SIEMENS

SINUMERIK 828D
SINUMERIK Integrate Run
MyScreens

Manuel de programmation

Valable pour

Logiciel CNC version 4.95

10/2020
6FC5398-4EP40-0DA0

Mentions légales

Signalétique d'avertissement

Ce manuel donne des consignes que vous devez respecter pour votre propre sécurité et pour éviter des dommages matériels. Les avertissements servant à votre sécurité personnelle sont accompagnés d'un triangle de danger, les avertissements concernant uniquement des dommages matériels sont dépourvus de ce triangle. Les avertissements sont représentés ci-après par ordre décroissant de niveau de risque.

⚠ **DANGER**
signifie que la non-application des mesures de sécurité appropriées **entraîne** la mort ou des blessures graves.

⚠ **ATTENTION**
signifie que la non-application des mesures de sécurité appropriées **peut entraîner** la mort ou des blessures graves.

⚠ **PRUDENCE**
signifie que la non-application des mesures de sécurité appropriées peut entraîner des blessures légères.

IMPORTANT
signifie que la non-application des mesures de sécurité appropriées peut entraîner un dommage matériel.

En présence de plusieurs niveaux de risque, c'est toujours l'avertissement correspondant au niveau le plus élevé qui est reproduit. Si un avertissement avec triangle de danger prévient des risques de dommages corporels, le même avertissement peut aussi contenir un avis de mise en garde contre des dommages matériels.

Personnes qualifiées

L'appareil/le système décrit dans cette documentation ne doit être manipulé que par du **personnel qualifié** pour chaque tâche spécifique. La documentation relative à cette tâche doit être observée, en particulier les consignes de sécurité et avertissements. Les personnes qualifiées sont, en raison de leur formation et de leur expérience, en mesure de reconnaître les risques liés au maniement de ce produit / système et de les éviter.

Utilisation des produits Siemens conforme à leur destination

Tenez compte des points suivants:

⚠ **ATTENTION**
Les produits Siemens ne doivent être utilisés que pour les cas d'application prévus dans le catalogue et dans la documentation technique correspondante. S'ils sont utilisés en liaison avec des produits et composants d'autres marques, ceux-ci doivent être recommandés ou agréés par Siemens. Le fonctionnement correct et sûr des produits suppose un transport, un entreposage, une mise en place, un montage, une mise en service, une utilisation et une maintenance dans les règles de l'art. Il faut respecter les conditions d'environnement admissibles ainsi que les indications dans les documentations afférentes.

Marques de fabrique

Toutes les désignations repérées par ® sont des marques déposées de Siemens AG. Les autres désignations dans ce document peuvent être des marques dont l'utilisation par des tiers à leurs propres fins peut enfreindre les droits de leurs propriétaires respectifs.

Exclusion de responsabilité

Nous avons vérifié la conformité du contenu du présent document avec le matériel et le logiciel qui y sont décrits. Ne pouvant toutefois exclure toute divergence, nous ne pouvons pas nous porter garants de la conformité intégrale. Si l'usage de ce manuel devait révéler des erreurs, nous en tiendrons compte et apporterons les corrections nécessaires dès la prochaine édition.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
ALLEMAGNE

Numéro de référence du document: 6FC5398-4EP40-0DA0
Ⓟ 07/2021 Sous réserve de modifications

Sommaire

Introduction 1

1.1 À propos de SINUMERIK

Des machines-outils courantes CNC simples aux machines modulaires les plus sophistiquées, en passant par les machines standardisées – les commandes CNC SINUMERIK offrent la solution idéale pour chaque machine. Qu'il s'agisse de fabrication de pièces uniques ou de fabrication de masse, de pièces simples ou complexes – SINUMERIK est la solution d'automatisation homogène hautement productive pour tous les secteurs de production – de la fabrication de prototypes et d'outils jusqu'à la fabrication en grandes séries, en passant par l'usinage de moules.

Pour plus d'informations, consulter le site Internet relatif à SINUMERIK (https://www.siemens.com/sinumerik).

1.2 À propos de cette documentation

Groupe cible

Le présent document s'adresse aux programmeurs et aux ingénieurs de projet.

Objectifs

Le Manuel de programmation permet au groupe cible de créer, d'écrire, de tester des programmes et des interfaces logicielles et de supprimer des erreurs.

Version standard

La présente documentation décrit la fonctionnalité de la version standard du système. Elle peut différer de l'étendue des fonctionnalités du système livré. Pour toute précision concernant les fonctionnalités du système livré, se reporter exclusivement aux documents de commande.

Le système peut comporter des fonctions opérationnelles supplémentaires qui ne sont pas mentionnées dans cette documentation. Le client ne peut toutefois pas faire valoir de droit en liaison avec ces fonctions, que ce soit dans le cas de matériels neufs ou dans le cadre d'interventions du service après-vente.

Pour des raisons de clarté, la présente documentation peut ne pas contenir toutes les informations détaillées concernant toutes les variantes du produit. En outre, elle ne peut pas tenir compte de tous les cas possibles d'installation, d'exploitation ou de maintenance.

Les options complémentaires ou les modifications apportées par le constructeur de machines au produit sont documentées par celui-ci.

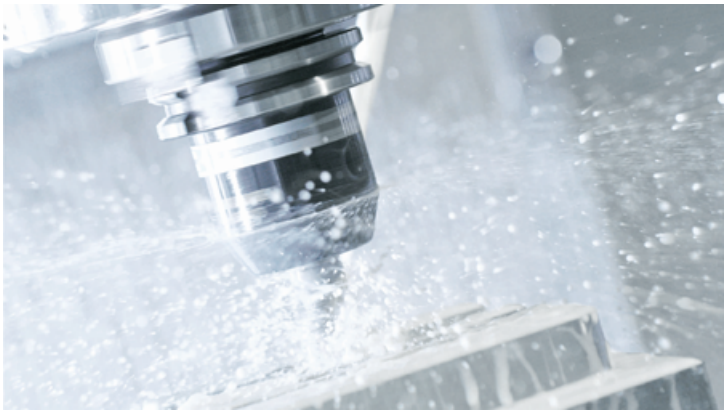
Pages Web non Siemens

Ce document peut contenir des liens vers des pages Web non Siemens. Siemens décline toute responsabilité quant au contenu de ces pages Web et ne considère pas comme siennes ces pages et leur contenu. Siemens ne contrôle pas les informations accessibles par ces pages Web et n'est pas non plus responsable du contenu et des informations qui y sont mis à disposition, leur utilisation étant aux risques et périls de l'utilisateur.

1.3 Documentation sur Internet

1.3.1 Vue d'ensemble de la documentation SINUMERIK 828D

Une documentation détaillée relative aux fonctions de SINUMERIK 828D à partir de la version 4.8 SP4 est disponible sous Vue d'ensemble de la documentation 828D (https://support.industry.siemens.com/cs/ww/en/view/109766724).



Il est possible d'afficher les documents ou de les télécharger aux formats PDF et HTML5.

La documentation est divisée en plusieurs catégories comme suit :

- Utilisateur : Commande
- Utilisateur : Programmation
- Constructeur/S.A.V. : Configuration
- Constructeur/S.A.V. : Mise en service
- Constructeur/S.A.V. : Fonctions
- Constructeur/S.A.V. : Safety Integrated
- SINUMERIK Integrate/MindApp
- Info et formation

1.3.2 Vue d'ensemble de la documentation pour les éléments de conduite SINUMERIK

Une documentation détaillée relative aux éléments de conduite SINUMERIK est disponible sous Vue d'ensemble de la documentation pour les éléments de conduite SINUMERIK (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW).

Il est possible d'afficher les documents ou de les télécharger aux formats PDF et HTML5.

La documentation est divisée en plusieurs catégories comme suit :

- Tableaux de commande
- Tableaux de commande de machine
- Machine Push Button Panel
- Unité portable/Mini-consoles
- Autres éléments de conduite

Une vue d'ensemble des documents, des contributions et des liens les plus importants relatifs à SINUMERIK se trouve sous Vue d'ensemble/page thématique SINUMERIK (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW).

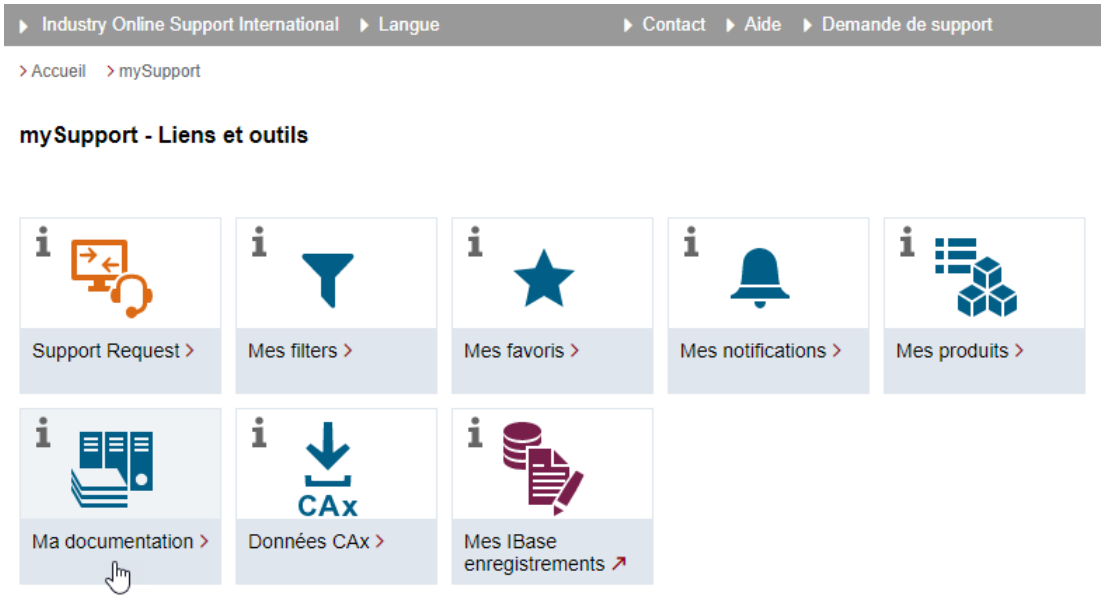
1.4 Remarques concernant la documentation technique

En cas de questions, suggestions ou corrections relatives à la documentation technique publiée dans Siemens Industry Online Support, utiliser le lien "Donner un avis" à la fin d'une contribution.

1.5 Documentation mySupport

Le système "Documentation mySupport" sur Internet permet à un utilisateur de composer sa propre documentation à partir des contenus Siemens et de l'adapter à sa documentation machine.

Démarrer l'application via le panneau "Ma documentation" sur la page d'accueil mySupport (https://support.industry.siemens.com/cs/ww/fr/my) :



L'exportation du manuel configuré est possible au format RTF, PDF ou XML.

Remarque

Les contenus Siemens qui prennent en charge l'application Documentation mySupport sont reconnaissables à la présence du lien "Configurer".

1.6 S.A.V. et assistance

Assistance produit

Pour plus d'informations sur le produit, voir sur Internet :

Assistance produit (https://support.industry.siemens.com/cs/ww/fr/)

Sous cette adresse figurent les éléments suivants :

- Informations produit actuelles (Informations sur le produit)
- FAQ (Foire aux Questions)
- Manuels
- Téléchargements
- Lettre d'information (newsletter) avec les dernières actualités sur vos produits
- Forum à destination des utilisateurs et des spécialistes pour un échange d'informations et d'expérience à l'échelle mondiale
- Interlocuteurs sur place via notre base de données d'interlocuteurs (→ "Contact")
- Informations sur le service après-vente, les réparations, les pièces de rechange et bien plus encore (→ "Services")

Assistance technique

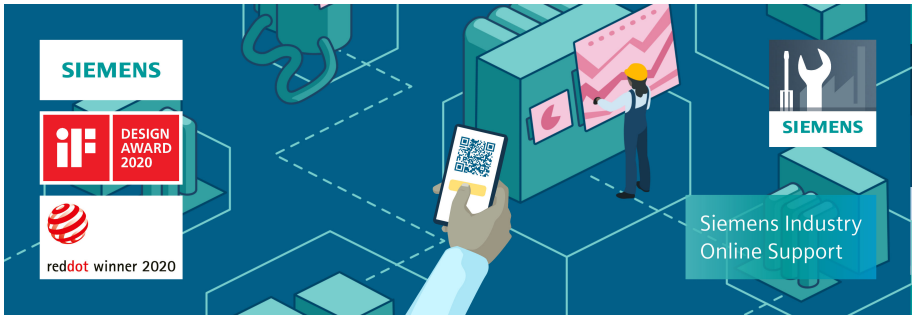
Les numéros de téléphones spécifiques à chaque pays pour l'assistance technique sont disponibles à cette adresse (https://support.industry.siemens.com/cs/ww/fr/sc/4868) dans la zone "Contact".

Pour poser une question technique, utiliser le formulaire en ligne dans la zone de demande d'assistance "Support Request".

Formation

L'adresse (https://www.siemens.com/sitrain) suivante fournit des informations sur SITRAIN. SITRAIN propose une offre de formations pour les produits, systèmes et solutions d'entraînement et d'automatisation Siemens.

Assistance Siemens pour les déplacements



L'application primée "Siemens Industry Online Support" permet d'accéder à tout moment et en tout lieu à plus de 300 000 documents relatifs aux produits Siemens Industry. L'application assiste les clients notamment dans les domaines d'utilisation suivants :

- Résolution de problèmes lors de la réalisation d'un projet
- Dépannage en cas de défauts
- Extension ou replanification d'une installation

En outre, elle donne accès à un forum technique et à d'autres contributions créées spécialement pour nos clients par nos experts :

- FAQ
- Exemples d'application
- Manuels
- Certificats
- Informations sur les produits et bien plus encore

L'application "Siemens Industry Online Support" est disponible pour Apple iOS et Android.

Code Data Matrix sur la plaque signalétique

Le code Data Matrix sur la plaque signalétique contient les données spécifiques de l'appareil. Ce code peut être lu avec n'importe quel smartphone et l'appli mobile "Industry Online Support" permet alors de visualiser les informations techniques de l'appareil correspondant.

1.7 Informations produit importantes

Utilisation de OpenSSL

Ce produit peut contenir les logiciels suivants :

- Logiciel développé par le projet OpenSSL pour une utilisation dans la boîte à outils OpenSSL
- Logiciel cryptographique créé par Eric Young
- Logiciel développé par Eric Young

Pour plus d'informations, voir sur Internet :

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

Respect du règlement général sur la protection des données

Siemens respecte les principes de la protection des données, en particulier les règles de limitation des données (protection de la vie privée dès la conception).

Pour ce produit, cela signifie que :

Le produit ne traite et n'enregistre aucune donnée à caractère personnel mais uniquement des données techniques fonctionnelles (p. ex. horodatage). Si l'utilisateur relie ces données à d'autres données (p. ex. plannings d'équipes) ou s'il enregistre des données à caractère personnel sur le même support (p. ex. disque dur) et crée par là même un lien avec des personnes, il est tenu de garantir lui-même le respect des dispositions légales en matière de protection des données.

Consignes de sécurité élémentaires 2

2.1 Consignes de sécurité générales

⚠ ATTENTION

Le non respect des consignes de sécurité et le manque de prise en compte des risques résiduels peuvent entraîner la mort

Le non respect des consignes de sécurité et des remarques relatives aux risques résiduels dans la documentation du matériel peut conduire à des accidents susceptibles d'entraîner la mort ou de causer des blessures graves.

- Respecter les consignes de sécurité figurant dans la documentation du matériel.
- Tenir compte des risques résiduels pour l'évaluation des risques.

⚠ ATTENTION

Danger de mort lié à des dysfonctionnements de la machine suite à un paramétrage incorrect ou modifié

Un paramétrage incorrect ou modifié peut entraîner des dysfonctionnements sur les machines, susceptibles de provoquer des blessures, voire la mort.

- Protéger le paramétrage contre l'accès non autorisé.
- Prendre les mesures appropriées pour palier aux défauts éventuels (p. ex. un arrêt ou une coupure d'urgence).

2.2 Garantie et responsabilité pour les exemples d'application

Les exemples d'application sont sans engagement et n'ont aucune prétention d'exhaustivité concernant la configuration, les équipements et les éventualités de toutes sortes. Les exemples d'application ne constituent pas des solutions client spécifiques, mais ont uniquement pour objet d'apporter une aide dans la résolution de problèmes typiques.

L'utilisateur est seul responsable de la mise en œuvre des produits selon les règles de l'art. Les exemples d'application ne vous dispensent pas des obligations de précaution lors de l'utilisation, de l'installation, de l'exploitation et de la maintenance.

2.3 Note relative à la sécurité

Siemens commercialise des produits et solutions comprenant des fonctions de sécurité industrielle qui contribuent à une exploitation sûre des installations, systèmes, machines et réseaux.

Pour garantir la sécurité des installations, systèmes, machines et réseaux contre les cybermenaces, il est nécessaire de mettre en œuvre - et de maintenir en permanence - un concept de sécurité industrielle global et de pointe. Les produits et solutions de Siemens constituent une partie de ce concept.

Il incombe aux clients d'empêcher tout accès non autorisé à ses installations, systèmes, machines et réseaux. Ces systèmes, machines et composants doivent uniquement être connectés au réseau d'entreprise ou à Internet si et dans la mesure où cela est nécessaire et seulement si des mesures de protection adéquates (ex: pare-feu et/ou segmentation du réseau) ont été prises.

Pour plus d'informations sur les mesures de protection pouvant être mises en œuvre dans le domaine de la sécurité industrielle, rendez-vous sur https://www.siemens.com/industrialsecurity (https://www.siemens.com/industrialsecurity).

Les produits et solutions Siemens font l'objet de développements continus pour être encore plus sûrs. Siemens recommande vivement d'effectuer des mises à jour dès que celles-ci sont disponibles et d'utiliser la dernière version des produits. L'utilisation de versions qui ne sont plus prises en charge et la non-application des dernières mises à jour peut augmenter le risque de cybermenaces pour nos clients.

Pour être informé des mises à jour produit, abonnez-vous au flux RSS Siemens Industrial Security à l'adresse suivante:
https://www.siemens.com/industrialsecurity (https://new.siemens.com/global/en/products/services/cert.html#Subscriptions)

Plus d'informations, voir sur Internet :

Manuel de configuration Industrial Security (https://support.industry.siemens.com/cs/ww/fr/view/108862708/en)

⚠ ATTENTION

États de fonctionnement non sûrs suite à une manipulation du logiciel

Les manipulations des logiciels, p. ex. les virus, chevaux de Troie ou vers, peuvent provoquer des états de fonctionnement non sûrs de l'installation, susceptibles de causer la mort, des blessures graves et des dommages matériels.

- Les logiciels doivent être maintenus à jour.
- Intégrer les composants d'entraînement et d'automatisation dans un concept global de sûreté industrielle (Industrial Security) de l'installation ou de la machine selon l'état actuel de la technique.
- Tenir compte de tous les produits utilisés dans le système global de sûreté industrielle (Industrial Security).
- Il convient de protéger les données stockées sur les supports de mémoire amovibles contre les logiciels nuisibles avec les mesures de protection appropriées, par exemple avec un antivirus.
- Contrôler à l'issue de la mise en service toutes les fonctions relatives à la sécurité.

Fonctionnalités 3

3.1 Fonctionnalités de "Run MyScreens"

Vue d'ensemble

"Run MyScreens" permet de créer des interfaces utilisateur représentant les extensions fonctionnelles spécifiques au constructeur de machines ou à l'utilisateur, ou encore de réaliser une présentation personnalisée.

Les interfaces utilisateur personnalisées créées permettent de générer entre autres des appels de cycle. La réalisation de boîtes de dialogue peut se faire directement dans la commande.

"Run MyScreens" est réalisée à l'aide d'un interpréteur et des fichiers de configuration qui contiennent la description des interfaces utilisateur.

"Run MyScreens" est configurée à l'aide de fichiers ASCII : Ces fichiers de configuration comportent la description de l'interface utilisateur. La syntaxe nécessaire à la création des fichiers est décrite dans les chapitres suivants.

Version de base

La fonction "Run MyScreens" est destinée à permettre au constructeur de machines de configurer ses propres boîtes de dialogue. Jusqu'à 5 boîtes de dialogue peuvent être configurées dès la version de base, soit dans l'arborescence de menus pour la conduite, soit pour des boîtes de dialogue de cycle personnalisées.

Option logicielle

Pour accroître le nombre des boîtes de dialogue, vous nécessitez l'une des options de logiciel suivantes :

- 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)

Autres conditions à prendre en compte

Les conditions suivantes sont à respecter :

- Les changements de boîtes de dialogue ne sont possibles qu'au sein d'un groupe fonctionnel.
- Les variables utilisateur ne doivent pas avoir le même nom que les variables système ou AP (voir aussi Tables de paramètres Variables système /PGAsl/)
- Les boîtes de dialogue activées par l'AP forment un groupe fonctionnel distinct (analogue aux images de cycles de mesure).

Utilitaires

- Editeur compatible UTF-8 (p. ex. l'éditeur intégré à l'interface utilisateur ou Bloc-notes)
- Un logiciel graphique est requis pour la création de graphiques ou d'images.

Noms de fichiers et codage

Remarque

Tenez compte du fait que, lorsque HMI Operate est utilisé dans la NCU, tous les noms de fichiers sur la carte CF sont écrits en minuscules (com, png, txt). C'est une particularité de Linux.

Sur la PCU vous pouvez écrire les noms de fichiers indifféremment en majuscules et en minuscules. Toutefois, il est recommandé de n'utiliser que des minuscules dans ce cas également, p. ex. pour un éventuel transfert ultérieur vers Linux.

Remarque

Lors de l'enregistrement des fichiers de configuration et de langue, s'assurer que le codage réglé dans l'éditeur utilisé est bien UTF-8.

Chemins d'enregistrement

Lors de l'enregistrement des fichiers de configuration, d'aide, etc. tenir compte de la convention suivante :

[Répertoire système siemens]	
Linux :	/card/siemens/sinumerik/hmi
Windows 7 :	C:\ProgramData\Siemens\MotionControl\siemens
[Répertoire système oem]	
Linux :	/card/oem/sinumerik/hmi
Windows 7 :	C:\ProgramData\Siemens\MotionControl\oem
[Répertoire système user]	
Linux :	/card/user/sinumerik/hmi
Windows 7 :	C:\ProgramData\Siemens\MotionControl\user

[Répertoire système addon]		
	Linux :	/card/addon/sinumerik/hmi
	Windows 7 :	C:\ProgramData\Siemens\MotionControl\addon
Emplacements		
	Configuration "Run MyScreens" :	[Répertoire système oem]/proj
	Fichier de configuration :	[Répertoire système oem]/cfg
	Fichiers de langue :	[Répertoire système oem]/lng
	Fichiers graphiques :	[Répertoire système oem]/Ico/ico[résolution] Exemple : [Répertoire système oem]/Ico/ico640
	Aide en ligne :	[Répertoire système oem]/hlp/[langue] Exemple : [Répertoire système oem]/hlp/eng

Utilisation

Les fonctions suivantes sont disponibles :

- Affichage de boîtes de dialogue et mise à disposition de :
  - touches logicielles
  - Variables
  - texte et texte d'aide
  - graphiques et images d'aide
- Appel de boîtes de dialogue par :
  - actionnement de touches logicielles (d'accès)
  - sélection d'AP / de CN
- Modification dynamique des boîtes de dialogue :
  - modifier, supprimer les touches logicielles
  - définir et réaliser des tableaux de variables
  - afficher, remplacer, supprimer les textes d'affichage (localisés ou non)
  - afficher, remplacer ou supprimer des graphiques
- Déclenchement d'actions en :
  - affichant les boîtes de dialogue
  - saisissant des valeurs (variables)
  - actionnant des touches logicielles
  - quittant les boîtes de dialogue
  - Changement de valeur d'une variable système ou utilisateur
- Échange de données entre boîtes de dialogue

- Variables :
  - lecture (variables CN, AP, utilisateur)
  - écriture (variables CN, AP, utilisateur)
  - combinaison par opérateurs mathématiques, de comparaison ou logiques
- Exécution de fonctions :
  - sous-programmes
  - fonctions de fichier
  - services PI
- Prise en compte des niveaux de protection en fonction des groupes d'utilisateurs

Mise en route

4

4.1 Introduction

Sur la base de l'exemple ci-après, vous apprendrez à connaître toutes les opérations nécessaires pour ajouter vos propres boîtes de dialogue à l'interface utilisateur SINUMERIK Operate à l'aide de Run MyScreens. Entre autres, vous apprendrez comment créer vos propres boîtes de dialogue, ajouter des images d'aide contextuelles et des appels de l'aide, définir des touches logicielles et comment vous pouvez naviguer entre les boîtes de dialogue.

4.2 Exemple de projet

4.2.1 Description du problème

Dans l'exemple de projet, vous créez les boîtes de dialogue décrites ci-après, y compris une aide contextuelle.

Boîte de dialogue 1

La première boîte de dialogue affichera les paramètres R (0 et 1) accessibles en écriture et les noms des axes géométriques (champs de saisie). Des images d'aide correspondantes seront intégrées pour les deux paramètres R. Une aide contextuelle sera intégrée pour les axes géométriques. La boîte de dialogue contient en outre des exemples de lignes de séparation (horizontales et verticales), champs bascule, champs de saisie/visualisation intégrant la sélection des unités et de barres de progression (avec ou sans changement de couleur).

