	Mittellinie
milling_refpoint.fxw	Bezugspunkt für z. B. eine Bearbeitung

C.3 XML Befehle

C.3.1 Überblick

Damit der X3D-Viewer auf die Grafiken zugreifen kann, wird die Szenenbeschreibungsdatei benötigt (*.xml). In der Szenenbeschreibungsdatei ordnen Sie die Szenennamen, die aus der Projektierung aufgerufen werden, den Zeiten in der *.x3d-Datei (TimeSensor), an denen sich die Grafiken befinden, zu.

C.3.2 Aufbau der Szenenbeschreibungsdatei

Im Folgenden wird der Aufbau der Szenenbeschreibungdatei dargestellt und jeweiligen XML-Befehle erläutert:

Aufbau und Beschreibung

```
<!-- Behandlung von ebenenabhängigen Texten bei der Drehbearbeitung -->
<TextPlane plane="G18_ZXY" />
```

Beschreibung

<TextPlane plane="G18_ZXY" />	Behandlung von ebenenabhängigen Texten (Achsnamen) spezifisch für Drehbearbeitungen. Wird z. B. ein Text mit der Kennung "Z" verwendet (bei G18 erste Geo-Achse), dann wird der entsprechende Geo-Achsbezeichner der Steuerung im Hilfebild dargestellt (z. B. "Z").

```
<!--Textausrichtung -->
<TextPosition center='true' />
```

Beschreibung

<TextPosition center='true' />	Kennzeichnung, dass die Texte zentriert ausgerichtet sind. Dies ist für die Darstellung der Texte in gedrehten Bildern von Bedeutung. Es wird empfohlen, diese Einstellung zu verwenden.

<!--Werkzeuggeschwindigkeit anpassen -->

```
<ToolSpeedFactors planeSpeedFactor="0.4" rapidSpeedFactor="0.7"
reducedSpeedFac¬tor="1.0"/>
```

Beschreibung

<ToolSpeedFactors	Sollen die Werkzeuggeschwindigkeiten modifiziert werden, kann dieser Eintrag hinzugefügt werden.
planeSpeedFactor="0.4"	Faktor für die Vorschubgeschwindigkeit (0.4 = 40%)
rapidSpeedFactor="0.7"	Faktor für die Eilganggeschwindigkeit (0.7 = 70%)
reducedSpeedFactor="0.1" />	Faktor für eine dritte Geschwindigkeit (0.1 = 10%)

<!--Definition einer Animation-->

```
<SceneKey name='BoringAnimation' masterRotationSpeed="-64.0"
maxRotAngle="45.0" begin-Time='0.110 endTime='0.118 view='camiso'
speedMaster='boringtool' Type="VIEW_3D_TURN_CYL">
```

Beschreibung

<SceneKey name='BoringAnimation'	Name der Animation
masterRotationSpeed="-64.0"	Drehrichtung und Drehgeschwindigkeit des Werkzeugs. Die Einstellung ist optional.
maxRotAngle="45.0"	Vermeidung des Aliaseffekts (Werkzeug scheint in die falschre Richtung zu drehen). Der Wert ist üblicherweise ein Viertel der Symmetrie des Werkzeugs. Die Einstellung ist optional.
beginTime='0.110	Startzeitpunkt der Animation auf dem Timesensor
endTime='0.118'	Endzeitpunkt der Animation auf dem Timesensor
view='camiso'	Kamera, die für die Animation verwendet wird
speedMaster='boringtool'	Name des Werkzeugs für die Animation
type="VIEW_3D_TURN_CYL">	View-Typ legt die Ansicht fest (siehe Kapitel View-Typ (Seite 327))

<!-- Elemente -->

```
<Element name='1' time='0.110' feed='rapid' dwell=’2’/>
<Element name='2' time='0.112' feed='rapid'/>
<Element name='3' time='0.114' feed='rapid'/>
<Element name='4' time='0.116' feed='plane'/>
<Element name='5' time='0.118' feed='plane'/>
```

Beschreibung

<Element name='1'	Gibt an, dass es sich um das erste Element der Animation handelt.
time='0.110'	Zeitpunkt des ersten Elements auf dem Timesensor
feed='rapid'	Verfahrgeschwindigkeit des Elements plane = Bearbeitungsvorschub rapid = Eilgang reduced = reduzierte Geschwindigkeit, langsamer als "plane"
dwell=’2’/>	Pause in der Verfahrbewegung der Animation. Ein drehendes Werkzeug bleibt in Rotation.