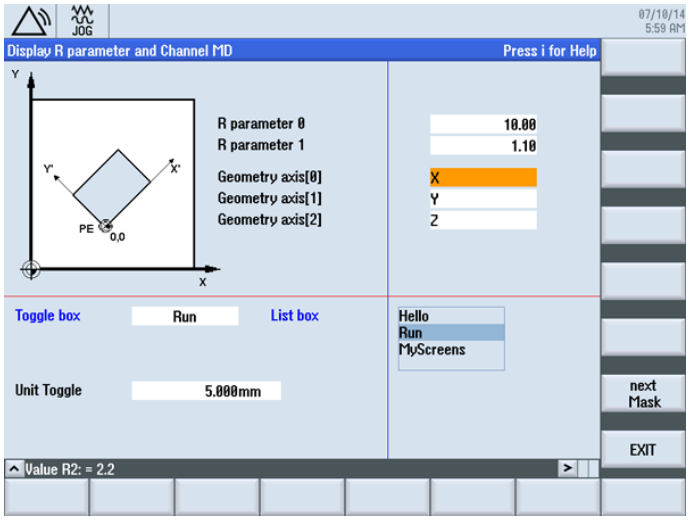


Figure 4-1 Paramètres R avec images d'aide

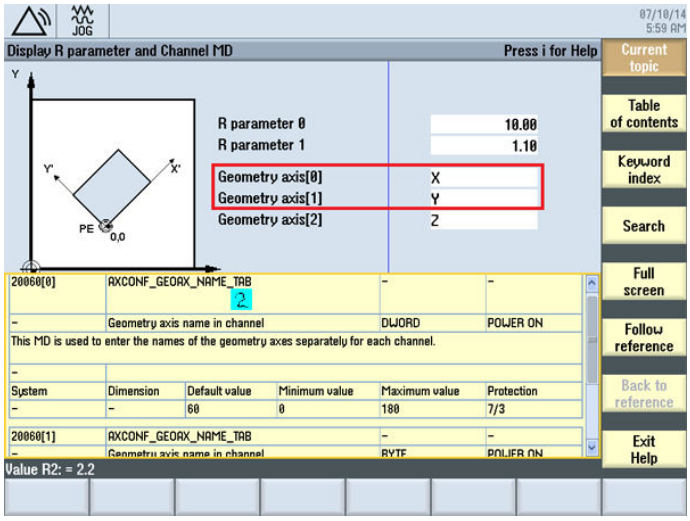


Figure 4-2 Axes géométriques avec aide contextuelle

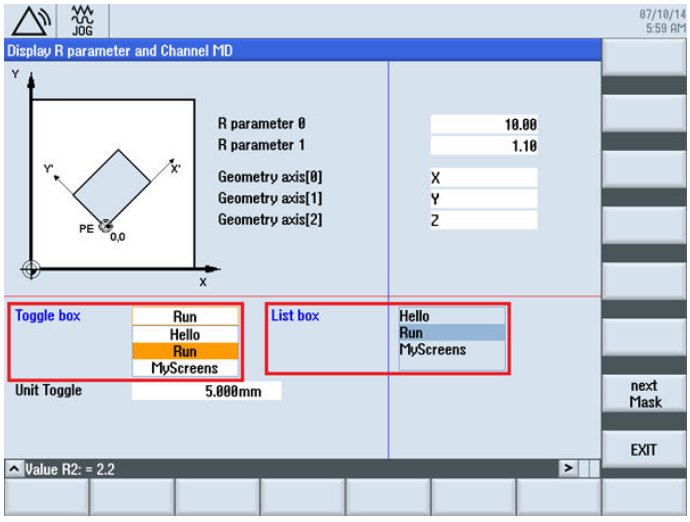


Figure 4-3 Champ bascule : liste et champ

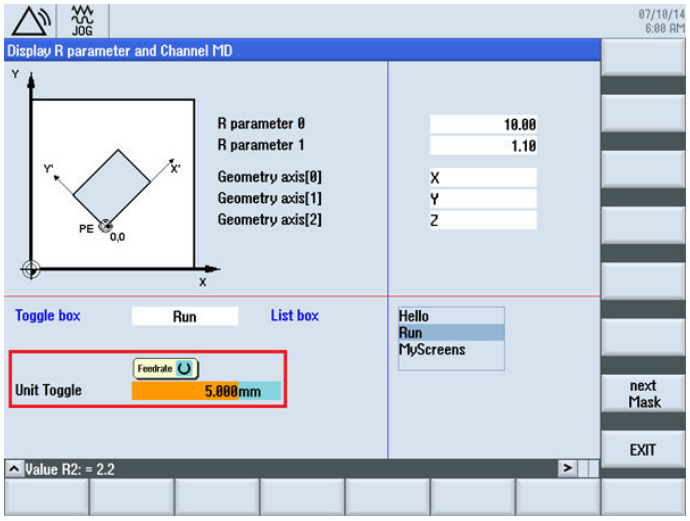


Figure 4-4 Champ de saisie/visualisation intégrant la sélection des unités

Boîte de dialogue 2

La deuxième boîte de dialogue affiche les valeurs SCM et SCP. La boîte de dialogue contient en outre des exemples de barres de progression avec et sans changement de couleur.

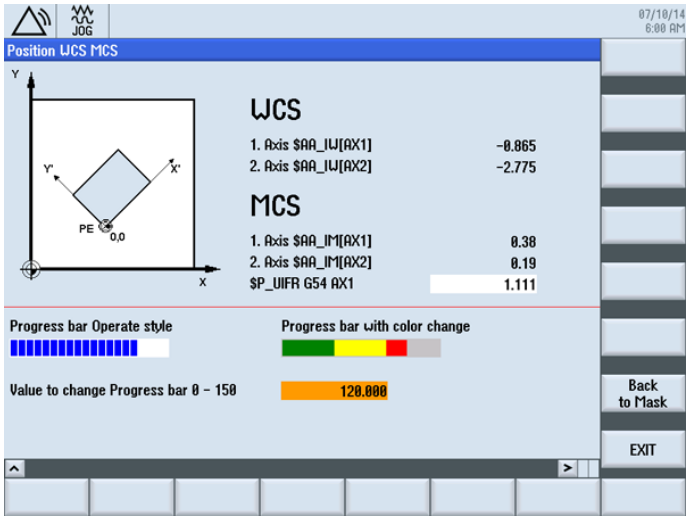


Figure 4-5 Barres de progression avec (à droite) ou sans (à gauche) changement de couleur

Navigation

L'appel de la première boîte de dialogue s'effectue à l'aide de la touche logicielle d'accès "START" dans le groupe fonctionnel Diagnostic. Utiliser pour ce faire la touche logicielle horizontale TL7.

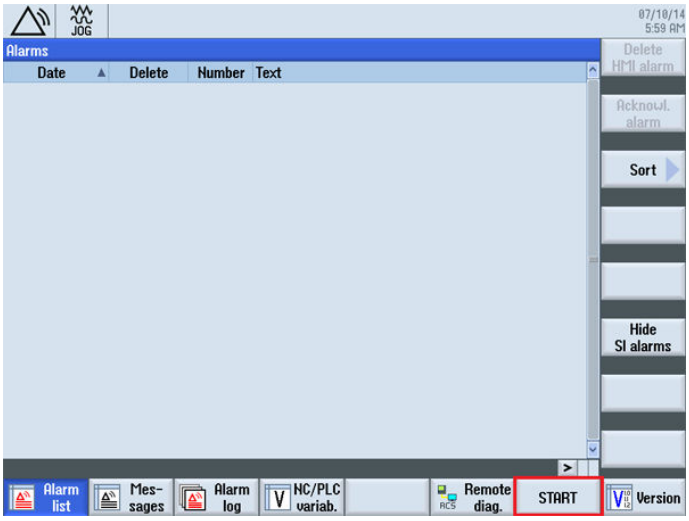


Figure 4-6 Touche logicielle d'accès „START" dans le groupe fonctionnel Machine, mode AUTO

A partir de la première boîte de dialogue, la seconde boîte de dialogue peut être appelée à l'aide de la touche logicielle "next Mask". La touche logicielle "EXIT" permet de revenir à la vue initiale du groupe fonctionnel (voir la figure ci-dessus).

La touche logicielle "EXIT" permet également de revenir à la vue initiale du groupe fonctionnel (voir la figure ci-dessus) à partir de la seconde boîte de dialogue. La touche logicielle "Back to Mask" permet de revenir à la première boîte de dialogue.

Marche à suivre

Les opérations nécessaires sont expliquées dans les chapitres suivants :

1. Création d'un fichier de configuration (fichier COM) (Page 25)
2. Enregistrement du fichier de configuration dans le répertoire OEM (Page 28)
3. Création de l'aide en ligne (Page 29)
4. Intégration de l'aide en ligne et enregistrement des fichiers dans le répertoire OEM (Page 32)
5. Copie de "easyscreen.ini" dans le répertoire OEM (Page 33)
6. Déclaration du fichier COM dans "easyscreen.ini" (Page 33)
7. Test du projet (Page 33)

4.2.2 Création du fichier de configuration (exemple)

Contenu du fichier de configuration

Dans un éditeur compatible UTF-8, créer le fichier de configuration "diag.com" pour les deux boîtes de dialogue.

```
; Identifiant de début de touche logicielle d'accès
//S(START)
   ; Touche logicielle d'accès, texte seul
   HS7=("START")
   ; Touche logicielle d'accès avec texte localisé et png
   ;HS7=([$80792,"\\sk_ok.png"])

   ; Méthode Press
   PRESS(HS7)
      ; Fonction LM ou LS
      LM("MASK1")
      ; LM avec indication d'un fichier com
      LM("MASK1","TEST.COM")
   END_PRESS
; Identifiant de fin de touche logicielle d'accès
//END

; Définition de la boîte de dialogue 1 avec titre et image
//M(MASK1/"Display R parameter and Channel MD"/"mz961_01.png")
   ; Définition des variables
   DEF VAR1 = (R2///,"R parameter 0"///"$R[0]"/200,50,150/400,50,100,)
   ; avec image d'aide
   DEF VAR2 = (R2///,"R parameter 1"//"mz961_02.png"/"$R[1]"/200,70,150/400,70,100)
   ; avec aide en ligne
   DEF ACHS_NAM1 = (S///"Press i for Help","Geometry axis[0]"////200,100,150/400,100,100//"sinume-
rik_md_1.html","20060[0]")
   ; avec aide en ligne
   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)
   ; Définition champs bascule et bascule d'unités
   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)
```

```
; Définition des touches logicielles dans la boîte de dialogue
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("next Mask")
VS8=("EXIT")

; Définition du bloc LOAD
LOAD
   ; Lecture de valeur avec RNP
   NOM_AXE1 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]")
   NOM_AXE2 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[1]")
   NOM_AXE3 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[2]")
   ; Affichage d'une ligne de la boîte de dialogue
   DLGL("Value R2: = " << RNP("$R[2]"))
   ; Ecriture d'une valeur avec WNP
   WNP("$R[3]",VAR0)

  ; Ligne de séparation verticale
  V_SEPARATOR(360,1,6,1)
  ; Ligne de séparation horizontale
  H_SEPARATOR(220,1,7,1)
END_LOAD

; Méthode Press
PRESS(VS7)
   ; Chargement d'une autre boîte de dialogue
   LM("MASK2")
; Identifiant de fin de la méthode Press
END_PRESS

; Méthode Press
PRESS(VS8)
```

```
      ; Sortie de la boîte de dialogue
      EXIT
   ; Identifiant de fin de la méthode Press
   END_PRESS
; Identifiant de fin de la boîte de dialogue 1
//END

; Définition de la boîte de dialogue 2 avec titre et image
//M(MASK2/"Position WCS MCS"/"mz961_01.png")
   ; Définition des variables
   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)

   ; Définition des barres de progression
   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)

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

  ; Définition des touches logicielles dans la boîte de dialogue
  HS1=("")
  HS2=("")
  HS3=("")
  HS4=("")
  HS5=("")
  HS6=("")
  HS7=("")
  HS8=("")
  VS1=("")
  VS2=("")
  VS3=("")
  VS4=("")
  VS5=("")
  VS6=("")
  VS7=("Back to Mask")
  VS8=("EXIT")
```

```
   : Méthode LOAD
   LOAD
      H_SEPARATOR(230,1,7,1)
   END_LOAD

   ; Méthode Change
   CHANGE(VAR5)
      ; Fonction PI_START
      PI_START("/NC,201,_N_SETUFR")
   ; Identifiant de fin de la méthode Change
   END_CHANGE

   ; Méthode Press
   PRESS(RECALL)
      ; Retour au masque appelant
      LM("MASK1")
   ; Identifiant de fin de la méthode Press
   END_PRESS

   ; Méthode Press
   PRESS(VS7)
      ; Retour au masque appelant
      LM("MASK1")
   ; Identifiant de fin de la méthode Press
   END_PRESS

   ; Méthode Press
   PRESS(VS8)
      ; Sortie de la boîte de dialogue pour retourner à l'application standard
      EXIT
   ; Identifiant de fin de la méthode Press
   END_PRESS
; Identifiant de fin de la boîte de dialogue
//END
```

4.2.3 Enregistrement du fichier de configuration dans le répertoire OEM (exemple)

Chemin d'enregistrement

Enregistrer le fichier de configuration "diag.com" dans le répertoire suivant :

[Répertoire système oem]/proj

4.2.4 Création de l'aide en ligne (exemple)

Créer le fichier HTML "sinumerik_md_1.html". L'exemple sous "Contenu de l'aide en ligne" illustre l'appel de l'aide contextuelle via **name="20060[0]"** et **name="20060[1]"**.

Remarques

Vous trouverez des instructions sur la création des fichiers HTML au chapitre Créer des fichiers HTML (Page 76).

Vous trouverez des instructions sur l'intégration de l'aide en ligne au chapitre Intégration de l'aide en ligne et enregistrement des fichiers dans le répertoire OEM (exemple) (Page 32).

Contenu de l'aide en ligne

L'exemple n'est pas commenté.

```
<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 Intégration de l'aide en ligne et enregistrement des fichiers dans le répertoire OEM (exemple)

Intégration de l'aide en ligne

La syntaxe d'intégration de l'aide en ligne est similaire à SINUMERIK Operate :

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

Remarque

Le nom du fichier HTML doit comporter exclusivement des caractères minuscules en raison du système LINUX !

Chemin d'enregistrement

Enregistrer le fichier HTML "sinumerik_md_1.html" pour l'aide en langue française dans le répertoire suivant :

[Répertoire système oem]/hlp/fra

Pour les autres langues, il sera éventuellement nécessaire de créer les dossiers (p. ex. chs, deu, eng, esp, ita...).

Une liste des codes de langue figure dans l'annexe "Listes de référence".

Informations complémentaires

Des informations détaillées sur la création de l'aide en ligne spécifique OEM figurent au chapitre Configuration de l'aide en ligne (Page 75).

4.2.6 Copie de "easyscreen.ini" dans le répertoire OEM (exemple)

Chemin d'enregistrement

Copier le fichier "easyscreen.ini" depuis le répertoire

[Répertoire système siemens]/cfg

dans le répertoire

[Répertoire système oem]/cfg.

4.2.7 Déclaration du fichier COM dans "easyscreen.ini" (exemple)

Adaptation dans le fichier "easyscreen.ini"

Effectuer les modifications suivantes dans le fichier "easyscreen.ini" dans le répertoire OEM. La déclaration du le fichier de configuration "diag.com" est ainsi terminée.

```
;#########################
;# 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 Test du projet (exemple)

Test de l'appel de la boîte de dialogue

Accéder au groupe fonctionnel Diagnostic. Cliquer sur la touche logicielle horizontale "START".

Les erreurs détectées par "Run MyScreens" lors de l'interprétation des fichiers de configuration sont consignées dans le fichier texte easyscreen_log.txt. De plus amples informations figurent au chapitre Traitement des erreurs (journal de bord) (Page 41).

Test des images d'aide contextuelles et de l'aide en ligne

Testez les images d'aide contextuelles et l'aide en ligne. Si des erreurs se produisent, vérifiez les chemins d'enregistrement.

Notions de base

5

5.1 Structure du fichier de configuration

Introduction

La description des nouvelles interfaces utilisateur est enregistrée dans des fichiers de configuration. Ces fichiers sont automatiquement interprétés et le résultat est affiché à l'écran. Les fichiers de configuration ne sont pas disponibles lors de la livraison et ils doivent être créés.

Un éditeur compatible UTF-8 (p. ex. l'éditeur intégré à l'interface utilisateur ou Bloc-notes) est utilisé pour créer les fichiers de configuration et de langue. La description peut également être expliquée par des commentaires. Un ";" est ajouté en signe de commentaire devant chaque explication.

Remarque

Lors de l'enregistrement des fichiers de configuration et de langue, s'assurer que le codage réglé dans l'éditeur utilisé est bien UTF-8.

Les mots-clés doivent cependant comporter uniquement des caractères contenus dans le jeu de caractères ASCII (y compris dans un fichier COM encodé en UTF-8). Dans le cas contraire, l'interprétation et, par conséquent, l'affichage correct des masques / boîtes de dialogue n'est pas garanti.

Configuration

Chaque groupe fonctionnel IHM dispose de touches logicielles d'accès fixes permettant de renvoyer aux boîtes de dialogue nouvellement créées.

Lors de l'appel "Charger masque" (LM) ou "Charger barre de touches logicielles" (LS) dans un fichier de configuration, il est possible de renommer le fichier dans lequel se trouve l'objet ouvert. Ainsi, la configuration peut être divisée en plusieurs catégories, par exemple toutes les fonctions d'un niveau de commande dans un fichier de configuration propre.

Structure du fichier de configuration

Un fichier de configuration se compose des éléments suivants :

1. description des touches logicielles d'accès
2. définition des boîtes de dialogue
3. définition des variables
4. Description des méthodes
5. définition des barres de touches logicielles

Remarque

Ordre

L'ordre indiqué dans le fichier de configuration doit absolument être respecté.

Exemple :

```
//S (START)                  ; Définition des touches logicielles d'accès (facultatif)
....
//END
//M (.....)                  : Définition de la boîte de dialogue
DEF .....                    ; Définition des variables
LOAD                         ; Description des blocs
...
END_LOAD
UNLOAD
...
END_UNLOAD
...
//END
//S (...)                    ; Définition d'une barre de touches logicielles
//END
```

Archivage des fichiers de configuration

Les fichiers de configuration sont enregistrés dans le répertoire [Répertoire système user]/proj ainsi que dans les répertoires [Répertoire système addon] et [Répertoire système oem] correspondants.

Conversion de textes à partir d'autres applications IHM

Procédure pour convertir un fichier texte avec un codage Codepage selon le codage texte UTF-8 :

1. Ouvrez le fichier texte sur un PG/PC dans un éditeur de texte.
2. Réglez le codage UTF-8 lors de l'enregistrement.

Le mécanisme de lecture via le codage Codepage est toujours pris en charge.

5.2 Structure de l'arborescence de commande

Principe de l'arborescence de commande

Plusieurs boîtes de dialogue liées entre elles forment une arborescence de commande. Un lien existe lorsqu'il est possible de passer d'une boîte de dialogue à l'autre. A l'aide des nouvelles touches logicielles horizontales ou verticales définies dans cette boîte de dialogue, il est possible de passer à la boîte de dialogue précédente ou à toute autre boîte de dialogue désirée.

Pour chaque touche logicielle d'accès, il est possible de créer une arborescence de commande :

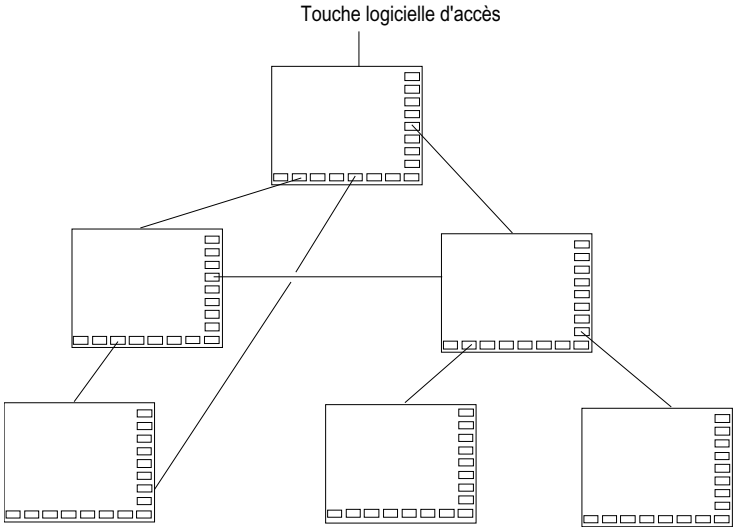


Figure 5-1 Arborescence de commande

Touches logicielles d'accès

Dans l'un des fichiers de configuration spécifiés dans "easyscreen.ini" sont définies une ou plusieurs touches logicielles (touches logicielles d'accès) servant de point de départ des séquences opératoires personnalisées.

La définition de la touche logicielle est liée à l'ouverture d'une boîte de dialogue personnalisée ou à une nouvelle barre de touches logicielles permettant d'effectuer d'autres actions.

En appuyant sur la touche logicielle d'accès, la boîte de dialogue affectée est chargée. Les touches logicielles correspondant à la boîte de dialogue sont alors activées. Les variables sont affichées sur les positions standard si aucune position spécifique n'a été configurée.

Retourner à l'application standard

Vous pouvez quitter la boîte de dialogue reconfigurée et revenir au groupe fonctionnel par défaut.

La touche <RECALL> vous permet de quitter les interfaces utilisateur reconfigurées dès lors que rien d'autre n'a été configuré pour cette touche.

Remarque

Appel de boîtes de dialogue dans le programme utilisateur de l'AP

Le choix d'une boîte de dialogue peut se faire à partir des touches logicielles mais également à partir de l'AP : un signal d'interface existe dans le DB19.DBB10 pour l'échange de signaux entre l'AP et l'IHM.

5.3 Définition et fonctions pour les touches logicielles d'accès

5.3.1 Définir la touche logicielle d'accès

Touche logicielle indépendante de la boîte de dialogue

Les touches logicielles d'accès sont des touches logicielles indépendantes de la boîte de dialogue qui ne sont pas appelées à partir d'une boîte de dialogue mais qui sont plutôt configurées **avant** la première boîte de dialogue. Afin d'accéder à la boîte de dialogue d'accueil ou à une barre de touches logicielles d'accès, la touche logicielle d'accès correspondante doit être définie.

Programmation

Le bloc de description d'une touche logicielle d'accès est le suivant :

```
//S(Start)                ; Identifiant de début de touche logicielle d'accès
HS1=(...)                 ; Définition de la touche logicielle d'accès : TL hori-
                          zontale 1
PRESS(HS1)                ; Méthode
  LM...                   ; Fonction LM ou LS
END_PRESS                 ; Fin de méthode
//END                     ; Identifiant de fin de touche logicielle d'accès
```

Positions autorisées pour touches logicielles d'accès

Dans les groupes fonctionnels, les positions autorisées pour les touches logicielles d'accès de "Run MyScreens" sont les suivantes :

Groupe fonctionnel	Position
Machine	TLH6
Paramètres	TLH7

Groupe fonctionnel	Position
Programme	TLH6 et TLH15 Cycles de mesure : TLH13 et TLH14
Gestionnaire de programmes	TLH2-8 et TLH12-16, si non occupés par des lecteurs
Diagnostic	TLH7
Mise en service	TLH7

Les touches logicielles d'accès sont configurées dans des fichiers spéciaux. Le nom de ces fichiers est déclaré dans le fichier "easyscreen.ini". Il correspond en principe au groupe fonctionnel concerné (p. ex. "startup.com" pour le groupe Mise en service). Le groupe fonctionnel Machine constitue une exception. Dans le groupe fonctionnel Machine se trouvent plusieurs fichiers spécifiques aux modes de fonctionnement ("ma_jog.com", "ma_auto.com").

La barre de touches logicielles comportant les touches logicielles d'accès se nomme "Start". L'utilisation de configurations existantes pour les touches logicielles d'accès demeure possible. La fonctionnalité de fusion ("Merge") entre les touches logicielles d'accès et les touches logicielles de l'application IHM correspondante (groupe fonctionnel) n'est pas prise en charge par le menu des touches logicielles d'accès.

Jusqu'au premier appel de boîte de dialogue, c'est-à-dire le moment à partir duquel la fonctionnalité complète est disponible (p. ex. exécution de blocs PRESS), un menu ou une barre de touches logicielles peuvent uniquement être remplacés dans leur totalité par un autre menu ou une autre barre.

Modèle de configuration des touches logicielles d'accès

Une description détaillée de toutes les positions admises pour les touches logicielles d'accès et leur configuration se trouve dans le fichier "easyscreen.ini" dans le répertoire suivant :

[Répertoire système siemens]/cfg

Ce fichier sert de modèle pour la configuration personnalisée des touches logicielles d'accès.

Voir aussi

Listes des touches logicielles d'accès

5.3.2 Fonctions pour touches logicielles d'accès

Fonctions pour touches logicielles indépendantes de la boîte de dialogue

Les touches logicielles d'accès permettent uniquement de déclencher certaines fonctions.

Les fonctions admises sont les suivantes :

- La **fonction LM** permet de charger une autre boîte de dialogue : **LM**("Descripteur"[,"Fichier"])
- La **fonction LS** permet d'afficher une autre barre de touches logicielles : **LS**("Descripteur"[, "Fichier"][, Merge])

- La **fonction "EXIT"** permet de quitter la nouvelle interface utilisateur réalisée et de revenir à l'application standard.
- La **fonction "EXITLS"** permet de quitter l'interface utilisateur courante et de charger une barre de touches logicielles définie.

Méthode PRESS

La touche logicielle est définie au sein du bloc de description et la fonction "LM" ou "LS" est attribuée dans la méthode PRESS.

Si la définition est signalée comme commentaire (point-virgule ; au début de la ligne) ou si le fichier de configuration est supprimé, la touche logicielle d'accès n'a pas de fonction.

```
//S(Start)                          ; Identifiant de début
HS6=("1er masque")                  ; Inscription du libellé "1er masque" sur la TL
                                    horizontale 6
PRESS(HS6)                          ; Méthode PRESS pour TL horizontale 6
  LM("Masque1")                     ; Chargement de la fonction Masque1, sachant que
                                    Masque1 doit être défini dans le même fichier.
END_PRESS                           ; Fin de la méthode PRESS
HS7=("2ème masque")                 ; Inscription du libellé "2ème masque" sur la TL
                                    horizontale 7
PRESS(HS7)                          ; Méthode PRESS pour TL horizontale 7
  LM("Masque2")                     ; Chargement de la fonction Masque2, sachant que
                                    Masque2 doit être défini dans le même fichier.
END_PRESS                           ; Fin de la méthode PRESS
//END                               ; Identifiant de fin du bloc d'accès
```

Exemple

```
HS1=("nouvelle barre de touches lo-
gicielles")
HS2=("aucune fonction")
PRESS(HS1)
  LS("Barre1")                      ; Chargement de la nouvelle barre de touches lo-
                                    gicielles
END_PRESS
PRESS (HS2)                         ; Méthode PRESS vide
END_PRESS
```

Différentes configurations des touches logicielles d'accès

Différentes configurations des touches logicielles d'accès sont regroupées. Ainsi, le nom du fichier à interpréter est d'abord lu à partir de "easyscreen.ini". La recherche de fichiers avec l'extension *.com s'effectue dans les répertoires suivants :

- [Répertoire système user]/proj
- [Répertoire système oem]/proj
- [Répertoire système addon]/proj
- [Répertoire système siemens]/proj

Les touches logicielles d'accès sont ensuite regroupées selon une même configuration, c'est-à-dire que les différentes touches logicielles sont comparées entre elles. S'il existe deux configurations ou plus pour une même touche logicielle, c'est toujours la configuration la plus récente qui est reprise dans la version Merge.

Les éventuelles barres de touches logicielles ou boîtes de dialogue sont ignorées. Si une touche logicielle contient une instruction sans indication de fichier, par exemple LM("test"), compte tenu que la barre de touches logicielles ou la boîte de dialogue souhaitée se trouve dans le même fichier, le nom de fichier correspondant est complété dans la version Merge interne, de manière à ce qu'aucune adaptation ne soit nécessaire. La configuration Merge obtenue s'affiche instantanément.

5.4 Traitement des erreurs (journal de bord)

Vue d'ensemble

Les erreurs détectées par "Run MyScreens" lors de l'interprétation des fichiers de configuration sont consignées dans le fichier texte "easyscreen_log.txt". Seules les erreurs des boîtes de dialogue actuellement sélectionnées sont renseignées. Les erreurs des boîtes de dialogue précédemment sélectionnées sont supprimées.

Le fichier contient les informations suivantes :

- L'action ayant provoqué une erreur.
- Le numéro de ligne et de colonne du premier caractère erroné.
- La totalité de la ligne erronée du fichier de configuration.
- Les entrées effectuées par la fonction DEBUG.

Remarque

Un horodatage actuel est mis en préfixe entre crochets devant chaque entrée. Cela peut, par exemple, être utile pour analyser des configurations à temps critique.

Enregistrement du fichier "easyscreen-log.txt"

Le fichier "easyscreen_log.txt" est enregistré dans le répertoire suivant :

[Répertoire système user]/log

Syntaxe

L'interprétation de la syntaxe débute lorsque la touche logicielle d'accès est définie et qu'une boîte de dialogue est configurée avec les identifiants de début et de fin ainsi qu'une ligne de définition.

```
//S(Start)
HS6=("1er masque")
PRESS(HS6)
  LM("Maske1")
END_PRESS
//END

//M(Maske1)
  DEF Var1=(R)
  DEF VAR2 = (R)
  LOAD
  VAR1 = VAR2 + 1             ; Un message d'erreur est consigné dans le journal de
                              bord, car VAR2 n'a pas de valeur
...
//END

; la syntaxe correcte serait
p. ex. :
//M(Maske1)
   DEF Var1=(R)
   DEF VAR2 = (R)

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

5.5 Remarques concernant "easyscreen.ini"

À partir de SINUMERIK Operate V4.7, le fichier "easycreen.ini" est étendu avec les entrées décrites dans ce chapitre. Le fichier "easyscreen.ini" se trouve dans le répertoire [Répertoire système siemens]/cfg.

Position initiale de l'image d'aide

Entrée dans "easyscreen.ini" :

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

Remarque :

La position initiale des images d'aide est définie sur la position de pixel configurée (Standard = true) indépendamment de la résolution.

Comportement d'étirement, hauteur de ligne et interligne par défaut

Entrée dans easyscreen.ini :

```
[GENERAL]
SymmetricalAspectRatio=false
DefaultLineHeight=18
DefaultLineSpacing=3
```

Remarques :

- L'entrée "SymmetricalAspectRatio" définit si les mêmes facteurs d'étirement dans le sens X ou Y doivent être utilisés lors de l'adaptation d'une configuration à une certaine résolution d'écran.
  - "false" (Standard) : avec des résolutions pour écran large, les champs et graphiques sont comprimés dans le sens Y (étirements non symétriques par rapport à 640x480). Par exemple, un carré configuré en 640x480 sera comprimé verticalement sur un panneau à écran large et deviendra un rectangle.
  - "true" : l'étirement est réalisé avec le même facteur dans les sens X et Y, ce qui permet aux champs et graphiques de conserver leurs proportions d'origine par rapport à la résolution 640x480. Par exemple, un carré configuré en 640x480 sera toujours présenté comme un carré sur un panneau à écran large.
- Les entrées "DefaultLineHeight" et "DefaultLineSpacing" permettent de définir la hauteur de ligne par défaut (Standard : 18 pixels) et l'interligne par défaut (par défaut : 3 pixels) par rapport à une résolution de 640x480. Ces valeurs ne prendront effet que si aucune position Y ou hauteur n'est indiquée dans la configuration de la position du texte abrégé ou du champ de saisie/visualisation.

Cycles dans l'écran de base Machine

- Accès au mode de fonctionnement JOG par HS6 avec possibilité de réaliser un appel de cycle dans l'écran de base Machine avec sélection [JOBSHOPINTEGRATION] :
  ```
  [JOBSHOPINTEGRATION]
  Integration = true
  ```
  ou par le bloc start dans la configuration
  ```
  LM("Mask1",,1)
  PRESS(VS8)
     GC("MOVE_RIDE") ; Appel de cycle créé
  EXIT:
  END_PRESS
  ```
- Sauvegarde des menus OPERATE avec la section [Integration] :
  Si la boîte de dialogue se compose uniquement de touches logicielles verticales, la barre de menus horizontale standard reste affichée avec les touches logicielles d'accès lors de l'ouverture de la boîte de dialogue. Si la boîte de dialogue se compose de touches logicielles horizontales et verticales, la barre de menus horizontale de la boîte de dialogue remplace la barre de menus horizontale avec les touches logicielles d'entrée.
  ```
  Intégration
  OperateMenusEnabled = true
  ```

Position du masque en fonction de la résolution : les FormPanels

Entrée dans "easyscreen.ini" :

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

Remarques :

La position du masque peut être définie de la manière suivante :

- La position du masque peut être indiquée en "pixels rapportés à 640x480".
  Exemple :
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**`0,219,335,174`**`)`
- La position du masque peut être couplée à la position dépendante de la résolution d'un FormPanel à partir d'une configuration de présentation d'écran Operate par défaut, p. ex. "FormPanel4" ("Fonctions auxiliaires") à partir de la configuration de présentation d'écran "slmastandardscreenlayout.SlMaStandardScreenLayout" de la zone "Machine"
  `//M(MyMask/"MyCaption"/"\`
  `\myhelp.png"/`**`"slmastandardscreenlayout.SlMaStandardScreenLayout.FormPanel4"`**
  `)`

  L'affichage alternatif de formes standard d'Operate ou le placement précis de masques Easyscreen à leur position se font ainsi en toute simplicité.
  Exemple :
  `//M(Mask/"Mask"//`**`"slstandardscreenlayout.SlStandardScreenLayout.LowerForm"`**`)`
  Il est toutefois possible d'utiliser une autre configuration d'écran au choix.
- La position du masque peut être couplée à une définition de FormPanels personnalisée et dépendante de la résolution dans le fichier "easyscreen.ini".
  Exemple :
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**`"MyPanel"`**`)`

5.6 Remarques pour utilisateurs passant à "Run MyScreens"

Remarque

Tenez compte du fait que, lorsque HMI Operate est utilisé dans la NCU, tous les noms de fichier sur la carte CF sont enregistrés en minuscules (com, png, txt).

Remarque

Lors de l'enregistrement des fichiers de configuration et de langue, s'assurer que le codage réglé dans l'éditeur utilisé est bien UTF-8.

Fichiers graphiques

Enregistrer systématiquement les fichiers d'image au format PNG "*.png".

Par exemple, pour les adaptations OEM, les fichiers doivent être enregistrés sous

[Répertoire système oem]/ico/[résolution]

Plus d'informations, voir chapitre Utiliser des images ou des graphiques (Page 66).

Adaptation du fichier de configuration

Vérifier les points suivants dans les fichiers de configuration :

- Comparer les touches logicielles d'accès avec les touches logicielles actuellement autorisées et les adapter, le cas échéant.
- Renommer les fichiers graphiques intégrés selon la section "Fichiers graphiques" ci-dessus.

Par exemple, pour les adaptations OEM, les données sont enregistrées dans le répertoire suivant :

[Répertoire système oem]/proj

[Répertoire système user]/proj A

[Répertoire système addon]/proj

Adaptation des fichiers d'aide

Tous les fichiers d'aide doivent être enregistrés au format UTF-8. Vérifiez vos fichiers existants et réenregistrez-les avec un éditeur approprié.

Les fichiers HTML sont enregistrés dans le répertoire suivant, par exemple pour le français :

[Répertoire système oem]/hlp/fra

[Répertoire système user]/hlp/fra

[Répertoire système addon]/hlp/fra

Les répertoires pour les autres langues doivent être créés conformément aux codes de langue correspondants.

Vérification de la licence "Run MyScreens"

Vérifier si le nombre de boîtes de dialogue insérées dépasse le nombre maximum de 5 boîtes de dialogue compris dans la version de base.

Pour accroître le nombre de boîtes de dialogue, l'une des options de logiciel suivantes est nécessaire :

- 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 Syntaxe de configuration étendue

A partir de SINUMERIK Operate V4.7, vous disposez d'une syntaxe simplifiée pour définir des masques, variables, touches logicielles et colonnes de tableau. Cette autre syntaxe améliore la lisibilité et la maintenabilité. Les propriétés et attributs peuvent être indiqués dans un ordre quelconque, les entrées vides sont ignorées. Une différence par rapport à la syntaxe utilisée jusqu'à présent est que la liste des propriétés et des attributs est maintenant entourée d'accolades "**{**" et "**}**" au lieu de parenthèses "**(**" et "**)**" .

Les propriétés et attributs sont indiqués comme suit :
`{<Nom> = <Valeur>, <Nom> = <Valeur>, …}`

La syntaxe précédente reste compatible.

Syntaxe étendue pour la définition de masques

//M {<nom du masque> [,**HD**=<titre>] [,**HLP**=<graphique>] [,**X**=<position X>] [,**Y**=<position Y>] [,**W**=<largeur>] [,**H**=<hauteur>] [,**VAR**=<>variable système ou utilisateur] [,**HLP_X**=<position X de l'image d'aide>] [,**HLP_Y**=<position Y de l'image d'aide>] [,**CM**=<orientation de colonne>] [,**CB**=comportement à l'ouverture de la boîte de dialogue] [,**XG**=<interpréter l'image d'aide comme un graphique X3d>] [,**PANEL**=<nom du FormPanel lié>][,**MC**=<couleur d'arrière-plan du masque>][,**HD_AL**=<orientation de l'en-tête du masque>] [,**LANGFILELIST**=<liste des fichiers de langue spécifiques au masque>]}

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

Syntaxe étendue pour la définition de variables

DEF <nom de variable> = {[**TYP**=<type>] [,**MIN**=<valeur minimale>] [,**MAX**=<valeur maximale>] [,**TGL**=<valeurs de basculement>] [,**VAL**=<préréglage>] [,**LT**=<texte long>] [,**ST**=<texte court>] [,**GT**=<texte du graphique>] [,**UT**=<texte d'unité>] [,**TT**=<texte d'info-bulle>] [,**TG**=<option de basculement>] [,**WR**=<mode de saisie>] [,**AC**=<niveau d'accès>] [,**AL**=<orientation du texte>] [,**FS**=<taille de police>] [,**LI**=<traitement des valeurs limites>] [,**UR**=<vitesse de rafraîchissement>] [,**CB**=<comportement à l'ouverture de la boîte de dialogue>] [,**HLP**=<image d'aide>] [,**VAR**=<variable système ou utilisateur>] >] [,**TXT_X**=<position X du texte court>] [,**TXT_Y**=<position Y du texte court>] [,**TXT_W**=<largeur du texte court>] [,**TXT_H**=<hauteur du texte court>] [,**X**=<position X du champ de saisie et de visualisation>] [,**Y**=<position Y du champ de saisie et de visualisation>] [,**W**=<largeur du champ de saisie et de visualisation>] [,**H**=<hauteur du champ de saisie et de visualisation>] [,**UT_DX**=<espacement entre champ d'entrée/sortie et champ d'unité>] [,**UT_W**=<largeur du champ d'unité>] [,**BC**=<couleur d'arrière-plan du champ de saisie et de visualisation>] [,**FC**=<couleur de premier plan du champ de saisie et de visualisation>] [,**BC_ST**=<couleur d'arrière-plan du texte court>] [,**FC_ST**=<couleur de premier plan du texte court>] [,**BC_GT**=<couleur d'arrière-plan du texte de graphique>] [,**FC_GT**=<couleur de premier plan du texte de graphique>] [,**BC_UT**=<couleur d'arrière-plan du texte d'unité>] [,**FC_UT**=<couleur de premier plan du texte d'unité>] [,**SC1**=<couleur de signal 1 pour la barre de progression>] [,**SC2**=<couleur de signal 2 pour la barre de progression>] [,**SVAL1**=<valeur de seuil 1 pour la barre de progression>] [,**SVAL2**=<valeur de seuil 2 pour la barre de progression>] [,**DT**=<type d'affichage>] [,**DO**=<orientation de l'affichage>] [,**OHLP**=<aide en ligne>][,**LINK_TGL**=<nom de la variable de basculement liée>]}

Exemples :

```
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=""Off"", 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"}
```

Syntaxe étendue pour la définition de touches logicielles

SK = {[**ST**=<libellé>] [,**AC**=<niveau d'accès>] [,**SE**=<état>]}

Exemples :

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

Syntaxe étendue pour la définition de colonnes de tableaux

{[**TYP**=<type>] [,**MIN**=<valeur minimale>] [,**MAX**=<valeur maximale>] [,**LT**=<texte long>] [,**ST**=<texte court>] [,**WR**=<mode de saisie>] [,**AC**=<niveau d'accès>] [,**AL**=<orientation du texte>] [,**FS**=<taille de police>] [,**LI**=<traitement des valeurs limites>] [,**UR**=<vitesse de rafraîchissement>] [,**HLP**=<image d'aide>] [,**VAR**=<variable système et utilisateur>] >] [,**W**=<largeur de colonne>] [,**OF1**=<Offset1>] [,**OF2**=<Offset2>] [,**OF3**=<Offset3>]}

Exemple :

```
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 et commande MutliTouch

Pour l'adaptation spécifique à SmartOperation et la commande MultiTouch, vous avez les possibilités suivantes :

- Adaptation automatique de la hauteur et de la largeur de champ par rapport à la taille utilisable minimale des champs et l'interligne (attribut de masque MA).
- Etirement optimisé des champs au pixel près pour les résolutions supérieures (attribut de masque PA).
- Définition de la hauteur de champ et de l'interligne de manière proportionnelle à la police de caractères (attribut de masque FA).
- Défilement libre des champs pour que le clavier virtuel ne masque pas la zone de saisie (attribut de masque KM).

Vous trouverez de plus amples informations à ce sujet à la section "Programmation" du chapitre Définition des propriétés de la boîte de dialogue (Page 54).

Boîtes de dialogue

6

6.1 Structure et éléments d'un dialogue

6.1.1 Définir un dialogue

Définition

Une boîte de dialogue est une partie de l'interface utilisateur qui se compose d'une ligne de titre, d'éléments du dialogue et/ou de graphique, d'une ligne d'affichage de messages ainsi que de 8 touches logicielles horizontales et 8 verticales.

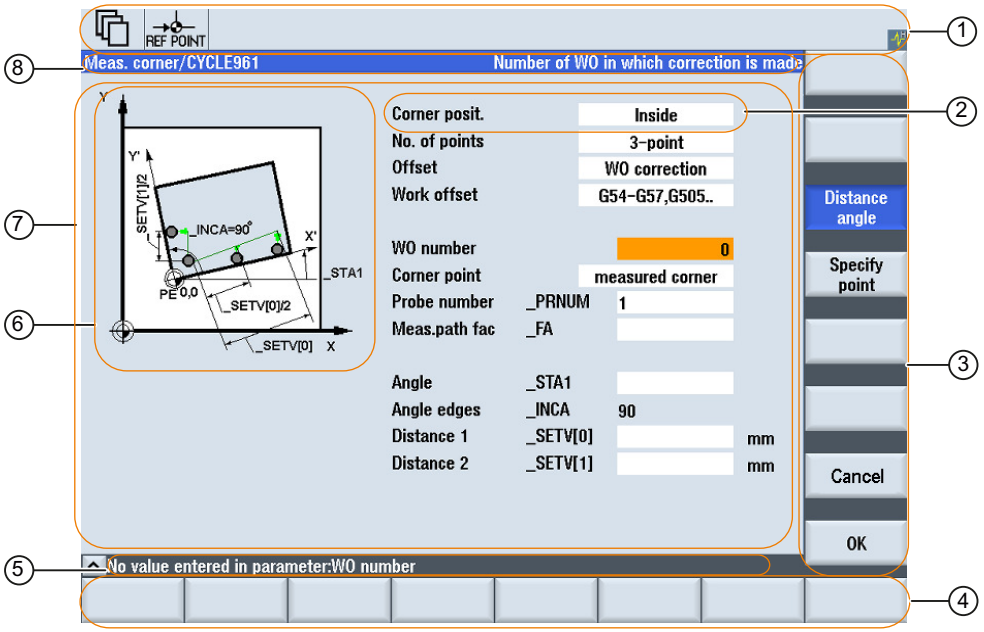
Les éléments de la boîte de dialogue sont les suivants :

- Variables
  - Valeurs limites / champ bascule
  - Réglage par défaut des variables
- Image d'aide
- Textes
- Attributs
- Variable système ou utilisateur
- Position texte court
- Position champ de saisie et de visualisation
- Couleurs

Propriétés d'une boîte de dialogue :

- Titre
- Graphique
- Dimension
- Variable système ou utilisateur

- Position Graphique
- Attributs



① Affichage de l'état de la machine (en-tête)
② Élément de boîte de dialogue
③ 8 touches logicielles verticales
④ 8 touches logicielles horizontales
⑤ Affichage de messages de diagnostic
⑥ Graphique
⑦ Boîte de dialogue
⑧ Ligne de titre de la boîte de dialogue avec titre et texte long

Figure 6-1 Structure de la boîte de dialogue

Vue d'ensemble

Généralement, la description d'une boîte de dialogue (bloc de description) est structurée de la façon suivante :

Bloc de description	Commentaire	Renvoi au chapitre
`//M...`	;Identifiant de début de la boîte de dialogue	
`DEF Var1=...` `...`	;Variables	Variables (Page 85)
`HS1=(...)` `...`	;Touches logicielles	Définition de la barre de touches logicielles (Page 66)

Bloc de description	Commentaire	Renvoi au chapitre
PRESS(HS1) LM... END_PRESS	;Identifiant de début de la méthode ;Actions ;Identifiant de fin de la méthode	Méthodes (Page 121)
//END	;Identifiant de fin de la boîte de dialogue	

Le bloc de description de la boîte de dialogue contient d'abord la définition de différentes variables signalées comme élément de dialogue, ainsi que celle des touches logicielles horizontales et verticales. Puis, différentes actions sont configurées dans les méthodes.

Remarque

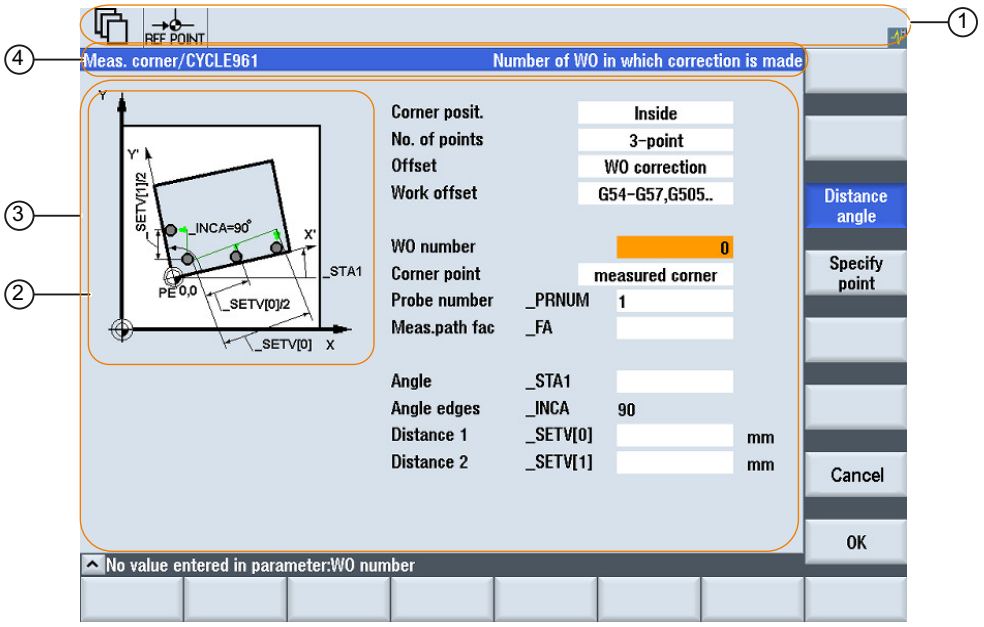
Dans le groupe fonctionnel "Machine", il faut respecter l'ordre suivant lors du changement de boîte de dialogue :

À la condition que les dimensions du masque suivant soient inférieures à celles du masque précédent ou que la position du masque suivant soit différente de celle du masque précédent, le changement de boîte de dialogue ne fonctionne que lorsque le masque suivant a été quitté avant de revenir au premier masque et que le premier masque a été rechargé.

6.1.2 Définition des propriétés de la boîte de dialogue

Description

L'identifiant de début de la boîte de dialogue définit en même temps les propriétés de la boîte de dialogue.