<!-- Ende der Animationsdefinition -->

```
</SceneKey>
```

Beschreibung

</SceneKey>	Ende der Animationsdefinition

<!-- vorhandene Animation als Quelle einer neuen Animation -->

```
<SceneKey source=''BoringAnimation' name=''BoringAnimationRear' mirror='
MirrorScreenX ' />
```

Beschreibung

<SceneKey source="BoringAnimation'	Name einer Animation, die als Quelle eine andere Animation verwenden soll.
name="BoringAnimationRear'	Name der neuen Animation, basierend auf der Quelle
mirror=' MirrorScreenX ' />	Spiegelung in Richtung der X-Achse, Ausgabe erfolgt in der neuen Animation.

```
<!-- Definition eines statischen Szene (Bild), (4 Beispiele) -->
<SceneKey name='Default' time='0.026' view='camiso'
type="VIEW_3D_DRILL_CUT"/>
<SceneKey name='Cut' time='0.046' view='camside' type="VIEW_SIDE"/>
<SceneKey name='Z1ink' time='0.056' view='camside' type="VIEW_SIDE"
 highLightedGroup='dad_Z1ink'/>
<SceneKey name='Z1abs' time='0.066' view='camside' type="VIEW_SIDE"
 highLightedGroup='dad_Z1abs'/>
```

Beschreibung

<SceneKeyname=’Z1abs'	Name des Bildes
time='0.066'	Zeitpunkt des Bilder auf dem Timesensor
view='camside'	Kamera, die für das Bild verwendet wird
type="VIEW_SIDE"	View-Typ legt die Ansicht fest (siehe Kapitel View-Typ (Seite 327))
highLightedGroup='dad_Z1abs'/>	Name der Gruppe, die hervorgehoben werden soll. Die Gruppe wurde im FluxStudio "Z1abs" genannt. Der Zusatz „dad_“ wird von Flux automatisch vorangestellt.

```
<!-- end of xml file –!>
```

C.3.3 Spiegelungen und Rotationen

Mit dem Befehl **mirror** wird die Spiegelung, beziehungsweise die Drehung des Bildes/Modells festgelegt.

Beschreibung

mirror="RotateScreenX"	Das Bild wird um die X-Achse gedreht
mirror="RotateScreenY"	Das Bild wird um die Y-Achse gedreht
mirror="RotateScreenZ"	Das Bild wird um die Z-Achse gedreht
mirror="MirrorScreenX”	Das Bild wird in Richtung der X-Achse gespiegelt
mirror="MirrorScreenY"	Das Bild wird in Richtung der Y-Achse gespiegelt
mirror="MirrorScreenZ"	Das Bild wird in Richtung der Z-Achse gespiegelt
mirror="RotatePieceX”	Das Modell wird um die X-Achse gedreht
mirror="RotatePieceY”	Das Modell wird um die Y-Achse gedreht
mirror="RotatePieceZ”	Das Modell wird um die Z-Achse gedreht
mirror="MirrorPieceX”	Das Modell wird in Richtung der X-Achse gespiegelt

mirror="MirrorPieceY"	Das Modell wird in Richtung der Y-Achse gespiegelt
mirror="MirrorPieceZ"	Das Modell wird in Richtung der Z-Achse gespiegelt

Alle Dreh- und Spiegelmöglichkeiten können kombiniert werden. Die Rotation erfordert die Festlegung eines Rotationswinkels.

Beispiele

mirror="RotatePieceZ=180"

mirror="RotatePieceZ=-90 MirrorScreenX"

mirror="RotateScreenY=90"

mirror="MirrorPieceX MirrorPieceY"

mirror="MirrorPieceZ RotatePieceX=-90"

C.3.4 View-Typ

Mit dem View-Typ legen Sie die Ansichten fest. Die Ansichten basieren auf Erfahrungswerten und Regeln, die zu einer sinnvollen Darstellung der Objekte führen.
Hierdurch wird sichergestellt, dass die Darstellung der Objekte passend zu der Einstellung des Koordinatensystems der Bedienoberfläche (MD 52000 $MCS_DISP_COORDINATE_SYSTEM, beziehungsweise MD 52001 $MCS_DISP_COORDINATE_SYSTEM_2) erfolgt.