① Affichage de l'état de la machine (en-tête)
② Graphique
③ Boîte de dialogue
④ Ligne de titre de la boîte de dialogue avec titre et texte long

Figure 6-2 Propriétés d'une boîte de dialogue

Programmation

Syntaxe :	**//M**(Descripteur/[Titre]/[Graphique]/[Dimension]/[Variable système ou utilisateur]/[Position graphique]/[Attributs]/Liste des fichiers de langue spécifiques au masque) Voir aussi le chapitre Syntaxe de configuration étendue (Page 47).
Description :	Définir une boîte de dialogue

Paramètres :	Descripteur			Nom de la boîte de dialogue
	Titre			Titre de la boîte de dialogue sous forme de texte ou de lien vers un texte (p. ex. $85011) d'un fichier texte localisé
	Graphique			Fichier graphique avec chemin entre guillemets
	Dimension			Position et dimensions de la boîte de dialogue en pixels (marge à gauche, marge en haut, largeur, hauteur), par rapport au coin supérieur gauche de l'écran. Les données sont séparées par une virgule.
	Variable système ou utilisateur			Variable système ou utilisateur affectée à la position actuelle du curseur. La position du curseur peut être indiquée à la CN ou à l'AP à l'aide de variables système ou utilisateur. La première variable possède l'index 1. L'ordre correspond à l'ordre de la configuration
	Position Graphique			Position du graphique en pixels (marge à gauche, marge en haut), par rapport au coin supérieur gauche de la boîte de dialogue. Les données sont séparées par une virgule.
	Attributs			Les indications des attributs sont séparées par des virgules. Les attributs possibles sont :
		CM		Column Mode : disposition des colonnes
			CM0	Préréglage : la répartition des colonnes est effectuée séparément pour chaque ligne.
			CM1	La répartition des colonnes de la ligne comportant le plus grand nombre de colonnes est appliquée à l'ensemble des lignes.
		CB		Bloc CHANGE : comportement à l'ouverture de la boîte de dialogue : les attributs cb, indiqués pour une définition de variables, ont la priorité pour la variable par rapport aux données globales de définition de la boîte de dialogue. Voir aussi AUTOHOTSPOT.
			CB0	Préréglage : Tous les blocs CHANGE de la boîte de dialogue sont exécutés lors de l'ouverture.
			CB1	Les blocs CHANGE ne sont exécutés que lorsque la valeur correspondante change.
		XG		Intégration d'une animation X3D comme image d'aide (uniquement dans la programmation par étapes opérationnelles) Exemple : `//M(Meas/ $85605/"myx3dhelpfile.hmi,,Z_Animation,,G17"///30,10/ XG1)`
			XG0	Préréglage = 0
			XG1	L'attribut de masque XG doit déjà être réglé sur 1 dans la définition de masque, il n'est plus possible de procéder à une modification en cours d'exécution. Remarque : L'indication dans la propriété de masque HLP doit aussi être définie en conséquence.
		AL		L'attribut de masque AL permet d'influer sur l'orientation du titre du masque. Exemple : Affichage centré du titre de fenêtre "Set password" : `//M(MY_PWD_SET/”Set password”//"EasyPwdModalLayout"// AL2)`
			AL0	justifié à gauche, préréglage

	AL1	justifié à droite
	AL2	centré

KM		Le défilement libre dans le masque (SlGfwScrollArea), lorsque le clavier virtuel doit être affiché sur un pupitre tactile multipoint, peut être influencé comme suit :
		• Dans la ligne de définition du masque avec l'attribut de masque "KM" (KeyboardMode)
		Exemple :
		`//M(MyMTMask/"MultiTouch Mask"/////KM1)`
		• En tant que réglage global pour tous les masques Run MyScreens dans le fichier "easyscreen.ini"
		Exemple :
		`[GENERAL]`
		`DefaultVirtualKeyboardMode=1`
		
		Remarque :
		Si l'attribut de masque KM est défini explicitement dans la configuration de masque, ce réglage est prioritaire par rapport au réglage global dans le fichier "easyscreen.ini".
		
		(Voir aussi le chapitre SmartOperation et commande MutliTouch (Page 49))

	KM1	Défilement libre actif (par défaut)
	KM0	Défilement libre inactif
PG		Orientation et aspect du titre de fenêtre en liaison avec des images PNG (effet uniquement dans l'éditeur !) Exemple : `//M(MyPg1SampleMask/"My PG1 Sample Mask"/////PG1)`
	PG0	(Par défaut)
	PG1	Orientation et aspect du titre de fenêtre en liaison avec des images PNG Indication du chemin (tout à gauche), nom de masque (à droite) L'affichage du texte long n'est plus possible dans ce mode. Vous pouvez également travailler avec ToolTips.

MA		Adaptation automatique de la hauteur de champ pour la commande MultiTouch (MutliTouch Adjustment)
		Pour un tableau de commande MultiTouch, l'adaptation englobe le cas échéant l'agrandissement à la taille utilisable minimale (largeur, hauteur) des champs (exception : Progressbar, animated GIF, CustomWidget) ainsi que l'interligne.
		L'adaptation MultiTouch peut :
		• être effectuée globalement pour tous les masques Run MyScreens dans le fichier "easyscreen.ini", par exemple
		`[GENERAL]`
		`DefaultMultiTouchAdjustmentLevel=1`
		• ou spécifiquement pour un masque dans la ligne de définition du masque avec l'attribut de masque "MA" pour écraser le réglage global dans le fichier "easyscreen.ini", par exemple

		`//M(MyMTMask/"MultiTouch Mask"////MA1)` Si l'attribut de masque MA est défini explicitement dans la configuration de masque, ce réglage est prioritaire par rapport au réglage global dans le fichier "easyscreen.ini". (Voir aussi le chapitre SmartOperation et commande MutliTouch (Page 49))
	MA0	Aucune adaptation n'a lieu.
	MA1	Seuls les champs pour lesquels la marge en haut et/ou la hauteur/largeur de champ n'ont pas été configurées (champs utilisant les positions par défaut et laissant calculer la marge en haut et/ou la hauteur/largeur de champ par Run MyScreens) sont adaptés. (Par défaut)
	MA2	Tous les champs sont adaptés quelles que soient les tailles de champ configurées.

PA		Étirement optimisé des champs au pixel près pour les résolutions supérieures (Pixel Fine Adjustment).
		Jusqu'à présent, toutes les positions des champs sont calculées par rapport à 640x480, puis zoomées avec les facteurs d'étirement correspondants, horizontaux et verticaux. Cette procédure a toutefois l'inconvénient de pouvoir provoquer des erreurs d'arrondi pour les "facteurs d'étirement défavorables". La hauteur de champ ou l'interligne peuvent par exemple varier d'un pixel d'une ligne à l'autre. Pour y remédier, il existe le mode "Pixel Fine Adjustment" dans lequel les positions des champs sont directement déterminées dans la résolution actuelle.
		`//M(MyMask/"My Mask"////PA1)`
		Ce réglage peut également être effectué globalement pour tous les masques Run MyScreens dans le fichier "easyscreen.ini", par exemple
		`[GENERAL]`
		`DefaultPixelFineAdjustment=1`
		Si l'attribut de masque MA est défini explicitement dans la configuration de masque, ce réglage est prioritaire par rapport au réglage global dans le fichier "easyscreen.ini".
		Si le masque est représenté avec Font Adjustment = FA1, le mode Pixel Fine Adjustment est automatiquement actif pour ce masque.
		(Voir aussi le chapitre SmartOperation et commande MutliTouch (Page 49))

	PA0	Étirement effectué comme jusqu'à présent (positions calculées par rapport à 640x480, puis étirées). (En raison de la compatibilité : par défaut)
	PA1	Étirement optimisé des champs au pixel près pour les résolutions supérieures

FA		Définition de la hauteur de champ et de l'interligne de manière proportionnelle à la police de caractères (Font Adjustment).
		Ce réglage peut également être effectué globalement pour tous les masques Run MyScreens dans le fichier "easyscreen.ini", par exemple
		`[GENERAL]`
		`DefaultFontAdjustment=1`
		Si l'attribut de masque FA est défini explicitement dans la configuration de masque, ce réglage est prioritaire par rapport au réglage global dans le fichier "easyscreen.ini".
		Les réglages pour DefaultLineHeight et DefaultLineSpacing n'ont aucune signification lorsque le mode Font Adjustment est actif.
		(Voir aussi le chapitre SmartOperation et commande MutliTouch (Page 49))

	FA0	La hauteur de champ et l'interligne sont étirés comme jusqu'à présent (en raison de la compatibilité : par défaut)

		FA1	La hauteur de champ et l'interligne sont définis de manière proportionnelle à la police de caractères (réglage automatique de l'attribut de masque PA=1).
		FA2	Comme FA1, les coordonnées X et la largeur de champ étant en plus étirées de manière proportionnelle à la police de caractères. (Réglage automatique de l'attribut de masque PA=1) (Possible uniquement en liaison avec l'attribut XG=1 !)
	NT		Type de navigation (Navigation Type)
		NT0	NT0 = Mode par défaut La navigation est identique aux masques du groupe fonctionnel dans lequel le masque Run MyScreens est intégré. C.-à-d. que la navigation s'effectue en fonction des lignes dans les masques de cycle de l'éditeur (voir NT2), géométriquement dans tous les autres masques "normaux" (voir NT1).
		NT1	Navigation géométrique (haut, bas, gauche, droite) jusqu'à ce que les bords du masque soient atteints (par défaut dans un environnement Run MyScreens normal, p. ex. Area "Custom")
		NT2	En fonction des lignes, c.-à-d. à l'intérieur de la ligne vers la droite jusqu'à ce que la fin de la ligne soit atteinte (par défaut dans les masques de cycle de l'éditeur)
	NR		Sens de navigation jusqu'à ce que la touche Entrée soit enfoncée (Return Navigate Direction)
		NR0	NR0 = désactivé (default), c.-à-d. que la navigation s'effectue en ordre tabulaire à l'intérieur du masque lorsque la touche Entrée est enfoncée (par défaut dans un environnement Run MyScreens normal)
		NR1	Lorsque la touche Entrée est enfoncée, la navigation se comporte comme lorsque la touche fléchée vers le haut est enfoncée.
		NR2	comme NR1, cependant vers le bas
		NR3	comme NR1, cependant vers la gauche
		NR4	comme NR1, cependant vers la droite
	TA		Position des tooltips
		TA0	automatique (par défaut)
		TA1	gauche
		TA2	droite
		TA3	en haut
		TA4	en bas
		TA5	en haut à gauche
		TA6	en bas à gauche
		TA7	en haut à droite
		TA8	en bas à droite

MC			Couleur d'arrière-plan du masque (Mask Color)
			Exemple :
			Définir la couleur d'arrière-plan du masque en bleu (= 6)
			`//M(MyMask/"MyMask"//////6)`
			ou
			`PRESS(HS1)`
			`  MC=6`
			`END_PRESS`

	Liste des fichiers de langue spécifiques au masque	séparés par une virgule

Accès aux propriétés de la boîte de dialogue

Au sein des méthodes (p. ex. Bloc PRESS), il est possible d'accéder en lecture et en écriture aux propriétés suivantes de la boîte de dialogue :

- HD = Titre (en-tête)
- HLP = Image d'aide
- VAR = Variable système ou utilisateur
- MC = Couleur d'arrière-plan du masque
- CM = Disposition des colonnes (lecture seule)
- CB = Comportement à l'ouverture (lecture seule)
- XG = Intégration X3d (lecture seule)
- AL = Orientation du titre du masque (lecture seule)

Exemple

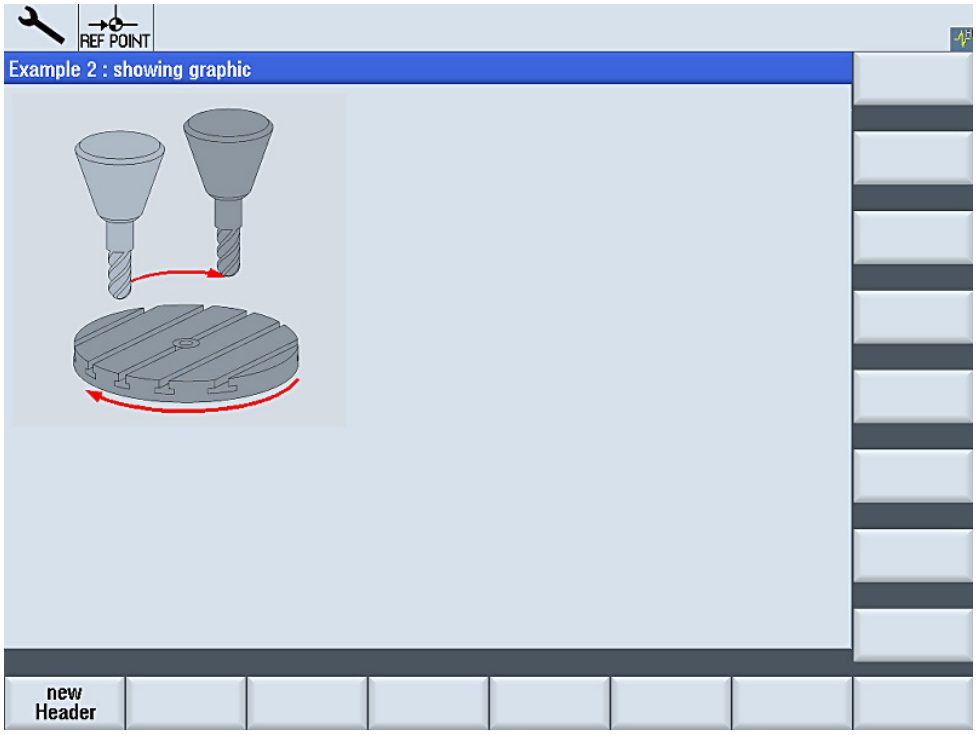


Figure 6-3 "Example 2: showing graphic"

```
//S(Start)
HS7=("Example", sel, 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
```

Voir aussi

Exemple de programmation pour le groupe "Custom" (Page 280)

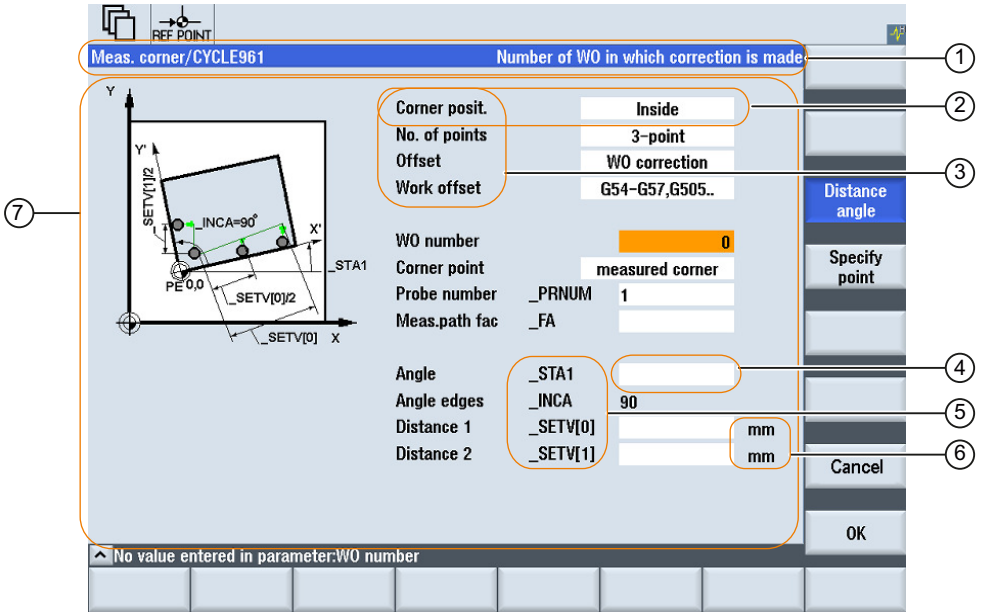
6.1.3 Définition des éléments de dialogue

Élément de boîte de dialogue

L'élément de boîte de dialogue est la partie visible d'une variable, c'est-à-dire le texte court, le texte graphique, les champs de saisie et de visualisation, le texte d'unité et l'info-bulle. Les éléments de boîte de dialogue sont affichés dans les lignes de la partie principale de la boîte de dialogue. Un ou plusieurs éléments de boîte de dialogue peuvent être définis par ligne.

Propriétés des variables

Toutes les variables sont valables uniquement pour la boîte de dialogue active. Ces caractéristiques sont affectées à l'aide de la définition d'une variable. Au sein des méthodes (p. ex. méthode PRESS), il est possible d'accéder aux valeurs des propriétés de la boîte de dialogue.



① Ligne de titre de la boîte de dialogue avec titre et texte long
② Élément de boîte de dialogue
③ Texte court
④ Champ de saisie et de visualisation
⑤ Texte du graphique
⑥ Texte des unités
⑦ Partie principale de la boîte de dialogue

Figure 6-4 Éléments d'une boîte de dialogue

Programmation - vue d'ensemble

Les paramètres individuels à séparer par des virgules sont placés entre parenthèses :

`DEF` [Descripteur] =	**Descripteur = Nom des variables**
	Type de variable
	/[Valeurs limites ou champ bascule]
	/[Valeur par défaut]
	/[Texte(texte long, texte court|image, texte graphique, texte d'unité)]
	/[Attributs]
	/[Image d'aide]
	/[Variable système ou utilisateur]
	/[Position texte court]

	/[Position champ de saisie et de visualisation(gauche, haut, largeur, hauteur)]
	/[Couleurs]
	/[Aide en ligne] (Page 75)

Voir aussi

Paramètres de variables (Page 93)

6.1.4 Définition de boîtes de dialogue à plusieurs colonnes

Vue d'ensemble

Dans une boîte de dialogue, plusieurs variables peuvent être représentées dans une ligne. Les variables sont toutes définies dans ce cas dans une seule ligne de définition du fichier de configuration.

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

Afin d'afficher plus clairement les différentes variables dans le fichier de configuration, les lignes de définition peuvent être coupées après chaque définition de variable et chaque virgule consécutive.

Le mot-clé "DEF" désigne toujours le début d'une nouvelle ligne :

```
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)
```

Remarque

Lors de la conception de boîtes de dialogue à plusieurs colonnes, il convient de tenir compte du fait qu'un très grand nombre de colonnes peut ralentir le système, le cas échéant.

6.1.5 Boîtes de dialogue concernant le mot de passe

Utilisation des boîtes de dialogue standard concernant le mot de passe

Les boîtes de dialogue prédéfinies suivantes concernant le mot de passe peuvent être intégrées dans la configuration de masques :

- Définition du mot de passe

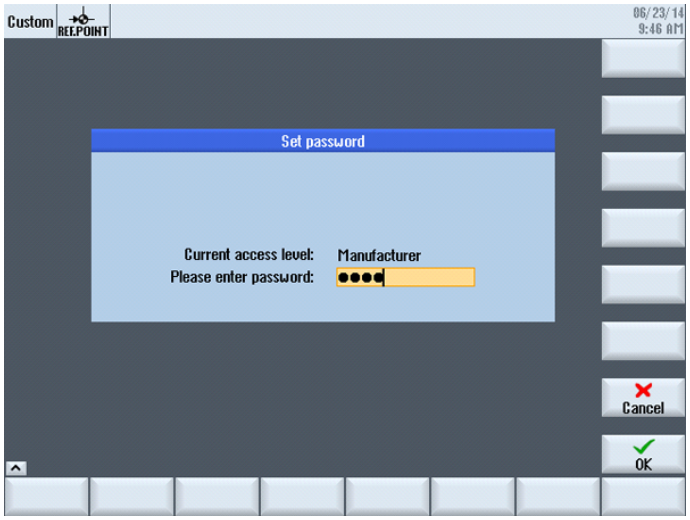


Figure 6-5 Boîte de dialogue Définir le mot de passe

- Modification du mot de passe

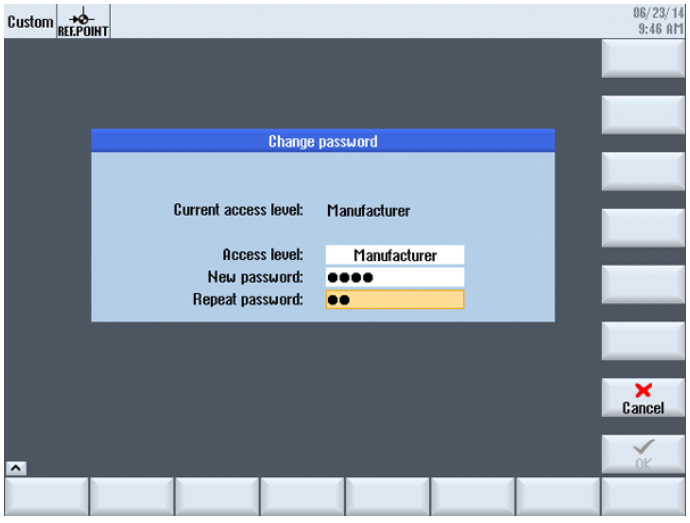


Figure 6-6 Boîte de dialogue Modifier le mot de passe

- Suppression du mot de passe

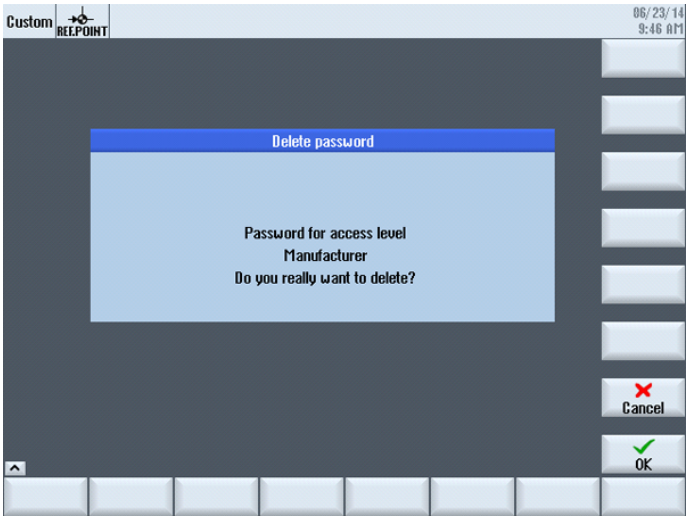


Figure 6-7 Boîte de dialogue Supprimer le mot de passe

Remarque

Ces boîtes de dialogue mettent à votre disposition la fonctionnalité de mot de passe. Les boîtes de dialogue ne correspondent pas aux boîtes de dialogue de SINUMERIK Operate.

Pour appeler les boîtes de dialogue, utiliser la fonction Load Softkey (LS) ou la fonction Load Mask (LM) :

1. Appel avec la fonction Load Softkey (LS) :
   `LS("Passwd","slespasswd.com",1)`
   Extension des touches logicielles verticales :
   - Touche logicielle 1 : Définir le mot de passe
   - Touche logicielle 2 : Modifier le mot de passe
   - Touche logicielle 3 : Supprimer le mot de passe
2. Une autre possibilité consiste à démarrer les trois masques directement en appelant la fonction Load Mask (LM) :
   - Définition du mot de passe : `LM("PWD_SET", "slespasswd.com",1)`
   - Modification du mot de passe : `LM("PWD_CHG", "slespasswd.com",1)`
   - Suppression du mot de passe : `LM("PWD_CLEAR", "slespasswd.com",1)`

6.1.6 Utiliser des images ou des graphiques

Utilisation de graphiques

Il faut distinguer :

- les images ou graphiques dans une zone graphique
- les images d'aide qui peuvent par exemple illustrer différentes variables et qui apparaissent en fondu enchaîné dans la zone graphique.
- les autres images d'aide qui peuvent être configurées à la place de texte court ou du champ E/S et qui peuvent être librement positionnées.

Emplacements

L'image correspondant à la résolution du moniteur raccordé est recherchée dans les répertoires de résolution respectifs dans l'ordre suivant :

[Répertoire système user]`/ico/ico<résolution>`
[Répertoire système oem]`/ico/ico<résolution>`
[Répertoire système addon]`/ico/ico<résolution>`

Si l'image n'est pas affichée ou n'a pas été trouvée, la copier dans l'un des répertoires pour la résolution 640 x 480 pixels :

[Répertoire système user]`/ico/ico640`
[Répertoire système oem]`/ico/ico640`
[Répertoire système addon]`/ico/ico640`

Remarque

Dans les différentes résolutions, les images sont positionnées de manière proportionnelle.

6.2 Définition de la barre de touches logicielles

Définition

L'ensemble des touches logicielles horizontales et verticales est désigné sous le terme de barre de touches logicielles. Des barres supplémentaires peuvent être définies pour remplacer partiellement ou entièrement les barres de touches logicielles existantes.

Le nom des touches logicielles est fixe. Il n'est pas nécessaire de programmer toutes les touches logicielles.

HSx x 1 - 8, Touches logicielles horizontales de 1 à 8

VSy y 1 - 8, Touches logicielles verticales de 1 à 8

Par principe, la description d'une barre de touches logicielles (bloc de description) est structurée ainsi :

Bloc de description	Commentaire	Renvoi au chapitre
`//S...`	;Identifiant de début de la barre de touches logicielles	
`HSx=...`	;Définition des touches logicielles	

`PRESS(HSx)`
`  LM...`
`END_PRESS`
	;Identifiant de début de la méthode
	;Actions
	;Identifiant de fin de la méthode
		voir chapitre Méthodes (Page 121)

`//END`	;Identifiant de fin de la barre de touches logicielles	

Description

Avec la définition de la barre de touches logicielles, des caractéristiques sont également attribuées à une touche logicielle.

Programmation

Syntaxe :	//S*(Descripteur)*
	...
	//END
		;Identifiant de début de la barre de touches logicielles
		
		;Identifiant de fin de la barre de touches logicielles

	Voir aussi le chapitre Syntaxe de configuration étendue (Page 47).	
Description :	Définir la barre de touches logicielles	
Paramètres :	Descripteur	Nom de la barre de touches logicielles
	Texte ou nom de fichier graphique	
Syntaxe :	**TL** = (Texte[, niveau d'accès][, état][, disposition de l'image de la touche logicielle][, disposition du texte par rapport à l'image de la touche logicielle])	
Description :	Définir une touche logicielle	
Paramètres :	TL	Touche logicielle, p. ex. TLH1 à TLH8, TLV1 à TLV8

	Texte	Saisir du texte	
	Nom du fichier image	"\\my_pic.png" ou à l'aide d'un fichier texte différent $85199 p. ex. avec le texte suivant dans le fichier texte (localisé) : 85100 0 0 "\\my_pic.png". La taille de l'image pour la représentation sur une touche logicielle dépend de l'OP utilisé :	
		OP 08 :	640 x 480 mm → 25 x 25 pixels
		OP 010 :	640 x 480 mm → 25 x 25 pixels
		OP 012 :	800 x 600 mm → 30 x 30 pixels
		OP 015 :	1024 x 768 mm → 40 x 40 pixels
		OP 019 :	1280 x 1024 mm → 72 x 72 pixels
	Niveau d'accès	ac0 à ac7 (ac7 : préréglage)	
	Etat	se1 : visible (préréglage) se2 : non activé (écriture grisée) se3 : présélectionné (dernière touche logicielle utilisée)	
	Disposition de l'image de la touche logicielle	PA (PictureAlignment)	
		Valeurs valides : 0 : gauche 1 : droite 2 : centré 3 : en haut (par défaut) 4 : en bas	
	Disposition du texte par rapport à l'image de la touche logicielle	TP (TextAlignedToPicture)	
		Valeurs valides : 0 : le texte n'est pas aligné sur l'image 1 : le texte est aligné sur l'image (par défaut)	

Remarque

Utiliser %n pour effectuer un retour à la ligne dans le texte du libellé de la touche logicielle.

Un maximum de 2 lignes de 9 caractères chacune est disponible.

Attribution d'un niveau d'accès

L'opérateur a uniquement accès aux informations qui correspondent à ce niveau d'accès et aux niveaux d'accès inférieurs (voir également AUTOHOTSPOT).

Exemple

```
//S(Barre1)                                    ; Identifiant de début de la barre de touches logiciel-
                                                 les
```

```
HS1=("NOUVEAU", ac6, se2)                    ; Définition de la touche logicielle HS1, affectation
                                               du libellé "NOUVEAU", du niveau de protection 6 et de
                                               l'état "non activé"
HS2=("\\image1.png")                         ; Affectation d'un graphique à la touche logicielle
HS3=("Exit")
HS4=(["Confirm","\\sk_ok.png"],PA0,TP1)      ; Touche logicielle avec texte et graphique, Tex-
                                               te="Confirm", Image="sk_ok.png", Disposition de l'ima-
                                               ge de la touche logicielle : gauche, le texte est ali-
                                               gné sur l'image

VS1=("Sous-masque")
VS2=($85011, ac7, se2)                       ; Définition de la touche logicielle VS2, affectation
                                               du texte provenant du fichier de langue, du niveau de
                                               protection 1 et de l'état "non activé".
VS3=("Annuler", ac1, se3)                    ; Définition de la touche logicielle VS3, affectation
                                               du libellé "Annulation", du niveau de protection 1 et
                                               de l'état "mis en relief".
VS4=("OK", ac6, se1)                         ; Définition de la touche logicielle VS4, affectation
                                               du libellé "OK", du niveau de protection 6 et de l'état
                                               "visible"
VS5=(SOFTKEY_CANCEL,,se1)                    ; Définition de la touche logicielle VS5 Annuler et af-
                                               fectation de l'état "visible"
VS6=(SOFTKEY_OK,,se1)                        : Définition de la touche logicielle VS6 OK et affec-
                                               tation de l'état "visible"
VS7=(["\\image1.png","Texte OEM"],,se1)      ; Définition de la touche logicielle VS7, affectation
                                               d'un graphique, du libellé "Texte OEM" et de l'état
                                               "visible"
VS8=(["\\image1.png",$83533],,se1)           ; Définition de la touche logicielle VS8, affectation
                                               d'un graphique, d'un texte provenant du fichier de lan-
                                               gue et de l'état "visible"

PRESS(HS1)                                   ; Identifiant de début de la méthode
  HS1.st="Calculer"                          ; Affectation d'un texte de libellé à la touche logi-
                                               cielle
...
END_PRESS                                    ; Identifiant de fin de la méthode

PRESS(RECALL)                                ; Identifiant de début de la méthode
  LM("Masque21")                             ; Chargement de la boîte de dialogue
END_PRESS                                    ; Identifiant de fin de la méthode
//END                                        ; Identifiant de fin de la barre de touches logicielles
```

6.2.1 Modifier les propriétés des touches logicielles en cours d'exécution

Description

Les caractéristiques Texte, Niveau d'accès et Etat d'une touche logicielle peuvent être modifiées en cours d'exécution.

Programmation

Syntaxe :	SK.st = "Texte" SK.ac = niveau d'accès SK.se = état SK.pa = "Disposition de l'image de la touche logicielle" SK.tp = "Disposition du texte par rapport à l'image de la touche logicielle"		;Touche logicielle avec libellé ;Touche logicielle avec niveau de protection ;Touche logicielle avec état ;Touche logicielle avec image ;Touche logicielle avec image et libellé
Description :	Affecter des propriétés		
Paramètres :	Texte		Texte de libellé entre guillemets
	Niveau d'accès		Plage de valeurs : 0 ... 7
	Etat	1: 2: 3:	visible et activé non activé (écriture grisée) présélectionné (dernière touche logicielle utilisée)
	Disposition de l'image de la touche logicielle	0: 1: 2: 3: 4:	gauche droite centré en haut (par défaut) en bas
	Disposition du texte par rapport à l'image de la touche logicielle	0: 1:	le texte n'est pas aligné sur l'image le texte est aligné sur l'image (par défaut)

Exemple

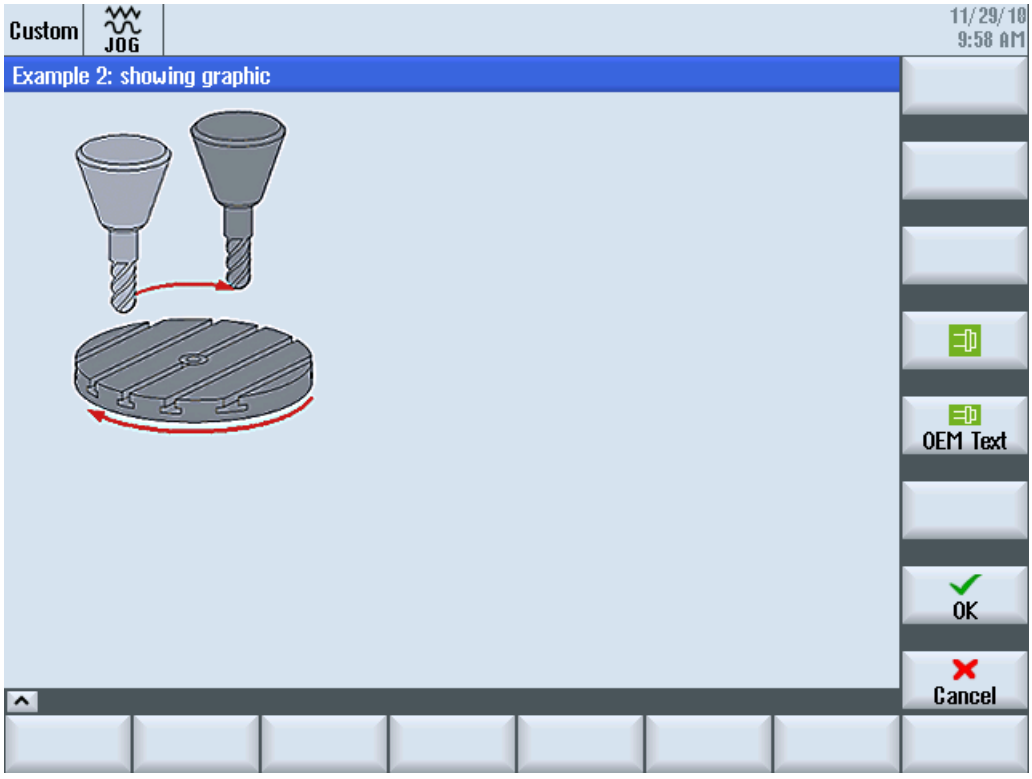


Figure 6-8 Exemple 3 : Graphique et touches logicielles

```
//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 Texte localisé

Vue d'ensemble

Les textes localisés sont utilisés pour :

- libellés des touches logicielles
- titres
- textes d'aide
- tout autre texte souhaité

Les textes localisés pour les boîtes de dialogue sont consignés dans des fichiers texte.

Ces fichiers texte se trouvent dans les répertoires suivants :

- [Répertoire système user]/lng
- [Répertoire système oem]/lng
- [Répertoire système addon]//lng

Remarque

Les fichiers texte doivent être stockés de manière analogue aux fichiers de projet.

Par exemple :

[Répertoire système user]/lng/[fichier texte]

[Répertoire système user]/proj/[fichier de configuration]

alsc.txt textes localisés pour les cycles Siemens standards
almc.txt textes localisés pour les cycles de mesure Siemens

Les fichiers texte utilisés en cours d'exécution du programme sont indiqués dans le fichier "easyscreen.ini" :

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

Le fichier "user.txt" est ici sélectionné comme exemple de fichier texte. En principe, le choix du nom est libre. Suivant la langue des textes contenus dans le fichier, l'abréviation correspondante doit être ajoutée en respectant la syntaxe suivante : Le nom et l'abréviation correspondant à la langue sont séparés par un caractère de soulignement, p. ex. "user_eng.txt".

Remarque

LngFile01 et LngFile02 ne peuvent pas être utilisés pour les textes utilisateur car ils sont déjà attribués dans le fichier "easyscreen.ini" standard.

Voir aussi

AUTOHOTSPOT

Fichiers de langue spécifiques au masque

Pour éviter les chevauchements des plages de nombres utilisées dans les fichiers de langue, seuls des fichiers de langue spécifiques doivent être utilisés dans un masque.

Par conséquent, les fichiers de langue à utiliser sont indiqués dans la ligne de définition de masque, séparés par une virgule. Le fichier indiqué en dernière position est prioritaire (même logique que dans le fichier "easyscreen.ini").

Tous les textes dépendant de la langue utilisés dans le masque sont recherchés en priorité dans ces fichiers de langue. Si le texte n'est pas trouvé dans ces fichiers, la recherche se poursuit ensuite dans les fichiers de texte configurés dans le fichier "easyscreen.ini".

Si aucun fichier de langue n'est défini dans le masque, la recherche est effectuée uniquement dans les fichiers de langue indiqués dans le fichier "easyscreen.ini".

Remarque

Si aucun fichier de langue n'existe dans la langue actuellement sélectionnée, le fichier de langue anglaise est utilisé (par défaut).

Exemples :

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

Cette méthode peut aussi être utilisée pour les touches logicielles d'accès :

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

Format des fichiers texte

Les fichiers texte doivent être enregistrés codés au format UTF-8.

Remarque

Lors de l'enregistrement des fichiers de configuration et de langue, s'assurer que le codage réglé dans l'éditeur utilisé est bien UTF-8.

Forme d'une entrée de texte

Syntaxe :	8xxxx 0 0 "Texte"		
Description :	Affectation de numéro de texte et de texte dans le fichier		
Paramètres :	xxxx	85000 à 89999 900000 à 999999	Plage de numéros d'identification de texte réservée à l'utilisateur. L'attribution des numéros doit être univoque.
	"Texte"		Texte apparaissant dans la boîte de dialogue
	%n		Caractère de commande dans le texte pour saut de ligne

Les deux paramètres 2 et 3 séparés par un espace sont des caractères de commande pour l'édition du texte d'alarme. Ils doivent avoir la valeur zéro pour des raisons d'harmonisation du format de texte avec les textes d'alarme.

Exemples de textes localisés :

```
85000 0 0 "Plan de référence"
85001 0 0 "Profondeur de perçage"
85002 0 0 "Pas de vis"
85003 0 0 "Rayon de la poche"
```

6.3 Configuration de l'aide en ligne

6.3.1 Aperçu

Outre l'aide en ligne exhaustive déjà disponible, vous pouvez créer une aide en ligne constructeur et l'intégrer à SINUMERIK Operate.

Cette aide en ligne est générée au format HTML, c'est-à-dire qu'elle est composée de documents HTML liés entre eux. Le thème recherché est appelé dans une fenêtre séparée via un sommaire ou un index alphabétique. Tout comme dans un logiciel de navigation (par ex. l'explorateur Windows), un récapitulatif est affiché dans la moitié gauche de la fenêtre et vous pouvez cliquer sur le thème souhaité dont l'explication sera affichée dans la moitié droite de la fenêtre.

Une sélection contextuelle des pages d'aide en ligne est impossible.

Marche à suivre

1. Créer des fichiers HTML
2. Créer un manuel d'aide
3. Intégrer l'aide en ligne dans SINUMERIK Operate
4. Archiver les fichiers d'aide

Autres cas d'application

Des aides en ligne relatives aux extensions OEM suivantes peuvent être créées et ajoutées au système d'aide en ligne SINUMERIK Operate :

- Aide en ligne relative aux **cycles et/ou fonctions M** du constructeur, qui complètent les possibilités de programmation des commandes SINUMERIK. L'aide en ligne est appelée de la même façon que l'aide en ligne "Programmation" de SINUMERIK Operate.
- Aide en ligne relative aux **variables OEM** du constructeur. Cette aide en ligne est appelée à partir de la vue des variables de SINUMERIK Operate.

Programmer l'aide en ligne

Pour d'autres aménagements de l'aide en ligne, vous pouvez utiliser le "Lot de programmation SINUMERIK HMI sl". Ce paquet de programmation permet le développement d'applications en langage évolué dans le langage de programmation C++ pour SINUMERIK Operate sur la NCU 7x0.

Remarque

Le "Pack de programmation SINUMERIK HMI sl" est une option logicielle à commander séparément. La documentation correspondante vous est fournie avec le pack de programmation.

6.3.2 Créer des fichiers HTML

Créer les fichiers d'aide au format HTML. Toutes les informations peuvent être archivées dans un seul fichier HTML ou réparties dans plusieurs.

Il est possible d'attribuer soi-même un nom aux fichiers en tenant compte toutefois de ce qui suit :

- Les renvois au sein des fichiers HTML doivent toujours être indiqués avec des chemins relatifs. C'est l'unique façon de garantir que les renvois fonctionnent tout autant sur l'ordinateur de développement que sur le système cible.
- S'il est prévu de sauter par lien à certains points à l'intérieur d'un fichier HTML, il faut définir pour cela des points d'ancrage.
  Exemple de point d'ancrage HTML :
  `<a name="myAnchor">This is an anchor</a>`
- Le contenu des documents HTML doit être archivé avec le codage UTF-8. C'est l'unique façon de garantir que les documents HTML sont bien affichés dans toutes les langues prises en charge par SINUMERIK Operate.
- Les sous-ensembles suivants de l'étendue des fonctions HTML sont pris en charge :

Balises HTML

Balise	Description	Commentaire
a	Anchor or link	Attributs pris en charge : href et name
address	Address	
big	Larger font	
blockquote	Indented paragraph	
body	Document body	Attributs pris en charge : bgcolor (#RRGGBB)
br	Line break	
center	Centered paragraph	
cite	Inline citation	Même effet que balise i
code	Code	Même effet que balise tt
dd	Definition data	
dfn	Definition	Même effet que balise i
div	Document division	Les attributs de bloc par défaut sont pris en charge

Balise	Description	Commentaire
dl	Definition list	Les attributs de bloc par défaut sont pris en charge
dt	Definition term	Les attributs de bloc par défaut sont pris en charge
em	Emphasized	Même effet que balise i
font	Font size, family, color	Attributs pris en charge : size, face, and color (#RRGGBB)
h1	Level 1 heading	Les attributs de bloc par défaut sont pris en charge
h2	Level 2 heading	Les attributs de bloc par défaut sont pris en charge
h3	Level 3 heading	Les attributs de bloc par défaut sont pris en charge
h4	Level 4 heading	Les attributs de bloc par défaut sont pris en charge
h5	Level 5 heading	Les attributs de bloc par défaut sont pris en charge
h6	Level 6 heading	Les attributs de bloc par défaut sont pris en charge
head	Document header	
hr	Horizontal line	Attributs pris en charge : width (peut être indiqué sous forme de valeur absolue ou relative)
html	HTML document	
i	Italic	
img	Image	Attributs pris en charge : src, width, height
kbd	User-entered text	
meta	Meta-information	
li	List item	
nobr	Non-breakable text	
ol	Ordered list	Les attributs par défaut pour listes sont pris en charge
p	Paragraph	Les attributs de bloc par défaut sont pris en charge (réglage par défaut : left-aligned)
pre	Preformated text	
s	Strikethrough	
samp	Sample code	Même effet que balise tt
small	Small font	
span	Grouped elements	
strong	Strong	Corps du texte mis en valeur, remplace la balise b
sub	Subscript	
sup	Superscript	
table	Table	Attributs pris en charge : border, bgcolor (#RRGGBB), cellspacing, cellpadding, width (absolue ou relative), height
tbody	Table body	Sans effet
td	Table data cell	Les attributs par défaut pour cellules de tableau sont pris en charge
tfoot	Table footer	Sans effet
th	Table header cell	Les attributs par défaut pour cellules de tableau sont pris en charge
thead	Table header	Utilisé pour l'impression de tableaux qui s'étendent sur plusieurs pages
title	Document title	
tr	Table row	Attributs pris en charge : bgcolor (#RRGGBB)
tt	Typewrite font	

Balise	Description	Commentaire
u	Underlined	
ul	Unordered list	Les attributs par défaut pour listes sont pris en charge
var	Variable	Même effet que balise tt

Attributs de bloc

Les attributs suivants sont pris en charge par les balises div, dl, dt, h1, h2, h3, h4, h5, h6, p :

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

Attributs par défaut pour listes

Les attributs suivants sont pris en charge par les balises ol et ul :

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

Attributs par défaut pour tableaux

Les attributs suivants sont pris en charge par les balises td et th :

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

Propriétés CSS

Le tableau suivant contient l'étendue des fonctions CSS prises en charge :

Property	Valeurs	Description
background-color	<color>	Couleur d'arrière-plan pour éléments
background-image	<uri>	Image d'arrière-plan pour éléments
color	<color>	Couleur d'avant-plan pour texte
text-indent	<length>px	Indentation de la première ligne d'un paragraphe en pixels
white-space	normal | pre | nowrap | pre-wrap	Définit comment les blancs sont traités dans les documents HTML
margin-top	<length>px	Largeur du bord de marge supérieur en pixels
margin-bottom	<length>px	Largeur du bord de marge inférieur en pixels
margin-left	<length>px	Largeur du bord de marge gauche en pixels
margin-right	<length>px	Largeur du bord de marge droit en pixels

Property	Valeurs	Description
vertical-align	baseline | sub | super | middle | top | bottom	Alignement vertical pour texte (dans les tableaux, seules les valeurs middle, top et bottom sont prises en charge)
border-color	<color>	Couleur du bord pour tables de textes
border-style	none | dotted | dashed | dot-dash | dot-dot-dash | solid | double | groove | ridge | inset | outset	Style du bord pour tables de textes
background	[<'background-color'> || <'background-image'>]	Abréviation pour background Property
page-break-before	[auto | always]	Saut de page avant un paragraphe / un tableau
page-break-after	[auto | always]	Saut de page après un paragraphe / un tableau
background-image	<uri>	Image d'arrière-plan pour éléments

Sélecteurs CSS pris en charge

Toutes les catégories de sélecteurs CSS 2.1 sont prises en charge, à l'exception des pseudo-catégories telles que : first-child, :visited et :hover.

6.3.3 Créer un manuel d'aide

Le manuel d'aide est un fichier XML dans lequel est déterminée la structure de l'aide en ligne. Vous définissez dans ce fichier :

- Documents HTML
- Sommaire et index alphabétique

Syntaxe pour le manuel d'aide

Balise	Quantité	Signification	
HMI_SL_HELP	1	Elément de racine du document XML	
I-BOOK I I I I I I I I I I	+	Désigne un manuel d'aide. Le nom est librement définissable à l'exception des noms prédéfinis par le système (sinumerik_alarm_plc_pmc par exemple). Dans l'exemple, le nom du manuel d'aide est : "hmi_myhelp" Attributs :	
		ref	Désigne le document HTML affiché en guise de page d'accueil pour le manuel d'aide.
		titel	Titre du manuel d'aide affiché dans le sommaire.
		helpdir	Répertoire qui contient l'aide en ligne du manuel d'aide.

Balise	Quantité	Signification	
I-ENTRY II II II II II II II	*	Chapitre de l'aide en ligne Attributs :	
		ref	Désigne le document HTML affiché en guise de page d'accueil pour le chapitre.
		titel	Titre du chapitre d'aide affiché dans le sommaire.
II-INDEX_ENTRY II II II II II II II	*	Mot-clé à afficher Attributs :	
		ref	Désigne le document HTML adressé pour ce mot-clé.
		titel	Titre du mot-clé affiché dans l'index alphabétique.

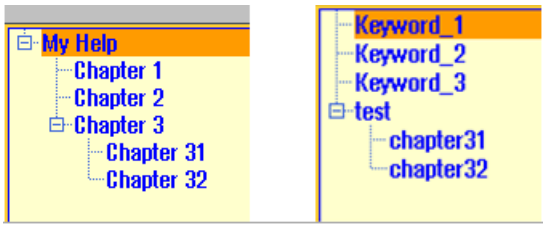
Pour la colonne "Nombre" :
* signifie 0 ou plus
+ signifie 1 ou plus

Exemple de manuel d'aide

L'exemple suivant illustre la structure du manuel d'aide "My Help". En outre, il s'agit de la base pour le sommaire et l'index alphabétique.

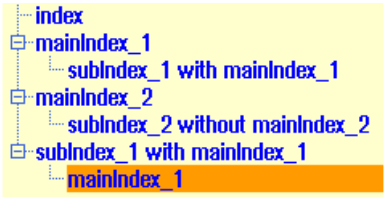
```
<?xml version="1.0" encoding="utf-8"?>
<HMI_SL_HELP language="en-US">
  <BOOK ref="index.html" title="My Help" helpdir="hmi_myhelp">
    <ENTRY ref="chapter_1.html" title="Chapter 1">
      <INDEX_ENTRY ref="chapter_1html#Keyword_1" title="Keyword_1"/>
      <INDEX_ENTRY ref="chapter_1.html#Keyword_2" 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>
```

Le manuel est composé de trois chapitres, le troisième étant divisé en deux sous-chapitres. Les différents mots-clés sont respectivement définis à l'intérieur du chapitre.



Il existe trois modes de formatage pour l'index alphabétique :

1. Entrée unique :
   `<INDEX_ENTRY ...title="index"/>`
2. Deux entrées à deux niveaux, chaque titre ayant une entrée principale et une entrée secondaire. Séparez les entrées par une virgule.
   `<INDEX_ENTRY ...title="mainIndex_1,subIndex_1 with mainIndex_1"/>`
3. Entrée à deux niveaux, le premier titre étant l'entrée principale et le deuxième l'entrée secondaire. Séparez les entrées par un point-virgule.
   `<INDEX_ENTRY ...title="mainIndex_2;subIndex_2 without mainIndex_1"/>`



6.3.4 Intégrer l'aide en ligne dans SINUMERIK Operate

Si vous souhaitez intégrer le manuel d'aide dans l'aide en ligne de SINUMERIK Operate, vous avez besoin du fichier "slhlp.xml".

Description du format de "slhlp.xml"

Balise	Quantité	Signification
CONFIGURATION	1	Élément de racine du document XML. Indique qu'il s'agit d'un fichier de configuration.
I-OnlineHelpFiles	1	Introduit la section des manuels d'aide en ligne.
II-<help_book>	*	Introduit la section d'un manuel d'aide.

Balise	**Quantité**	**Signification**	
III-EntriesFile III III III III III	1	Nom du fichier du manuel d'aide avec le sommaire et l'index alphabétique. Attributs :	
		value	Nom du fichier XML
		type	Type de données de la valeur (QString)
III-Technology III III III III III III III III III IIII	0, 1	Indique la technologie pour laquelle le manuel d'aide est valable. "All" s'applique à toutes les technologies. Si le manuel d'aide s'applique à plusieurs technologies, il faut les séparer par une virgule. Valeurs possibles : All, Universal, Milling, Turning, Grinding, Stroking, Punching Attributs :	
		value	Indication de la technologie
		type	Type de données de la valeur (QString)
III -DisableSearch III III III III III	0, 1	Désactiver la recherche par mot-clé pour le manuel d'aide. Attributs :	
		value	true, false
		type	Type de données de la valeur (bool)
III-DisableFullTextSearch III III III III	0, 1	Désactiver la recherche dans tout le texte pour le manuel d'aide. Attributs :	
		value	true, false
		type	Type de données de la valeur (bool)
III-DisableIndex III III III III	0, 1	Désactiver l'index alphabétique pour le manuel d'aide. Attributs :	
		value	true, false
		type	Type de données de la valeur (bool)
III-DisableContent III III III III	0, 1	Désactiver le sommaire pour le manuel d'aide. Attributs :	
		value	true, false
		type	Type de données de la valeur (bool)
III-DefaultLanguage III III III III III	0, 1	Code pour la langue à afficher si la langue nationale actuelle est disponible pour le manuel d'aide. Attributs :	
		value	chs, deu, eng, esp, fra, ita, ...
		type	Type de données de la valeur (QString)

Pour la colonne "Nombre" :
* signifie 0 ou plus

Exemple d'un fichier "slhlp.xml"

L'exemple suivant permet de mettre à disposition le manuel d'aide "hmi_myhelp.xml" dans SINUMERIK Operate.

L'index alphabétique n'est pas activé pour le manuel d'aide.

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

6.3.5 Archiver les fichiers d'aide

Archiver les fichiers d'aide dans le système cible

1. Ouvrez le répertoire /**oem**/sinumerik/hmi/hlp et créez un nouveau dossier pour la langue souhaitée. Utilisez pour cela le code de langue spécifié.
   Il est impératif d'écrire les noms de dossier en minuscules.
   Si vous souhaitez intégrer par ex. une aide pour l'allemand et l'anglais, créez les dossiers "deu" et "eng".
2. Créez le manuel d'aide, par ex. "hmi_myhelp.xml" respectivement dans le dossier "deu" et "eng".
3. Copiez les fichiers d'aide dans les répertoires, par ex. /**oem**/sinumerik/hmi/hlp/deu/hmi_myhelp pour les fichiers d'aide en allemand et /**oem**/sinumerik/hmi/hlp/eng/hmi_myhelp pour les fichiers d'aide en anglais.
4. Collez le fichier de configuration "slhlp.xml" dans le répertoire /**oem**/sinumerik/hmi/cfg.
5. Redémarrez l'IHM.

Remarque

Lors de l'affichage des sommaire et index alphabétique d'un manuel d'aide, les fichiers d'aide sont archivés en format binaire (slhlp_<manuel_aide_*.hmi) dans le répertoire /siemens/sinumerik/sys_cache/hmi/hlp pour permettre un traitement plus rapide. Si vous modifiez le manuel d'aide, vous devez toujours effacer ces fichiers.

6.3.6 Fichiers d'aide au format PDF

En plus des fichiers d'aide au format HTML, vous pouvez également intégrer des informations au format PDF dans le logiciel de commande. Des fichiers d'aide au format PDF peuvent être ouverts à l'aide de raccourcis depuis la table des matière ou l'index ou directement depuis des fichiers HTML.

Enregistrement de fichiers d'aide PDF

Copiez les fichiers PDF dans l'un des répertoires suivants :

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

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

Intégration de fichiers d'aide au format PDF

Intégrez les fichiers avec l'extension "pdf" dans des configurations de boîte de dialogue ou configurations de table des matières ou d'index, exactement comme avec l'extension "html" :

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

Créez une référence à l'aide au format PDF depuis les fichiers HTML sous la forme d'un lien :

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

Remarque

Dans l'aide au format PDF il n'est pas possible de sélectionner des repères de saut contextuels ou d'effectuer des sauts vers d'autres fichiers HTML ou PDF.

La fonction de recherche est possible uniquement au sein d'un fichier PDF. Une recherche de niveau supérieur dans plusieurs fichiers d'aide au format PDF n'est pas prise en charge.

Variables

7

7.1 Définition des variables

Valeur de variable

La principale caractéristique d'une variable est la valeur de la variable.

Il est possible d'affecter la valeur des variables par :

- préréglage lors de la définition des variables
- affectation d'une variable système ou utilisateur
- une méthode

Programmation

Syntaxe :	Descripteur.**val** = valeur de variable Descripteur = valeur de variable	
Description :	valeur de variable val (value)	
Paramètres :	Descripteur :	Nom de la variable
	valeur de variable :	valeur de la variable
Exemple :	`VAR3 = VAR4 + SIN(VAR5)` `VAR3.VAL = VAR4 + SIN(VAR5)`	

Etat de la variable

La caractéristique Etat de la variable permet de demander en cours d'exécution si une variable contient une valeur valide. Cette propriété peut être lue et écrite avec la valeur FALSE = 0.

Programmation

Syntaxe :	descripteur.**vld**	
Description :	état de la variable vld (validation)	
Paramètres :	Descripteur :	Nom de la variable
		Le résultat de l'interrogation peut être :
	FALSE =	valeur non valide
	TRUE =	valeur valide

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

7.2 Exemples d'application

Variables auxiliaires

Les variables auxiliaires sont des variables de calcul internes. Les variables de calcul sont définies comme les variables, mais ne possèdent aucune caractéristique en dehors de la valeur de variable et de l'état. Ainsi, les variables auxiliaires ne sont pas visibles dans la boîte de dialogue. Les variables auxiliaires sont de type VARIANT.

Programmation

Syntaxe :	DEF [Descripteur]	
Description :	Les variables de calcul internes sont de type VARIANT.	
Paramètres :	Descripteur :	Nom de la variable auxiliaire
Exemple :	`DEF OTTO ;Définition d'une variable auxiliaire`	

Syntaxe :	Descripteur.val = Valeur de variable auxiliaire Descripteur = Valeur de variable auxiliaire	
Description :	La valeur d'une variable auxiliaire est attribuée dans une méthode.	
Paramètres :	Descripteur :	Nom de la variable auxiliaire
	Valeur de variable auxiliaire :	Contenu de la variable auxiliaire

Exemple :