Beschreibung

"VIEW_STATIC"	Direkte Ansicht ohne Umwandlung
"VIEW_3D_TURN_CYL"	3D-Ansicht für Drehbearbeitungen (Zylinder)
"VIEW_3D_MILL_CUBE"	3D-Ansicht für Fräsbearbeitungen (Quader)
"VIEW_3D_DRILL_CUT"	3D-Ansicht für Bohrbearbeitungen (Schnittdarstellung)
"VIEW_SIDE"	2D-Ansicht für Bohrbearbeitungen (Schnittdarstellung)
"VIEW_TOP_GEO_AX_1"	2D-Ansicht aus Richtung der 1. Geometrieachse
"VIEW_TOP_GEO_AX_2"	2D-Ansicht aus Richtung der 2. Geometrieachse
"VIEW_TOP_GEO_AX_3"	2D-Ansicht aus Richtung der 3. Geometrieachse

C.4 Konvertierung in hmi-Datei

Hinweis

Die X3D-Dateien inklusiv der dazu gehörenden XML-Dateien werden im Hochlauf des HMI zu HMI-Dateien konvertiert.

Für jede X3D-Datei muss eine entsprechende XML-Datei mit gleichem Namen angelegt werden. Die X3D Dateien und die XML-Dateien müssen Sie dazu im Verzeichnis `Suchpfad des HMI \ico\x3d\turning` bzw. `milling` ablegen. Das Vorgehen ist analog zu .ts Dateien.

C.5 Anzeige in Create MyHMI /3GL

C.5.1 X3D-Viewer

Um in einer eigenen OA-Anwendung gezielt Hilfebilder darzustellen, müssen Sie das X3D-Viewer-Widget in die eigene OA-Anwendung einbinden.

Das X3D-Viewer-Widget stellt Schnittstellen zur Verfügung, welche die Präsentation von X3D-Inhalte innerhalb des HMI ermöglichen.

Um grafische Szenen darzustellen, steht die Klasse SlX3dViewerWidget zur Verfügung. Die Definition der Klasse liegt in der entsprechenden Headerdatei slx3dviewerwidget.h im globalen GUI Include-Verzeichnis `\hmi_prog\gui\include`.

C.5.2 Klasse SlX3dViewerWidget

Die Klasse bietet ein flexibel einsetzbares Widget, das die Inhalte aus einer zur Laufzeit angegebenen Modelldatei autonom darstellt und gegebenenfalls die Animation ablaufen lässt.

Das Interface der Klasse besteht aus Konstruktor, Destruktor und zwei Methoden zur Steuerung der grafischen Ausgabe.

Als direkte Ableitung von der Qt-Klasse QWidget steht ein weit umfangreicheres und hier nicht weiter beschriebenes Interface zur Verfügung (z. B. show(), hide() und resize(...). Nähere Informationen hierzu finden Sie in der Qt-Dokumentation).

C.5.3 Öffentliche Methoden

SlX3dViewerWidget (QWidget* pParent = 0)

Konstruktor des X3D-Viewer-Widgets.

Parameter	Bedeutung
pParent	Der Parameter wird an den Konstruktor des Qwidget weitergereicht.

~SlX3dViewerWidget ( )

Destruktor des X3D-Viewer-Widgets

Parameter	Bedeutung
-	-

C.5.4 Öffentliche Slots

void viewSceneSlot (const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene, int nChannel, int nPlane, SlStepTechnology nTechnology)

Mit der Methode viewSceneSlot wird der X3D-Viewer angewiesen, die statische Szene rsScene und die animierte Szene rsAnimationSzene aus der Datei rsFileName zu laden und im zeitlichen Wechsel darzustellen.

Konkret wird wiederholt zunächst die statische Szene für eine feste Zeitspanne und anschließend die animierte Szene angezeigt.

Wird keine statische Szene angegeben, dann wird sofort die Animation dargestellt; entsprechend kann auch die Angabe einer animierten Szene fehlen.

Kanalnummer, Ebene und Technologie werden benutzt, um die Szenen in die richtige Position (abhängig vom eingestellten Maschinen-Koordinatensystem) zu drehen.

Parameter	Bedeutung
rsFileName	Name der Datei, in welcher die darzustellenden Szenen enthalten sind.
rsScene	Name der statischen Szene.
rsAnimationScene	Name der animierten Szene.
nChannel	Kanalnummer

Parameter	Bedeutung
nPlane	Ebene
nTechnology	Technologie Für den Aufzählungtyp SlStepTechnology sind folgende Konstanten definiert: • SL_STEP_NO_TECHNOLOGY • SL_STEP_MILLING • SL_STEP_TURNING • SL_STEP_SURFACE_GRINDING • SL_STEP_CIRCULAR_GRINDING

void viewSceneSlot (const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene)

Eine vereinfachte Form der Methode viewSceneSlot, mit der der X3D-Viewer angewiesen werden kann, die statische Szene rsScene und/oder die animierte Szene rsAnimationScene aus der Datei rsFileName zu laden und darzustellen.