```
LOAD
  OTTO = "Test"             : Affectation de la valeur "Test" à la variable auxiliai-
END_LOAD                    re Otto
LOAD
  OTTO = REG[9].VAL         ; Affectation de la valeur de registre à la variable au-
END_LOAD                    xiliaire Otto
```

Calcul avec des variables

Les variables sont calculées après chaque sortie d'un champ de saisie et de visualisation (à l'aide de la touche ENTER ou Toggle). Le calcul est configuré dans une méthode CHANGE et lancé à chaque modification de la valeur.

Si une variable possède une valeur valide, il est possible d'interroger l'état de la variable, par exemple :

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

Indiquer le chemin d'une variable système de façon indirecte

Le chemin d'une variable système peut également être indiqué indirectement, c'est-à-dire en fonction d'une autre variable :

```
PRESS(HS1)
  ACHSE=ACHSE+1
  WEG.VAR="$AA_DTBW["<<ACHSE<<"]"       ;Indiquer l'adresse d'axe avec une variable
END_PRESS
```

Modification du libellé des touches logicielles

Exemple :

```
HS3.st = "Nouveau texte"                ;Modifier le libellé des touches logicielles
```

7.3 Exemple 1 : attribution de type de variable, de texte, d'image d'aide, de couleurs, d'infobulles

Exemple 1a

L'exemple suivant définit une variable pour laquelle les propriétés Type de variable, Texte, Image d'aide et Couleurs sont définies.

DEF Var1 = (R///,"Valeur réelle",,"mm"//"Var1.png"////8,2)			
	Type de variable :		REAL
	Textes :		
		Texte court :	valeur réelle
		Texte d'unité :	mm
	Image d'aide :		Var1.png
	Couleurs :		
		Couleur de premier plan :	8 (marron)
		Couleur d'arrière-plan :	2 (orange)

Exemple 1b

L'exemple suivant définit une variable pour laquelle les propriétés Type de variable, Préréglage, Texte, Info-bulle, Mode de saisie et Position du texte court sont définies.

DEF Var2 = (I//5/"","Valeur","","," Texte d'info-bulle"/wr2///20,250,50)			
	Type de variable :		INTEGER
	Valeur par défaut :		5
	Textes :		
		Texte court :	Valeur (ID du texte langue possible)
		Info-bulle :	Texte de l'info-bulle
	Attributs :		
		Mode de saisie	Lecture et écriture
	Position texte court :		
		Marge à gauche	20
		Marge supérieure	250
		Largeur :	50

Voir aussi

Paramètres de variables (Page 93)

7.4 Exemple 2 : attribution de type de variable, de valeurs limites, d'attributs, de position du texte court

Exemple 2

L'exemple suivant définit une variable pour laquelle les propriétés Type de variable, Valeurs limites, Mode de saisie, Orientation et Position sont définies.

DEF Var2 = (I/0,10///wr1,al1/// , ,300)			
	Type de variable :		INTEGER
	Valeurs limites ou entrées de champ bascule :		MIN : 0 MAX : 10
	Attributs :		
		Mode de saisie	lecture seule
		Orientation du texte court	justifié à droite
	Position texte court :		
		Largeur :	300

Voir aussi

Paramètres de variables (Page 93)

7.5 Exemple 3 : attribution de type de variable, de valeurs par défaut, de variable système ou utilisateur, de position du champ de saisie et de visu

Exemple 3

L'exemple suivant définit une variable pour laquelle les propriétés Type de variable, Préréglage, Variable système ou utilisateur et Position sont définies.

DEF Var3 =(R//10////"$R[1]"//300,10,200//")			
	Type de variable :		REAL
	Valeur par défaut :		10
	Variable système ou utilisateur :		$R[1] (Paramètre R 1)
	Position texte court :		Position standard par rapport au champ de saisie et de visualisation
	Position du champ de saisie et de visualisation :		
		Marge à gauche	300
		Marge supérieure	10
		Largeur :	200

Voir aussi

Paramètres de variables (Page 93)

7.6 Exemple 4 : champ bascule et champ de liste

Exemple 4a

Différentes entrées avec le champ bascule :

```
DEF Var1 = (I/* 0,1,2,3)                ;Champ bascule simple pour basculer entre des valeurs numériques
DEF Var2 = (S/* "On", "Off")            ;Champ bascule simple pour basculer entre des chaînes de caractères
DEF Var3 = (B/* 1="On", 0="Off")        ;Champ bascule étendu pour basculer entre des valeurs numériques avec un texte d'affichage affecté à chaque valeur numérique
DEF Var4 = (R/* ARR1)                   ;Champ bascule qui récupère ses valeurs à basculer à partir d'un tableau
```

Exemple 4b

Le champ de liste correspond à la configuration d'un champ bascule, mais il faut en outre définir le type d'affichage du champ de liste (attribut de variable DT = 4).

DEF VAR_LISTBOX_Text = (S/*$80000,$80001,$80002,$80003,$80004/$80001//DT4////200,,340,60)			
	Type de variable :		STRING
	Valeurs limites / champ bascule :		*$80000,$80001,$80002,$80003,$80004 (liste des textes localisés à afficher)
	Valeur par défaut :		$80001
	Attributs :		
		Type d'affichage :	4 (champ de liste)
	Position du champ de saisie et de visualisation :		
		Marge à gauche :	200
		Largeur :	340
		Hauteur :	60
	Couleurs :		
		Couleur de premier plan :	6 (bleu)
		Couleur d'arrière-plan :	10 (blanc)

7.7 Exemple 5 : affichage d'une image

Exemple 5

Afficher une image à la place d'un texte court : la taille et la position de l'image sont indiquées sous "Position champ de saisie et de visualisation (gauche, haut, largeur, hauteur)"

DEF VAR6= (V///,"\\image1.png" ////160,40,50,50)			
	Type de variable :		VARIANT
	Textes :		
		Texte court :	image1.png
	Position champ de saisie et de visualisation :		
		Marge à gauche :	160
		Marge supérieure :	40
		Largeur :	50
		Hauteur :	50

7.8 Exemple 6 : barre de progression

La barre de progression est un type d'affichage spécial du champ de saisie et de visualisation qui n'est conçu que pour un affichage sans saisie.

Il existe deux types de barres de progression :

1. Les barres de progression avec jusqu'à deux changements de couleur, par exemple pour afficher la température ou la charge (voir exemple 6a)
2. Les barres de progression qui affichent une progression (sans changement de couleur) de type Operate (voir exemple 6b)

Exemple 6a

Barres de progression avec deux changements de couleur :

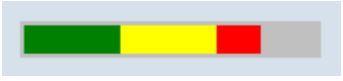


Figure 7-1 Barres de progression avec deux changements de couleur

DEF PROGGY0 = (R/0,150,50,100///DT1,DO0//"$R[10]"//,,150/3,4,,,,,,9,7)		
Type de variable :		REAL
Valeurs limites / champ bascule :		
	MIN :	0
	MAX :	150
	Valeur de signal SVAL1 :	50
	Valeur de signal SVAL2 :	100
Attributs :		
	Mode d'affichage DT :	1 (barre de progression)
	Option de visualisation DO :	0 (de gauche à droite (par défaut))
Variable système ou utilisateur :		$R[10]
Position du champ de saisie et de visualisation :		
	Largeur :	150
Couleurs :		
	Couleur de premier plan :	3 (vert foncé)
	Couleur d'arrière-plan :	4 (gris clair)
	Couleur de signal SC1 :	9 (jaune)
	Couleur de signal SC2 :	7 (rouge)

Pour utiliser une barre de progression avec changement de couleur, le mode d'affichage DT (DisplayType) doit être réglé sur 1.

L'orientation de la barre de progression est déterminée par l'attribut Option de visualisation DO (DisplayOption) :

0 : de gauche à droite (par défaut)

1 : de droite à gauche

2 : de bas en haut

3 : de haut en bas

Pour la représentation de la barre de progression, une valeur MIN et une valeur MAX doivent être définies (dans l'exemple MIN : 0, MAX : 150).

Ce réglage permet déjà d'afficher une barre de progression avec la couleur de premier plan 3 (= vert foncé) et la couleur d'arrière-plan 4 (= gris clair) en fonction de la valeur actuelle de la variable PROGGY0.

En option, une ou deux valeurs de signal SVAL1 et SVAL2 (paramètre Valeur limite) peuvent être définies (dans l'exemple SVAL1 : 50 et SVAL2 : 100). La couleur de premier plan de la barre de progression change pour ces valeurs de signal. Les couleurs de signal correspondantes sont déterminées par les paramètres SC1 et SC2 (dans l'exemple ci-dessus SC1 : 9 (= jaune) et SC2 : 7 (= rouge)

La règle suivante s'applique pour l'indication des valeurs limites pour la barre de progression :

MIN < SVAL1 < SVAL2 < MAX.

Exemple 6b

Barre de progression sans changement de couleur :

Figure 7-2 Barre de progression

DEF PROGGY0 = (R/0,150///DT2,DO0//"$R[10]"//,,150/6,10)		
Type de variable :		REAL
Valeurs limites / champ bascule :		
	MIN :	0
	MAX :	150
Attributs :		
	Type d'affichage DT :	2 (barre de progression)
	Option de visualisation DO :	0 (de gauche à droite (par défaut))
Variable système ou utilisateur :		$R[10]
Position du champ de saisie et de visualisation :		
	Largeur :	150
Couleurs :		
	Couleur de premier plan :	6 (bleu)
	Couleur d'arrière-plan :	10 (blanc)

Pour utiliser une barre de progression sans changement de couleur, le mode d'affichage DT (DisplayType) doit être réglé sur 2.

L'orientation de la barre de progression est déterminée par l'attribut Option de visualisation DO (DisplayOption) (voir explication de l'exemple 6a).

Pour la représentation de la barre de progression, une valeur MIN et une valeur MAX doivent être définies (dans l'exemple MIN : 0, MAX : 150).

Ce réglage permet d'afficher une barre de progression avec la couleur de premier plan 6 (= bleu) et la couleur d'arrière-plan 10 (= blanc) en fonction de la valeur actuelle de la variable PROGGY0.

7.9 Exemple 7 : mode de saisie de mot de passe (astérisque)

Exemple 7

Pour réaliser un champ à saisie masquée, par exemple pour la saisie d'un mot de passe, le mode d'affichage DT doit être réglé sur 5. Des astérisques remplacent les caractères saisis.

Figure 7-3 Mode de saisie de mot de passe (astérisque)

DEF VAR_SET_PWD=(S//""//DT5)			
	Type de variable :		STRING
	Préréglage :		Chaîne vide
	Attributs :		
		Mode d'affichage DT :	5 (Mode de saisie de mot de passe)

7.10 Paramètres de variables

Vue d'ensemble des paramètres

Dans la vue d'ensemble suivante, les paramètres des variables sont brièvement présentés. Vous trouverez une description détaillée dans les chapitres suivants.

Paramètres	**Description**	
Type de variable (Page 100)	Le type de variable doit être indiqué.	
	R[x] :	REAL (+ chiffre après la virgule)
	I :	INTEGER
	S[x] :	STRING (+ chiffre pour longueur d'une chaîne de caractères)
	C :	CHARACTER (caractère unique)
	B :	BOOL
	V :	VARIANT

Paramètres	Description	
Valeurs limites (Page 88)	Valeur limite MIN, valeur limite MAX Préréglage : vide Les valeurs limites sont séparées par une virgule. Les valeurs limites peuvent être indiquées au format décimal ou sous forme de caractères du type "A", "F", pour les types I, C et R.	
	Pour la représentation d'une barre de progression avec changement de couleur (attribut de variable DT = 1), il est possible de configurer en option deux couleurs de signal SC1 et SC2 (Signal Color) qui sont utilisées comme couleur de premier plan de la barre en cas de dépassement des valeurs de signal SVAL1 ou SVAL2 (Signal Value). La valeur de signal est indiquée sous forme de valeur entière. Voir l'exemple d'utilisation concernant la barre de progression (Page 90).	
Valeur par défaut (Page 104)	Si aucune valeur par défaut n'est configurée et qu'aucune variable système ou utilisateur n'est affectée aux variables, le premier élément du champ bascule est attribué. Si aucun champ bascule n'est défini, aucun préréglage n'est effectué. La variable prend l'état "non calculée". Préréglage : aucune valeur par défaut	
Champ bascule (Page 102)	Liste avec des entrées indiquées dans le champ de saisie et de visualisation : La liste est introduite par *, les entrées sont séparées par des virgules. Une valeur peut être affectée aux entrées. L'entrée pour la valeur limite est interprétée comme une liste par le champ bascule. Si seulement une * est saisie, un champ bascule variable est créé. Préréglage : aucun	
Textes (Page 87)	L'ordre est défini à l'avance. A la place du texte court, une image peut également être affichée. Préréglage : vide	
	Texte long :	texte de la ligne d'affichage
	Texte court :	nom de l'élément de boîte de dialogue
	Texte graphique :	le texte se rapporte aux termes du graphique
	Texte d'unité :	unité de l'élément de boîte de dialogue
	Info-bulle	Elles servent d'information courte dans une configuration de masque pour les champs d'affichage et Toggle. L'information est configurée via le texte clair et l'ID de texte langue.

Paramètres	Description	
Attributs (Page 88)	Les attributs influencent les caractéristiques suivantes : • Mode d'affichage • Option de visualisation • Vitesse de rafraîchissement • Symbole de basculement • Info-bulle • Mode de saisie • Niveau d'accès • Orientation du texte court • Taille des caractères • Valeurs limites Les attributs sont séparés par une virgule, l'ordre n'a pas d'importance. Il est possible de paramétrer individuellement chaque composant.	
	Mode d'affichage	dt0 : par défaut (champ de saisie et de visualisation ou champ bascule) (default) dt1 : barre de progression (progress bar) avec changement de couleur dt2 : Barre de progression (progress bar) sans changement de couleur de type Operate dt4 : Champ de liste dt5 : Mode de saisie de mot de passe (astérisque)
	Option de visualisation	En particulier p. ex. pour les modes d'affichage dt1 et dt2 (barre de progression), il peut s'avérer nécessaire de configurer également une option de visualisation en association avec le mode d'affichage. do0 : de gauche à droite (par défaut) do1 : de droite à gauche do2 : de bas en haut do3 : de haut en bas
	Vitesse de rafraîchissement	L'attribut UR (update rate) permet de contrôler le rafraîchissement de l'affichage et donc l'exécution du bloc CHANGE configuré pour des variables ou des colonnes de grille. Selon la configuration, la charge de la CPU peut être considérablement réduite ce qui permet d'obtenir des temps de réponse beaucoup plus courts au niveau de l'interface utilisateur. ur0 : SlCap::standardUpdateRate() (actuellement = 200 ms, valeur par défaut) ur1 : 50 ms ur2 : 100 ms ur3 : 200 ms ur4 : 500 ms ur5 : 1000 ms ur6 : 2000 ms ur7 : 5000 ms ur8 : 10000 ms
	Symbole de basculement	tg0 : Symbole de basculement désactivé (par défaut) tg1 : Symbole de basculement activé

Paramètres	Description	
		Si l'attribut TG est réglé sur 1, le symbole de basculement apparaît également dans l'info-bulle du champ de saisie. Exemple : `DEF OFFS = (R//123.456/,,,,"My ToolTip"/TG1)`

	Info-bulle	Le texte de l'info-bulle peut également être modifié en cours d'exécution à l'aide de la propriété de variable "TT".
		Exemple :
		`PRESS(VS1)`
		`–  MyVar.TT = "My new ToolTip"`
		`END_PRESS`
		ou
		`DEF MyVar=(R///,,,,"My ToolTip text")`

	Mode de saisie	wr0 : champ de saisie et de visualisation non visible, texte court visible wr1 : lire (aucun focus possible pour la saisie) wr2 : lire et écrire (la ligne apparaît en blanc) wr3 : wr1 avec focus wr4 : tous les éléments des variables non visibles, aucun focus possible wr5 : la valeur saisie est immédiatement enregistrée après chaque activation de touche (contrairement à wr2, où l'enregistrement a lieu après avoir quitté le champ ou avec l'activation de la touche RETURN). Préréglage : wr2
	Niveau d'accès	vide : toujours inscriptible ac0...ac7 : Niveaux de protection Si le niveau d'accès n'est pas suffisant, la ligne apparaît en gris, réglage par défaut : ac7
	Orientation du texte court	al0 : justifié à gauche al1 : justifié à droite al2 : centré Préréglage : al0
	Taille des caractères	fs1 : Taille des caractères par défaut (8 Pt) fs2 : Taille des caractères double Préréglage : fs1 L'espacement entre les lignes est défini. Pour la taille des caractères standard, la boîte de dialogue peut contenir 16 lignes. Le texte du graphique et des unités peut être configuré uniquement en taille standard de caractères.
	Valeurs limites	Permet de vérifier si la valeur de la variable se trouve entre les limites MIN et MAX indiquées. Préréglage : en fonction des valeurs limites indiquées li0 : pas de contrôle li1 : contrôle par rapport à Min li2 : contrôle par rapport à Max li3 : contrôle par rapport à Min et Max
	Comportement lors de l'ouverture	Les attributs cb, indiqués pour une définition de variables, ont la priorité pour la variable par rapport aux données globales cb de la définition de

Paramètres	Description	
		la boîte de dialogue. Lorsqu'il y a plusieurs attributs, ils sont séparés par des virgules. (Voir aussi AUTOHOTSPOT)
	Appel de la méthode CHANGE	CB0 : La méthode CHANGE est déclenchée lors de l'affichage du masque si la variable a une valeur valide à ce moment-là (p. ex. par un préréglage ou une variable CN/AP). CB1 : La méthode CHANGE n'est pas déclenchée de manière explicite lors de l'affichage du masque. Si la variable contient une variable CN/AP configurée, la méthode CHANGE est évidemment appelée malgré tout.
	Empty Input (EI)	Avec l'attribut de variable "EI" (= Empy Input), la réaction du champ de saisie en cas de chaîne de caractères vide peut être configurée. EI0 : Standard, c'est-à-dire que les entrées sont acceptés dans la zone de saisie. EI1 : en cas d'entrée vide, le champ de saisie est en état d'erreur et la valeur précédente valable est réactivée (undo).
Image d'aide (Page 87)	Fichier de l'image d'aide :	Nom du fichier png Préréglage : vide
	Le nom du fichier de l'image d'aide est indiqué entre guillemets. L'image est automatiquement du graphique affichée (à la place du graphique utilisé jusqu'à présent) lorsque le curseur passe sur cette variable.	
Variable système ou utilisateur (Page 89)	Des paramètres système ou utilisateur de la CN ou de l'AP peuvent être affectées aux variables. La variable système ou utilisateur est indiquée entre guillemets. Bibliographie : Tables de paramètres Variables système, /PGAsl/	
Position texte court (Page 105)	Position du texte court (marge à gauche, marge en haut, largeur) Les positions sont données en pixels par rapport au coin supérieur gauche de la partie principale de la boîte de dialogue. Les indications sont séparées par une virgule.	
Position champ de saisie et de visualisation (Page 105)	Position du champ de saisie et de visualisation (marge à gauche, marge en haut, largeur) Les positions sont données en pixels par rapport au coin supérieur gauche de la partie principale de la boîte de dialogue. Les indications sont séparées par une virgule. Si cette position est modifiée, les positions du texte court, du texte graphique et du texte d'unité sont également modifiées.	

Paramètres	Description
Couleurs (Page 87)	Couleur de premier plan et d'arrière-plan pour les champs de saisie/visualisation, textes courts, textes graphiques, textes d'unité et couleurs de signal pour les barres de progression : Les couleurs sont séparées par une virgule. Pour l'indication de la couleur, voir le chapitre AUTOHOTSPOT. Préréglage pour champ de saisie/visualisation : Couleur de premier plan : noir, couleur d'arrière-plan : blanc Les couleurs par défaut du champ de saisie et de visualisation dépendent du mode d'écriture "wr" : Préréglage pour texte court, texte graphique, texte d'unité : Couleur de premier plan : noir, couleur d'arrière-plan : transparent.
	Pour la représentation d'une barre de progression avec changement de couleur (attribut de variable DT = 1), il est possible de configurer en option deux couleurs de signal SC1 et SC2 (Signal Color) qui sont utilisées comme couleur de premier plan de la barre en cas de dépassement des valeurs de signal SVAL1 ou SVAL2 (Signal Value). Voir l'exemple d'utilisation concernant la barre de progression (Page 90).
	Dans la définition de variable, les couleurs sont attendues dans l'ordre suivant : 1. Couleur de premier plan du champ de saisie et de visualisation : FC 2. Couleur d'arrière-plan du champ de saisie et de visualisation : BC 3. Couleur de premier plan du texte court : FC_ST 4. Couleur d'arrière-plan du texte court : BC_ST 5. Couleur de premier plan du texte graphique : FC_GT 6. Couleur d'arrière-plan du texte graphique : BC_GT 7. Couleur de premier plan du texte d'unité : FC_UT 8. Couleur d'arrière-plan du texte d'unité : BC_UT 9. Couleur de signal 1 10. Couleur de signal 2
Fichier d'aide en ligne	Le nom du fichier d'aide en ligne est indiqué entre guillemets.

Champ de saisie/visualisation intégrant la sélection des unités	Les champs de saisie/visualisation intégrant la sélection des unités permettent de commuter entre différentes unités. Quand on navigue dans le champ de saisie avec le curseur, la sélection intégrée des unités est mise en évidence (il n'est pas nécessaire de placer explicitement le focus dessus). En outre, une info-bulle s'affiche avec un symbole de basculement et donne une indication décrivant cette fonctionnalité.
	Exemples :
	`DEF VarEdit=(R////////200,,100///"VarTgl")`
	`VarTgl=(S/*0="mm",1="inch"/0//WR2////302,,40)`
	ou
	`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 : modifier les caractéristiques

Lors d'une modification, une nouvelle valeur est affectée aux variables avec la séquence descripteur.propriété = valeur. L'expression située à droite du signe d'égalité est évaluée et attribuée à la variable ou à la propriété de la variable.

Descripteur.**ac** = niveau d'accès	(ac : access level)
Descripteur.**al** = orientation du texte	(al : alignment)
Descripteur.**bc** = couleur d'arrière-plan du champ de saisie/visualisation	(bc : back color)
Descripteur.**bc_gt** = couleur d'arrière-plan du texte graphique	(bc : back color) (gt : graphic text)
Descripteur.**bc_st** = couleur d'arrière-plan du texte court	(bc : back color) (st : short text)
Descripteur.**bc_ut** = couleur d'arrière-plan du texte d'unité	(bc : back color) (ut : unit text)
Descripteur.**do** = option de visualisation	(do : display option)
Descripteur.**dt** = mode d'affichage	(dt : display type)
Descripteur.**.fc** = couleur de premier plan du champ de saisie/visualisation	(fc : front color)
Descripteur **fc_gt** = couleur de premier plan du texte graphique	(fc : front color) (gt : graphic text)
Descripteur.**fc_st** = couleur de premier plan du texte court	(fc : front color) (st : short text)
Descripteur.**fc_ut** = couleur de premier plan du texte d'unité	(fc : front color) (ut : unit text)
Descripteur.**fs** = taille de la police	(fs : font size)
Descripteur.**gt** = texte du graphique	(gt : graphic text)
Descripteur.**hlp** = image d'aide	(hlp : help)
Descripteur.**li** = valeur limite	(li : limit)
Descripteur.**lt** = texte long	(lt : long text)
Descripteur.**max** = valeur limite MAX	(max : maximum)
Descripteur.**min** = valeur limite MIN	(min : minimum)
Descripteur.**sc** = couleur de signal	(sc : signal color)
Descripteur.**st** = texte court	(st : short text)
Descripteur.**tg** = symbole de basculement	(tg : toggle)
Descripteur.**tt** = info-bulle	(tt : tooltip
Descripteur.**typ** = type de variable	(typ : type)
Descripteur.**ur** = vitesse de rafraîchissement	(ur : update rate)
Descripteur.**ut** = texte d'unité	(ut : unit text)
Descripteur.**val** = valeur de variable	(val : value)
Descripteur.**var** = variable système ou utilisateur	(var : variable)
Descripteur.**vld** = état de la variable	(vld : validation)
Descripteur.**wr** = mode de saisie	(wr : write)

Voir aussi

Syntaxe de configuration étendue (Page 47)

7.11 Détails relatifs au type de variable

Type de variable INTEGER

Les extensions suivantes sont possibles pour le type "INTEGER" pour la détermination de la représentation dans le champ de saisie et de visualisation et de l'utilisation de la mémoire.

2ème caractère dans le type de données d'extension

Format d'affichage	
B	binaire
D	décimal à signe
H	hexadécimal
Pas d'indication	décimal à signe

3ème et/ou 4ème caractère dans le type de données d'extension

Utilisation de la mémoire	
B	Octet
W	Mot
D	Double mot
BU	Octets sans signe
WU	Mot sans signe
DU	Double mot sans signe

Ordre des caractères pour le type de données INTEGER

1. "I" Identification systématique en tant qu'INTEGER
2. Format d'affichage
3. Utilisation de la mémoire
4. "U" sans signe

Définitions de type INTEGER valide :	
IB	Variable entier 32 bits format binaire
IBD	Variable entier 32 bits format binaire
IBW	Variable entier 16 bits format binaire
IBB	Variable entier 8 bits format binaire
I	Variable entier 32 bits format décimal avec signe
IDD	Variable entier 32 bits format décimal avec signe
IDW	Variable entier 16 bits format décimal avec signe
IDB	Variable entier 8 bits format décimal avec signe
IDDU	Variable entier 32 bits format décimal sans signe
IDWU	Variable entier 16 bits format décimal sans signe
IDBU	Variable entier 8 bits format décimal sans signe

Définitions de type INTEGER valide :	
IH	Variable entier 32 bits format hexadécimal
IHDU	Variable entier 32 bits format hexadécimal
IHWU	Variable entier 16 bits format hexadécimal
IHBU	Variable entier 8 bits format hexadécimal

Type de variable VARIANT

Le type de variable VARIANT est déterminé par le type de données de la dernière affectation de valeur. Si la valeur attribuée ou saisie commence par '-'. '+', '.' ou par un chiffre ('0'-'9'), alors la valeur est interprétée comme étant numérique. Dans tous les autres cas elle est interprétée comme une chaîne de caractères.

Il peut être appelé à l'aide de la fonction ISNUM ou ISSTR. Le type VARIANT est principalement adapté à l'écriture en code CN de noms de variables ou de valeur numériques.

Programmation

Il est possible de vérifier le type de données des variables :

Syntaxe :	**ISNUM** (VAR)	
Paramètres :	VAR	Nom de la variable dont le type de données doit être vérifié.
		Le résultat de l'interrogation peut être :
	FALSE =	aucune variable numérique (type de données = STRING)
	TRUE =	variable numérique (type de données = REAL)

Syntaxe :	**ISSTR** (VAR)	
Paramètres :	VAR	Nom de la variable dont le type de données doit être vérifié.
		Le résultat de l'interrogation peut être :
	FALSE =	variable numérique (type de données = REAL)
	TRUE =	aucune variable numérique (type de données = STRING)
Exemple :		

```
IF ISNUM(VAR1) == TRUE
IF ISSTR(REG[4]+2) == TRUE
```

Il est possible de modifier le mode d'affichage des variables :

- Pour le type INTEGER, le type d'affichage peut être modifié.

B	binaire
D	décimal à signe
H	hexadécimal
sans signe	
avec un U pour unsigned	

- Pour le type Real, seul le nombre de chiffre après la virgule peut être modifié.
  Une modification du type n'est pas autorisée et entraîne la consignation d'un message d'erreur dans le fichier "easyscreen_log.txt".
  Exemple :
  ```
  Var1.typ = "IBW"
  Var2.typ = "R3"
  ```

Formats de nombre

Les nombres peuvent être représentés sous forme binaire, décimale, hexadécimale ou exponentielle :

binaire	B01110110
décimal	123,45
hexadécimal	HF1A9
exponentiel	-1.23EX-3
Exemples :	
	`VAR1 = HF1A9`
	`REG[0]= B01110110`
	`DEF VAR7 = (R//-1.23EX-3)`

Remarque

Lors de la génération de code par la fonction "GC", seules les valeurs numériques décimales ou exponentielles sont prises en compte et **non** les valeurs binaires et hexadécimales.

Voir aussi

Paramètres de variables (Page 93)

7.12 Détails relatifs au champ Toggle

Description

Avec les extensions du champ bascule, des textes (entrées dans le champ bascule) peuvent être affichés en fonction des variables CN/AP. Une variable qui utilise une extension de champ bascule, ne peut être que lue. Appuyer sur la touche INSERT permet de développer la liste du champ bascule.

Programmation

Syntaxe :	**DEF VAR1=(IB/+ $85000/15///"DB90.DBB5")** ou **DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run")**	
Description :	En affichant la boîte de dialogue, le contenu du numéro de texte $85015 est affiché dans le champ de saisie et de visualisation. La valeur par défaut 15 est renseignée dans les variables système DB90.DBB5. Si la valeur de la variable système DB90.DBB5 change, le numéro de texte affiché est reconstitué à chaque modification $(85000 + <DB90.DBB5>).	
Paramètres :	Type de variable	Type de la variable spécifiée dans variable système ou utilisateur
	Numéro de texte	Numéro (base) du texte localisé qui est valable en tant que numéro de base
	Variable système ou utilisateur	Variable système ou utilisateur (offset), à l'aide de laquelle le numéro de texte définitif (base + offset) est constitué.

Images dépendant du champ bascule

Le champ bascule affiche alternativement des deux graphiques : Si l'octet de mémento possède la valeur 1, "image1.png" s'affiche. Si l'octet de mémento possède la valeur 2, "image2.png" s'affiche.

```
DEF VAR1=(IDB/*1="\\image1.png", 2="\\image2.png"//,$85000/wr1//"MB[130]"//160,40,50,50)
```

La position et la taille de l'image sont indiquées sous "Position champ de saisie et de visualisation (gauche, haut, largeur, hauteur)"

Touche de basculement virtuelle

Pour les champs bascules sans liste configurée, p. ex. `DEF NoTglList=(R/*)`, il n'existe aucune liste. Le prochain élément n'est généré que lorsque la touche de basculement est actionnée ou que la méthode CHANGE de la variable correspondante est exécutée.

Dans ce cas, pour l'utilisation d'un écran tactile, un petit clavier virtuel comportant uniquement la touche de basculement s'affiche à droite du champ bascule.

L'affichage de la touche virtuelle de basculement peut également être obtenu indépendamment d'un écran tactile avec l'entrée suivante dans le fichier de configuration "slguiconfig.ini".

```
[VirtualKeyboard]
; Force la fonction touche de basculement virtuelle easyscreen
ForceEasyscreenVirtualToggleKey = true
```

Voir aussi

Paramètres de variables (Page 93)

7.13 Détails relatifs aux valeurs par défaut

Vue d'ensemble

En fonction de l'affectation du champ de variable (champ d'affichage / de saisie ou champ bascule) qui peut être une valeur par défaut, une variable système/utilisateur, ou les deux, différents états de la variable peuvent être ainsi obtenus (non calculée : le basculement n'est possible que si une valeur valide a été affectée à la variable).

Incidence des valeurs par défaut

Condition			**Réaction du type de champ**
Type de champ	**Valeur par défaut**	**Variable système ou utilisateur**	
Champ d'E/S	Oui	Oui	Ecriture de la valeur par défaut dans la variable système ou utilisateur
	Non	Oui	Utiliser la variable système ou utilisateur comme valeur par défaut
	Erreur	Oui	Non calculée, la variable système ou utilisateur n'est pas écrite/utilisée
	Oui	Non	Valeur par défaut
	Non	Non	Non calculée
	Erreur	Non	Non calculée
	Oui	Erreur	Non calculée
	Non	Erreur	Non calculée
	Erreur	Erreur	Non calculée
Bascule	Oui	Oui	Ecriture de la valeur par défaut dans la variable système ou utilisateur
	Non	Oui	Utiliser la variable système ou utilisateur comme valeur par défaut
	Erreur	Oui	Non calculée, la variable système ou utilisateur n'est pas écrite/utilisée
	Oui	Non	Valeur par défaut
	Non	Non	Valeur par défaut = le premier élément du champ bascule
	Erreur	Non	Non calculée
	Oui	Erreur	Non calculée
	Non	Erreur	Non calculée
	Erreur	Erreur	Non calculée

Voir aussi

Paramètres de variables (Page 93)

7.14 Détails concernant la position du texte court, la position du champ de saisie et de visualisation

Vue d'ensemble

Le texte court et le texte graphique ainsi que le champ de saisie et de visualisation et le texte d'unité forment chacun une unité. Ainsi, les indications de position pour le texte court s'appliquent également au texte graphique et aux indications pour le champ de saisie et de visualisation et au texte d'unité.

Programmation

L'indication de position configurée écrase la valeur par défaut, c'est-à-dire qu'il ne peut y avoir modification que d'une seule valeur. Si aucune indication de position n'est configurée pour les éléments de dialogue suivants, les indications de l'élément précédent sont reprises.

Si les positions ne sont indiquées pour aucun élément de dialogue, les valeurs par défaut sont utilisées. Par défaut, la largeur de colonne pour le texte court et le champ de saisie et de visualisation est définie pour chaque ligne à partir du nombre de colonnes et de la largeur de ligne maximale, ainsi la largeur de colonne = largeur de ligne maximale/nombre de colonnes.

La largeur du texte graphique et d'unité est fixe et optimisée pour les demandes de la gestion de programmation. Lorsque le texte de graphique ou d'unité a été configuré, la largeur du texte court ou du champ de saisie et de visualisation est conformément réduite.

L'ordre du texte court et du champ de saisie et de visualisation peut être interverti en indiquant la position.

Ecart entre le champ de saisie/visualisation et le champ d'unité et largeur du champ d'unité

Il est possible de configurer l'écart entre un champ de saisie/visualisation et un champ d'unité ainsi que la largeur du champ d'unité.

Pour cela, dans la ligne de définition de la section dédiée à la position de saisie/visualisation, saisir l'écart entre le champ de saisie/visualisation et le champ d'unité (p. ex. 7 pixels) et/ou la largeur du champ d'unité (p. ex. 60 pixels), séparés par une virgule :

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

Ou :

```
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}
```

Dans ce cas (quand l'écart entre champ de saisie/visualisation et champ d'unité et/ou la largeur du champ d'unité est configuré), les points suivants sont à prendre en compte :

- La largeur configurée pour le champ de saisie/visualisation ne tient **pas** compte de la largeur fixe du champ d'unité (valeur fixe : 50 pixels). En d'autres termes, seule la largeur du champ de saisie/visualisation doit être configurée.
- Si aucune largeur n'est configurée pour le champ d'unité, la largeur de 50 pixels s'applique par défaut.
- Si aucun écart n'est configuré entre le champ de saisie/visualisation et le champ d'unité, l'écart de 0 pixel s'applique par défaut.

Particularité en cas de champ bascule associé

Condition :

- un champ bascule associé est configuré pour une variable,
- un écart est configuré entre le champ de saisie/visualisation et le champ d'unité et/ou une largeur de champ d'unité est configurée pour la variable et
- la variable ne possède pas de texte d'unité.

Dans ce cas, le champ bascule associé est placé à la position configurée pour le champ d'unité de la variable.

La position éventuellement configurée pour la partie saisie/visualisation du champ bascule associé est ainsi ignorée.

Exemple

Dans l'exemple ci-dessous, le champ bascule associé **F_Unit** est positionné automatiquement avec un écart de **7 pixels** par rapport au champ de saisie/visualisation de la variable **VarF**, et défini avec une largeur de **59 pixels**.

```
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)
```

Voir aussi

Paramètres de variables (Page 93)

7.15 Utilisation de chaînes de caractères

Chaînes

Lors de la configuration, il est possible d'utiliser des chaînes (strings) afin d'élaborer l'affichage de texte de façon dynamique ou de concaténer différents textes pour la génération de code.

Règles

Lors de l'utilisation de variables de chaîne, les règles suivantes doivent être respectées :

- Les combinaisons sont traitées de gauche à droite.
- Les expressions imbriquées sont résolues de l'intérieur vers l'extérieur.
- L'emploi des majuscules et minuscules n'a pas d'importance.
- Des variables de string sont généralement affichées de manière justifiée à gauche.

Les chaînes peuvent être supprimées en renvoyant à une chaîne vide.

Les chaînes peuvent être ajoutées à droite du signe d'égalité par l'opérateur "<<". Des guillemets (") dans une chaîne sont caractérisés par deux guillemets successifs. L'égalité des chaînes peut être vérifiée dans les instructions IF.

Exemple

Réglage par défaut pour les exemples suivants :

```
VAR1.VAL = "Ceci est un"

VAR8.VAL = 4

VAR14.VAL = 15

VAR2.VAL = "défaut"

$85001 = "Ceci est un"

$85002 = "texte d'alarme"
```

Edition de chaînes :

- Concaténation de chaînes :
  ```
  VAR12.VAL = VAR1 << " défaut."   ;résultat : "Ceci est un défaut"
  ```
- Suppression d'une variable :
  ```
  VAR10.VAL = ""   ;Résultat : Chaîne vide
  ```
- Placer une variable avec une variable de texte :
  ```
  VAR11.VAL = VAR1.VAL   ;Résultat : "Ceci est un"
  ```
- Adaptation de type de données :
  ```
  VAR13.VAL ="Ceci est le " << (VAR14 - VAR8) << ". défaut"
              ;Résultat : "Ceci est le 11. défaut"
  ```
- Traitement des valeurs numériques :
  ```
  VAR13.VAL = "Défaut " << VAR14.VAL << " : " << $85001 << $85002
           ;Résultat : "Défaut 15 : Ceci est un texte d'alarme"
  IF VAR15 == "Défaut"   ;Chaînes d'instruction IF
    VAR16 = 18.1234
           ;Résultat : VAR16 égale à 18.1234,
           ;si VAR15 est égal à "Défaut".
  ENDIF
  ```

- Guillemets dans une chaîne :
```
VAR2="Bonjour ceci est un "Test""
   ;Résultat : Bonjour ceci est un "Test"
```
- Chaînes de variable système ou utilisateur dépendant des contenus de variable :
```
VAR2.Var = "$R[" << VAR8 << "]"   ;Résultat : $R[4]
```

Voir aussi

Fonctions STRING (Page 182)

7.16 Variable CURPOS

Description

La variable CURPOS permet d'interroger ou de modifier la position du curseur dans la zone de saisie active de la boîte de dialogue actuelle. La variable indique combien de caractères sont situés avant le curseur. Si le curseur est placé au début du champ de saisie, CURPOS affiche la valeur 0. Si on modifie la valeur de CURPOS, le curseur est placé dans le champ de saisie à l'emplacement correspondant.

Afin de pouvoir réagir aux modifications de valeur de variable, il est possible de surveiller les changements à l'aide d'une méthode CHANGE. Si la valeur de CURPOS change, la méthode CHANGE est démarrée et les instructions contenues sont exécutées.

7.17 Variable CURVER

Description

La caractéristique CURVER (Current Version) permet d'adapter la programmation pour traiter les différentes versions. La variable CURVER n'est accessible qu'en lecture seule.

Remarque

Lors de la génération de code, il est automatiquement généré avec la version la plus récente même s'il a été auparavant décompilée avec une version plus ancienne. La commande "GC" génère toujours la version la plus récente. Dans le code généré, un identifiant supplémentaire de la version générée est ajoutée dans le commentaire utile pour les versions > 0.

Règles

C'est toujours le dialogue le plus récent avec toutes ses variables qui est affiché.

- Les variables existantes ne doivent pas être modifiées.
- Les nouvelles variables sont ajoutées dans la programmation (de cycle) existante dans l'ordre souhaité.

- Il n'est pas possible de retirer des variables d'un dialogue d'une version à la suivante.
- Le dialogue doit contenir toutes les variables de toutes les versions.

Exemple

```
(IF CURVER==1 ...)                ; CURVER reçoit automatiquement la version du co-
                                    de décompilé en cas de décompilation.
```

7.18 Variable ENTRY

Description

La variable ENTRY permet de vérifier comment la boîte de dialogue a été appelée.

Programmation

Syntaxe :	**ENTRY**	
Description :	La variable ENTRY n'est accessible qu'en lecture.	
Valeur retournée :		Le résultat de l'interrogation peut être :
	0 =	aucune aide à la programmation
	1 =	Lancement d'un masque à l'aide d'une touche logicielle ; aucun code n'a été généré à ce stade (préréglage, comme configuré)
	2 =	Lancement d'un masque à l'aide d'une touche logicielle ; un code est généré (préréglage à partir du dernier code généré par ce masque)
	3 =	Décompilation avec commentaire d'utilisation (lignes #)
	4 =	Code_typ = 0 : Code CN avec commentaire d'utilisation (lignes #)
	5 =	Code_typ = 1 : Code CN sans commentaire d'utilisation (lignes #)

Exemple

```
IF ENTRY == 0
  DLGL("La boîte de dialogue n'a pas été appelée par programmation")
ELSE
  DLGL("La boîte de dialogue a été appelée par programmation")
ENDIF
```

7.19 Variable ERR

Description

La variable ERR permet de vérifier si la ligne précédente respective a été exécutée sans erreur.

Programmation

Syntaxe :	**ERR**	
Description :	La variable ERR n'est accessible qu'en lecture.	
Valeur retournée :		Le résultat de l'interrogation peut être :
	FALSE =	la ligne précédente a été exécutée sans erreur
	TRUE =	la ligne précédente n'a pas été exécutée sans erreur

Exemple

```
VAR4 = Filet[VAR1,"KDM",3]              : Lecture de la valeur dans le tableau
IF ERR == TRUE                          ; Interrogation si la valeur a été trouvée
                                        dans le tableau
VAR5 = "Erreur lors de l'accès au ta-
bleau"
                                        ; Si la valeur n'a pas été trouvée dans le
                                        tableau, la valeur "Erreur lors de l'accès
                                        au tableau" est affectée à la variable.
ELSE
  VAR5 = "Tout OK"                      ; Si la valeur a été trouvée dans le tableau,
                                        la valeur "Tout OK" est affectée à la varia-
                                        ble.
ENDIF
```

7.20 Variable FILE_ERR

Description

La variable FILE_ERR permet de vérifier si la commande GC ou CP précédente a été exécutée sans erreur.

Programmation

Syntaxe :	**FILE_ERR**
Description :	La variable FILE_ERR peut uniquement être lue.

Valeur de retour :		Les résultats possibles sont :
	0 =	Opération en ordre
	1 =	Lecteur/chemin inexistant
	2 =	Erreur d'accès au chemin / au fichier
	3 =	Lecteur pas prêt
	4 =	Nom de fichier erroné
	5 =	Fichier déjà ouvert
	6 =	Accès refusé
	7 =	Chemin de destination inexistant ou non autorisé
	8 =	La source de la copie correspond à la destination
	10 =	Erreur interne : Avec FILE_ERR = 10, il s'agit d'une erreur ne pouvant pas être classée dans les autres catégories.

Exemple

```
CP("D:\source.mpf","E:\target.mpf")
                                                ; Copie de source.mpf vers E:\tar-
                                                get.mpf
IF FILE_ERR > 0                                 ; Interrogation si une erreur est sur-
                                                venue
IF FILE_ERR == 1                                ; Interrogation de numéros d'erreur
                                                particuliers et affichage du texte
                                                d'erreur correspondant
VAR5 = "Lecteur/chemin inexistant"
ELSE
IF FILE_ERR == 2
VAR5 = "Erreur d'accès au chemin / au fichier"
ELSE
IF FILE_ERR == 3
            VAR5 = "Nom de fichier incorrect"
         ENDIF
ENDIF
   ENDIF
ELSE
   VAR5 = "Tout OK"                             ; Si aucune erreur n'est survenue
                                                dans CP (ou GC), affichage de "Tout
                                                OK"
ENDIF
```

7.21 Variable FOC

Description

La variable FOC permet de piloter le pointeur de saisie (champ de saisie et de visualisation actuel actif) dans une boîte de dialogue. La réaction du curseur à gauche, à droite, en haut, en bas, ainsi que PGUP, PGDN sont prédéfinis.

Remarque

La commande FOC ne doit pas être déclenchée par un événement de navigation. La position du curseur ne doit être modifiée que dans les méthodes PRESS de touche logicielle, méthodes CHANGE, etc.

Les variables avec le mode d'entrée wr = 0 et wr = 4 et les variables auxiliaires ne peuvent pas recevoir le focus.

Programmation

Syntaxe :	**FOC**	
Description :	La variable peut être lue et écrite.	
Valeur retournée :	Lecture	Le nom de la variable ayant le focus est fourni en tant que résultat.
	Ecriture	Une chaîne ou une valeur numérique peut être affectée. Une chaîne est interprétée comme nom de variable et une valeur numérique comme index de variable.

Exemple

```
IF FOC == "Var1"              ; Lecture du pointeur de saisie (focus)
  REG[1] = Var1
ELSE
  REG[1] = Var2
ENDIF