Parameter	Bedeutung
rsFileName	Name der Datei, in welcher die darzustellenden Szenen enthalten sind.
rsScene	Name der statischen Szene.
rsAnimationScene	Name der animierten Szene.

C.5.5 Bibliotheken

Um den X3D-Viewer in eigenen Projekten einsetzen zu können, muss die Liste der Bibliotheksabhängigkeiten um den Eintrag 'slx3dviewer.lib' erweitert werden.

C.5.6 Implementierungsbeispiel

Ein Implementierungsbeispiel finden Sie im Paket Create MyHMI/3GL unter **\examples \GUIFrameWork\SlExGuiX3D**.

C.5.7 Maschinendaten

Anzeige-MD 9104 $MM_ANIMATION_TIME_DELAY

Zeitverzögerung bis zum Einsetzen der Animation bei Hilfebildern in Sekunden. Bei Hilfebildern, die ausschließlich animiert sind, wirkt die Einstellung nicht.

Die Einstellung wirkt global für alle Animationen des SINUMERIK Operate.

C.5.8 Hinweise zur Anwendung

- Die Animation wird unterbrochen, wenn das X3D-Viewer-Widget abgeblendet wird. Es besteht keine Notwendigkeit, eine animationslose Szenenauswahl für diesen Fall anzuwählen.
- Eine häufige oder vielfache Instanziierung des X3D-Viewer-Widget sollten Sie im Hinblick auf Performance und Speicherbedarf vermeiden. Für solche Anwendungsszenarien wird die Nutzung (Implementierung) eines X3D-Viewer-Singletons empfohlen.
- Der X3D-Viewer erlaubt auch den Einsatz in Signal-Slot-Konzepten.
- Sollte ein Fehler auftreten (z. B. Datei nicht gefunden oder unbekannter Szenenname), dann blendet das X3D-Viewer-Widget einen Meldungsbereich auf. Dieser Bereich wird automatisch wieder abgeblendet, sobald eine andere Hilfebilddatei angesteuert wird.

C.6 Anzeige in Run MyScreens

Dieses Kapiel richtet sich an versierte "Run MyScreens"-Entwickler. Das notwendige Hintergrundwissen kann in der zugehörigen Dokumentation nachgelesen werden.

Zusätzlich zur Verwendung von Bildern im Format .bmp oder .png können auch animierte Hilfebilder durch den X3D-Viewer dargestellt werden.

Für die Einbindung verwenden Sie wie gewohnt die Run MyScreens-Schnittstelle für Hilfebilder.

Zur Ausgabe von Bildern im Format .bmp oder .png wird in der Definition eines Dialogs im Abschnitt Attribute die Angabe XG0 eingetragen bzw. der Parameter wird leer gelassen. Um X3D-Hilfebilder in einen Dialog einzubinden, müssen Sie in den Attributen die Angabe **XG1** eintragen.

```
//M(MASK_F_DR_O1_MAIN/$85407////52,80/XG1)
```

Zur Ansteuerung der einzelnen Hilfebilder werden die folgenden Parameter benötigt, deren Bedeutung in Kapitel 5.3 Öffentliche Slots beschrieben ist:

1. Dateiname
2. StaticScene (optional)
3. AnimationScene (optional)
4. Technologie (optional)
5. Ebene (optional)

Die Parameter werden in dieser Reihenfolge und durch Komma getrennt in einer Zeichenkette zusammengefasst.

```
Hlp = "Dateiname,StaticScene,AnimationScene,Technologie,Ebene"
```

Beispiele

- Das Default-Hilfebild kann mit der vorgegebene Variable Hlp gesetzt werden.
  Im folgenden Beispiel wird aus der Datei "MyDlgHelp.hmi" die Animation "MyAnimation" ausgegeben. Es ist keine StaticScene angegeben, d. h. es wird keine statische Szene ausgegeben. Zur Technologie und zur Ebene werden ebenfalls keine Angaben gemacht, d. h. es werden Defaultwerte verwendet.
  ```
  Hlp = "MyDlgHelp.hmi,,MyAnimation"
  ```
- Für die einzelnen Parameter der Eingabemaske kann an der zugehörigen Variablen die Eigenschaft hlp auf ein spezifisches Hilfebild gesetzt werden.
  Im folgenden Beispiel wird aus der Datei "MyDlgHelp.hmi" die statische Scene "MyParam" und die Animation "MyAnimation" in der Ebene G17 ausgegeben. Zur Technologie wird keine Angabe gemacht, d.h. es wird ein Defaultwert verwendet.
  ```
  VarMyParam.hlp = "MyDlgHelp.hmi,MyParam,MyAnimation,,G17"
  ```

Index

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z