FOC = "Var1"                  ; Le pointeur de saisie est affecté à la variable 1.
FOC = 3                       ; Le pointeur de saisie est affecté au 3ème élément de
                              la boîte de dialogue avec WR ≥ 2.
```

Voir aussi

FOCUS (Page 125)

7.22 Variable S_ALEVEL

Description

La caractéristique de masque S_ALEVEL permet d'interroger le niveau d'accès actuel dans la configuration.

Programmation

Syntaxe :	**S_ALEVEL**
Description :	Interrogation du niveau d'accès actuel
Valeur retournée :	0: Système
	1: Constructeur
	2: Maintenance
	3: Utilisateur
	4: Commutateur à clé 3
	5: Commutateur à clé 2
	6: Commutateur à clé 1
	7: Commutateur à clé 0

Exemple

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

Voir aussi

ACCESSLEVEL (Page 122)

7.23 Variable S_CHAN

Description

Avec la variable S_CHAN, le numéro du canal courant peut être transmis pour l'affichage ou pour une évaluation.

Programmation

Syntaxe :	**S_CHAN**
Description :	Interrogation du numéro de canal actuel
Valeur retournée :	Numéro de canal

Exemple

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

Voir aussi

CHANNEL (Page 124)

7.24 Variable S_CONTROL

Description

La caractéristique de masque S_CONTROL permet d'interroger le nom de commande actuel dans la configuration.

Programmation

Syntaxe :	**S_CONTROL**
Description :	Interrogation du nom de commande actuel
Valeur retournée :	Le nom de commande est le nom de section dans "mmc.ini"

Exemple

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

Voir aussi

CONTROL (Page 124)

7.25 Variable S_LANG

Description

La caractéristique de masque S_LANG permet d'interroger la langue actuelle dans la configuration.

Programmation

Syntaxe :	**S_LANG**
Description :	Interrogation de la langue actuelle
Valeur retournée :	Code langue issu de resources.xml p. ex. "deu", "eng", "fra", etc.

Exemple

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

Voir aussi

LANGUAGE (Page 126)

7.26 Variable S_NCCODEREADONLY

Description

La caractéristique de masque S_NCCODEREADONLY n'est prise en compte que lorsqu'un cycle avec une boîte de dialogue "Run MyScreens" est décompilé dans l'éditeur. S_NCCODEREADONLY permet de déterminer si un code CN décompilé provenant de l'éditeur peut être modifié ou non.

En ce qui concerne la valeur retournée

- TRUE : Le code CN décompilé provenant de l'éditeur peut être modifié.
- FALSE : Le code CN décompilé provenant de l'éditeur ne peut pas être modifié (readonly), p. ex. parce qu'il est déjà en prétraitement (= TRUE).

7.27 Variables S_RESX et S_RESY

Description

Les caractéristiques de masque S_RESX et S_RESY permettent d'interroger la résolution actuelle ou ses composantes X et Y dans la configuration.

Exemple

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

Voir aussi

RESOLUTION (Page 130)

Commandes de programmation 8

8.1 Opérateurs

Vue d'ensemble

En programmation, il est possible d'utiliser les opérateurs suivants :

- Opérateurs mathématiques
- Opérateurs relationnels
- Opérateurs logiques (booléens)
- Opérateurs de bit
- Fonctions trigonométriques

8.1.1 Opérateurs mathématiques

Vue d'ensemble

Opérateurs mathématiques	Désignation
+	Addition
-	Soustraction
*	Multiplication
/	Division
MOD	Opération modulo
( )	Parenthèses
AND	Opérateur ET
OR	Opérateur OU
NOT	Opérateur NOT
ROUND	Arrondir les nombres à virgule

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

ROUND

L'opérateur ROUND est utilisé pour arrondir des nombres ayant jusqu'à 12 chiffres après la virgule pendant l'exécution d'une boîte de dialogue. Les chiffres après la virgule ne peuvent pas être repris par les champs de variables dans l'affichage.

Utilisation

ROUND est piloté par l'utilisateur par deux paramètres :

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

Résultat : `VAR2 = 5,2339`

VAR1 contient le nombre à arrondir. Le paramètre "4" indique le nombre de chiffres après la virgule dans le résultat à mémoriser dans VAR2.

Fonctions trigonométriques

Fonctions trigonométriques	Désignation
SIN(x)	Sinus de x
COS(x)	Cosinus de x
TAN(x)	Tangente de x
ATAN(x, y)	Arc tangente de x/y
SQRT(x)	Racine carrée de x
ABS(x)	Valeur absolue de x
SDEG(x)	Conversion en degré
SRAD(x)	Conversion en radian
CALC_ASIN(x)	Arc sinus de x
CALC_ACOS(x)	Arc cosinus de x

Remarque

Les fonctions fonctionnent avec des valeurs d'arc. Pour la conversion, les fonctions SDEG() et SRAD() peuvent être utilisées.

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

Fonctions mathématiques

Fonctions mathématiques	Désignation
CALC_CEIL(x)	Détermine le prochain nombre entier supérieur de x (arrondi à l'unité supérieure)
CALC_FLOOR(x)	Détermine le prochain nombre entier inférieur de x (arrondi à l'unité inférieure)
CALC_LOG(x)	Détermine le logarithme (naturel) pour la base e de x
CALC_LOG10(x)	Détermine le logarithme pour la base 10 de x
CALC_POW(x, y)	Détermine x puissance y (x à la puissance y)
CALC_MIN(x, y)	Détermine le plus petit nombre entre x et y
CALC_MAX(x, y)	Détermine le plus petit nombre entre x et y

Fonction Random : nombre aléatoire

Syntaxe :	**RANDOM** (valeur limite inférieure, valeur limite supérieure)	
Description :	La fonction **RANDOM** fournit un nombre pseudo-aléatoire dans une plage donnée.	
Paramètres :	Valeur limite inférieure	Valeur limite inférieure >= -32767, valeur limite inférieure < valeur limite supérieure
	Valeur limite supérieure	Valeur limite supérieure <= 32767

Exemple

```
REG[0] = RANDOM(-10,10)         ; résultat possible = -3
```

Constantes

Constantes	
PI	3.14159265358979323846
FALSE	0
TRUE	1

Exemple : `VAR1.VAL = PI`

Opérateurs relationnels

Opérateurs relationnels	
==	égal à
<>	différent de
>	supérieur à
<	inférieur à
>=	supérieur ou égal à
<=	inférieur ou égal à

Exemple :

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

Conditions

La profondeur d'imbrication est illimitée.

Condition avec une instruction :

```
IF
...
ENDIF
```

Condition avec deux instructions :

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

8.1.2 Opérateurs de bit

Vue d'ensemble

Opérateurs de bit	Désignation
BOR	OU bit à bit
BXOR	XOR bit à bit
BAND	ET bit à bit
BNOT	NI bit à bit
SHL	Décalage bits à gauche
SHR	Décalage bits à droite

Opérateur SHL

L'opérateur SHL (SHIFT LEFT) permet de décaler les bits vers la gauche. Il est possible d'indiquer la valeur à décaler et le nombre de pas de décalage directement ou sous forme de variable. Lorsque la limite du format de données est atteinte, les bits sont décalés au-delà sans message d'erreur.

Utilisation

Syntaxe :	variable = valeur **SHL**nombre de pas	
Description :	Décalage vers la gauche	
Paramètres :	valeur	valeur à décaler
	nombre de pas	nombre de pas de décalage

Exemple

```
PRESS(VS1)
```

```
  VAR01 = 16 SHL 2           ; Résultat = 64
  VAR02 = VAR02 SHL VAR04    ; Le contenu de VAR02 est converti en 32 bits sans signe
                             et décalé à gauche de la valeur de VAR04 bits. Puis, la
                             valeur de 32 bits est de nouveau convertie au format de
                             la variable VAR02.
END_PRESS
```

Opérateur SHR

L'opérateur SHR (SHIFT RIGHT) permet de décaler les bits vers la droite. Il est possible d'indiquer la valeur à décaler et le nombre de pas de décalage directement ou sous forme de variable. Lorsque la limite du format de données est atteinte, les bits sont décalés au-delà sans message d'erreur.

Utilisation

Syntaxe :	variable = valeur **SHR**nombre de pas		
Description :	Décalage vers la droite		
Paramètres :	valeur	valeur à décaler	
	nombre de pas	nombre de pas de décalage	

Exemple

```
PRESS(VS1)
  VAR01 = 16 SHR 2           ; Résultat = 4
  VAR02 = VAR02 SHR VAR04    ; Le contenu de VAR02 est converti en 32 bits sans
                             signe et décalé à droite de la valeur de VAR04
                             bits. Puis, la valeur de 32 bits est de nouveau
                             convertie au format de la variable VAR02.
END_PRESS
```

8.2 Méthodes

Aperçu

Dans les boîtes de dialogue et dans les barres de touches logicielles dépendant des boîtes de dialogue (barres de touches logicielles appelées par une boîte de dialogue nouvellement configurée), des actions définies peuvent être déclenchées par différents événements (quitter le champ de saisie, activation de touches logicielles). Ces actions sont configurées dans des méthodes.

La programmation de base d'une méthode se présente de la façon suivante :

Bloc de description	Commentaire	Renvoi au chapitre
`PRESS(HS1)`	;Identifiant de début de la méthode	
`LM...` `  LS...`	;Fonctions	voir chapitre Fonctions (Page 134)
`Var1.st = ...`	;Modification des propriétés	voir chapitre Définition de la barre de touches logicielles (Page 66) et chapitre Définition des propriétés de la boîte de dialogue (Page 54)

`Var2 = Var3 + Var4`
`...`
`EXIT`
	;Calcul avec des variables	voir chapitre Définition des variables (Page 85)

`END_PRESS`	;Identifiant de fin de la méthode	

8.2.1 ACCESSLEVEL

Description

La méthode ACCESSLEVEL est exécutée lorsque le niveau d'accès actuel a changé pour un masque ouvert.

Programmation

Syntaxe :	ACCESSLEVEL <Instructions> END_ACCESSLEVEL
Description :	Niveau d'accès
Paramètres :	- aucun -

Voir aussi

Variable S_ALEVEL (Page 113)

8.2.2 CHANGE

Description

Les méthodes CHANGE sont utilisées lorsqu'une valeur de variable est modifiée. Ainsi, les calculs de variable qui sont utilisés immédiatement lors de la modification de variable sont configurés au sein d'une méthode CHANGE.

On distingue la méthode CHANGE globale et spécifique à l'élément :

- La **méthode CHANGE spécifique à l'élément** est utilisée lorsque la valeur des variables spécifiques change. Si une variable système ou utilisateur est affectée à une variable, la valeur de variable est régulièrement actualisée dans une méthode CHANGE.
- La **méthode CHANGE globale** est utilisée lorsque la valeur d'une variable quelconque change et qu'aucune méthode CHANGE spécifique à l'élément n'est configurée.

Programmation "spécifique à l'élément"

Syntaxe :	CHANGE(Descripteur) ... END_CHANGE	
Description :	Modification de la valeur des variables spécifiées	
Paramètres :	Descripteur	Nom de la variable

Exemple

```
DEF VAR1=(I//////"DB20.DBB1")        ; Une variable système est affectée à Var1
CHANGE(VAR1)
  IF VAR1.Val <> 1
  VAR1.st="Outil OK !"               ; Si la valeur des variables système est ≠ 1,
                                     le texte court de la variable est le suivant :
                                     Outil OK !
    otto=1
  ELSE
    VAR1.st="Attention erreur !"     ; Si la valeur des variables système est = 1,
                                     le texte court de la variable est le suivant :
                                     Attention erreur !
    otto=2
  ENDIF
  VAR2.Var=2
END_CHANGE
```

Programmation "globale"

Syntaxe :	CHANGE() ... END_CHANGE
Description :	Modification d'une valeur de variable quelconque
Paramètres :	- aucun -

Exemple

```
CHANGE()
  EXIT                                ; Si une valeur de variable quelconque est mo-
                                      difiée, la boîte de dialogue se ferme.
END_CHANGE
```

8.2.3 CHANNEL

Description

La méthode CHANNEL est exécutée lorsque le canal actuel a changé pour un masque ouvert, c'est-à-dire lorsqu'une commutation de canaux a eu lieu.

Programmation

Syntaxe :	CHANNEL <Instructions> END_CHANNEL
Description :	Commutation entre canaux
Paramètres :	- aucun -

Voir aussi

Variable S_CHAN (Page 113)

8.2.4 CONTROL

Description

La méthode CONTROL est exécutée lorsque la commande actuelle a changé pour un masque ouvert, c'est-à-dire typiquement lors d'une commutation 1:n.

Programmation

Syntaxe :	CONTROL <Instructions> END_CONTROL
Description :	Commutation de commande
Paramètres :	- aucun -

Voir aussi

Variable S_CONTROL (Page 114)

8.2.5 FOCUS

Description

La méthode FOCUS est utilisée si le focus (curseur) est placé dans un autre champ de la boîte de dialogue.

La méthode FOCUS ne doit pas être déclenchée par un événement de navigation. La position du curseur ne doit être modifiée que dans les méthodes PRESS, CHANGE, etc. des touches logicielles. La réaction des mouvements du curseur est prédéfinie.

Remarque

Au sein de la méthode FOCUS, le positionnement sur une autre variable et le chargement d'une nouvelle boîte de dialogue sont proscrits.

Programmation

Syntaxe :	FOCUS ... END_FOCUS
Description :	Positionnement du curseur
Paramètres :	- aucun -

Exemple

```
FOCUS
  DLGL("Le focus a été placé sur la variable"<< FOC <<.)
END_FOCUS
```

Voir aussi

Variable FOC (Page 112)

8.2.6 LANGUAGE

Description

La méthode LANGUAGE est exécutée lorsque la langue actuelle a changé pour un masque ouvert.

Programmation

Syntaxe :	LANGUAGE <Instructions> END_LANGUAGE
Description :	Langue
Paramètres :	- aucun -

Voir aussi

Variable S_LANG (Page 114)

8.2.7 LOAD

Description

La méthode LOAD est utilisée lorsque les définitions de variables et de touches logicielles ont été interprétées (DEF Var1= ..., HS1= ...). La boîte de dialogue ne s'affiche pas encore à ce moment.

Programmation

Syntaxe :	LOAD ... END_LOAD
Description :	Chargement
Paramètres :	- aucun -

Exemple

```
LOAD                          ; Identifiant de début
  Masque1.Hd =  $85111        ; Affectation d'un texte de titre de boîte de dialogue à
                                partir du fichier de langue
  VAR1.Min = 0                ; Affectation de la valeur limite MIN à la variable
  VAR1.Max = 1000             ; Affectation de la valeur limite MAX à la variable
END_LOAD                      ; Identifiant de fin
```

Voir aussi

Définition des éléments graphiques (Page 193)

8.2.8 UNLOAD

Description

La méthode UNLOAD est utilisée avant qu'une boîte de dialogue ne soit déchargée.

Programmation

Syntaxe :	UNLOAD ... END_UNLOAD
Description :	Déchargement
Paramètres :	- aucun -

Exemple

```
UNLOAD
  REG[1] = VAR1           ; Archivage de la variable dans le registre
END_UNLOAD
```

8.2.9 OUTPUT

Description

La méthode OUTPUT est utilisée lorsque la fonction "CG" est appelée. Au sein d'une méthode OUTPUT, les variables et les variables d'aide sont configurées comme code CN. L'enchaînement des différents éléments d'une ligne de code est effectué avec un espace.

Remarque

Le code CN peut être généré avec les fonctions de fichier dans un fichier à part et déplacé dans la CN.

Programmation

Syntaxe :	OUTPUT(Descripteur) ... END_OUTPUT	
Description :	Prendre des variables dans le programme CN	
Paramètres :	Descripteur	Nom de la méthode OUTPUT

Numéros de bloc et identifiants de masquage

La méthode OUTPUT ne doit pas contenir de numéros de lignes ni d'identifiants de masquage si les numéros de lignes directement spécifiés à l'aide de la gestion de programmation dans le programme pièce et les identifiants de masquage doivent être conservés lors de décompilations.

Les modifications avec l'éditeur dans le programme pièce entraînent le comportement suivant :

Condition	Comportement
Le nombre de blocs ne change pas.	Les numéros de bloc sont conservés.
Le nombre de blocs diminue.	Les plus grands numéros de bloc sont barrés.
Le nombre de blocs augmente.	Les nouveaux blocs ne deviennent pas des numéros de bloc.

Exemple

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

8.2.10 PRESS

Description

La méthode PRESS est utilisée lorsque la touche logicielle correspondante est pressée.

Programmation

Syntaxe :	PRESS(Touche logicielle) ... END_PRESS
Désignation :	Actionnement d'une touche logicielle

Paramètres :	Touche logicielle	Nom de la touche logicielle: HS1 - HS8 et VS1 - VS8	
	RECALL	Touche <RECALL>	
	ENTER	Touche <ENTER>, voir PRESS(ENTER) (Page 129)	
	TOGGLE	Touche <TOGGLE>, voir PRESS(TOGGLE) (Page 130)	
	PU	Page préc	Image en haut
	PD	Page suiv	Image en bas
	SL	Défilement vers la gauche	Curseur vers la gauche
	SR	Défilement vers la droite	Curseur vers la droite
	SU	Défilement vers le haut	Curseur vers le haut
	SD	Défilement vers le bas	Curseur vers le bas

Exemple

```
HS1 = ("autre barre de touches
logicielles")
HS2=("aucune fonction")
PRESS(HS1)
  LS("Barre1")                    ; Chargement d'une autre barre de touches logiciel-
                                  les
  Var2 = Var3 + Var1
END_PRESS
PRESS (HS2)
END_PRESS
PRESS(PU)
  INDEX = INDEX -7
  CALL("UP1")
END_PRESS
```

8.2.11 PRESS(ENTER)

Description

La méthode PRESS(ENTER) est toujours appelée lorsque la touche Entrée est enfoncée pour une variable avec un champ de saisie et de visualisation et le mode de saisie WR3 ou WR5 :

- WR3 : Navigation sur le champ et actionnement de la touche Entrée
- WR5 : En mode de saisie, validation de la valeur avec la touche Entrée

Programmation

Syntaxe :	PRESS(ENTER) <Instructions> END_PRESS
Description :	Touche Entrée enfoncée
Paramètres :	- aucun -

8.2.12 PRESS(TOGGLE)

Description

La méthode PRESS(TOGGLE) est toujours appelée lorsque la touche de basculement est enfoncée, indépendamment de la variable sur laquelle se trouve actuellement le focus.

Si nécessaire, il est possible de déterminer sur quelle variable se trouve actuellement le focus à l'aide de la propriété de masque FOC.

Programmation

Syntaxe :	PRESS(TOGGLE) <Instructions> END_PRESS
Description :	Touche de basculement enfoncée
Paramètres :	- aucun -

Exemple

```
PRESS(TOGGLE)
   DLGL("Toggle key pressed at variable " << FOC)        ; la propriété de masque FOC permet de déter-
                                                          miner sur quelle variable se trouve actuelle-
                                                          ment le focus

END_PRESS
```

8.2.13 RESOLUTION

Description

La méthode RESOLUTION est exécutée lorsque la résolution actuelle a changé pour un masque ouvert, c'est-à-dire typiquement lors d'une commutation de TCU.

Programmation

Syntaxe :	RESOLUTION <Instructions> END_RESOLUTION
Description :	Résolution
Paramètres :	- aucun -

Voir aussi

Variables S_RESX et S_RESY (Page 115)

8.2.14 RESUME

Description

La méthode RESUME est appelée lorsque le masque est interrompu, p. ex. par un changement de zone, puis réaffiché. Les changements de valeur sont alors de nouveau traités, le cas échéant, la temporisation et le bloc RESUME sont exécutés.

Remarque

Les fonctions LS, LM, DLGL, EXIT et EXITLS ne doivent pas être configurées dans les méthodes SUSPEND ou RESUME. L'utilisation des fonctions dans ces méthodes empêchera l'exécution des fonctions.

Programmation

Syntaxe :	RESUME <Instruction> END_RESUME
Description :	Masque de nouveau actif
Paramètres :	- aucun -

8.2.15 SUSPEND

Description

La méthode SUSPEND est appelée lorsque le masque est interrompu, mais pas déchargé. Ceci est par exemple le cas lorsque le masque est quitté par un simple changement de zone, mais n'est pas explicitement déchargé. Le masque est alors conservé en arrière-plan mais aucun changement de valeur ou aucune temporisation n'est traité. Le masque est en pause.

Remarque

Les fonctions LS, LM, DLGL, EXIT et EXITLS ne doivent pas être configurées dans les méthodes SUSPEND ou RESUME. L'utilisation des fonctions dans ces méthodes empêchera l'exécution des fonctions.

Programmation

Syntaxe :	SUSPEND <Instruction> END_SUSPEND
Description :	Interruption du masque
Paramètres :	- aucun -

Exemple

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

8.2.16 Exemple : gestion de version avec les méthodes OUTPUT

Vue d'ensemble

Les boîtes de dialogue existantes peuvent être complétées par des variables supplémentaires dans le cadre des extensions. La version est indiquée entre parenthèses dans les définitions des variables supplémentaires après le nom de la variable : (0 = d'origine, n'est pas mentionné), 1 = version 1, 2 = version 2, ...

Exemple :

```
DEF var100=(R//1)              ; Version originale, correspond à la version 0
DEF var101(1)=(S//"Bonjour")   ; Extension à partir de la version 1
```

Lors de la rédaction de la méthode OUTPUT, il est possible de se référer à une version spécifique, par rapport à la totalité des définitions.

Exemple :

```
OUTPUT(NC1)                    ; Seules les variables de la version originale sont
                               proposées dans la méthode OUTPUT.
OUTPUT(NC1,1)                  ; Les variables de la version originale et les ex-
                               tensions avec l'identifiant de version 1 sont propo-
                               sées dans la méthode OUTPUT.
```

La méthode OUTPUT pour la version originale n'a pas besoin d'identifiant de version, il est cependant possible d'écrire 0. OUTPUT(NC1) correspond à OUTPUT(NC1,0). L'identifiant de version n dans la méthode OUTPUT permet d'englober toutes les variables des versions 0, 1, 2... jusqu'à n.

Programmation avec identificateur de version

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

OUTPUT(NC1)
var100",,"var101
END_OUTPUT

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

OUTPUT(NC1)                        D'origine et en plus la nouvelle version
var100","var101
END_OUTPUT
```

```
...

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

8.3 Fonctions

Vue d'ensemble

Dans des boîtes de dialogue et dans les barres de touches logicielles relatives à la boîte de dialogue, il y a plusieurs fonctions qui peuvent être lancées par différents événements, par exemple quitter le champ de saisie, appuyer sur une touche logicielle, et qui sont configurées dans les méthodes.

Sous-programmes

Il est possible de configurer dans des sous-programmes des instructions de configuration qui se répètent ou non et qui regroupent une procédure particulière. Les sous-programmes peuvent à tout moment être chargés dans un programme principal ou dans un autre sous-programme puis être traités aussi souvent que souhaité ; c'est pourquoi les instructions ne doivent pas être configurées plusieurs fois. Les blocs de description des boîtes de dialogue ou des barres de touches logicielles sont considérés comme un programme principal.

Fonctions externes

A l'aide des fonctions externes, des fonctions supplémentaires et spécifiques à l'utilisateur peuvent être ajoutées. Les fonctions externes sont placées dans un fichier DLL et déclarées par une entrée dans les lignes de définition du fichier de configuration.

Services PI

La fonction PI_START permet de démarrer des services PI (services d'instance de programme) de l'AP dans la zone CN.

Voir aussi

Function (FCT) (Page 155)

Services PI (Page 170)

8.3.1 Lecture et écriture de paramètres d'entraînement : RDOP, WDOP, MRDOP

Description

Les fonctions RDOP, WDOP et MRDOP permettent de lire et d'écrire des paramètres d'entraînement.

Remarque

Ne pas lire les paramètres d'entraînement à un cycle plus rapide que 1 seconde, plutôt plus lentement.

Motif : Autrement la communication avec les entraînements peut être extrêmement perturbée ou des défaillances peuvent survenir.

Remarque

Si une erreur survient lors de la lecture ou de l'écriture de paramètres d'entraînement, la propriété de masque ERR est définie en conséquence.

Programmation

Syntaxe :	**RDOP**("descripteur de l'objet entraînement", "numéro de paramètre")	
Description :	Lecture d'un paramètre d'entraînement (Drive Object Parameter)	
Paramètres :	Descripteur de l'objet entraînement	Le descripteur de l'objet entraînement peut être récupéré de la colonne "DO name" dans "Diagnostic du système d'entraînement" dans le groupe fonctionnel Diagnostic (> ETC > TLH 8 : Drive system) (voir figure ci-dessous).
	Numéro de paramètre	

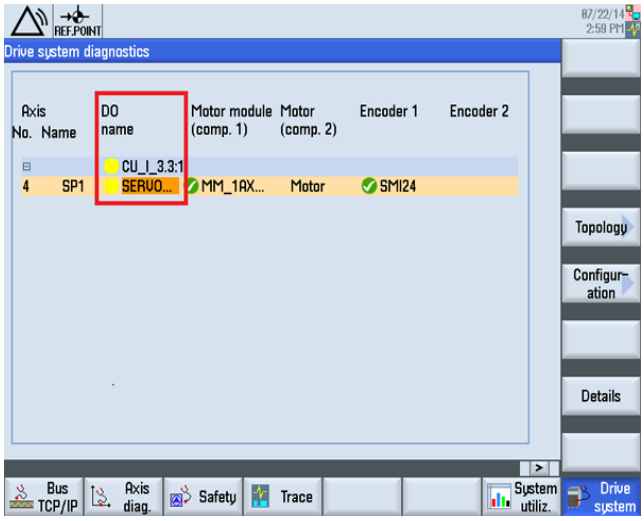


Figure 8-1 Descripteur de l'objet entraînement

Syntaxe :	**MRDOP**("descripteur de l'objet entraînement", "numéro de paramètre1"*"numéro de paramètre"[*...], indice de registre)	
Description :	Lecture multiple de paramètres d'entraînement. L'instruction MRDOP permet de transmettre plusieurs paramètres d'entraînement d'un objet entraînement avec un accès au registre. Cet accès est nettement plus rapide que la lecture par accès individuel.	
Paramètres :	Descripteur de l'objet entraînement	Le descripteur de l'objet entraînement peut par exemple être récupéré de la vue initiale du groupe fonctionnel "Mise en service".
	Numéro de paramètre 1 à n	Pour les noms de variable, "*" sert de séparateur. Les valeurs sont reprises dans le registre REG[indice de registre] dans l'ordre d'apparition des noms de variables dans l'instruction.
	Indice de registre	La valeur de la première variable se trouve dans REG[indice de registre]. La valeur de la deuxième variable se trouve dans REG[indice de registre + 1].

Syntaxe :	**WDOP**("descripteur de l'objet entraînement", "numéro de paramètre", valeur)
Description :	Ecriture d'un paramètre d'entraînement (Drive Object Parameter)

Paramètres :	Descripteur de l'objet entraînement	Le descripteur de l'objet entraînement peut par exemple être récupéré de la vue initiale du groupe fonctionnel "Mise en service".
	Numéro de paramètre	Numéro de paramètre
	Valeur	Valeur à écrire

Exemples

Lire la température moteur r0035 de l'objet entraînement "SERVO_3.3:2" :

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

Lire la température moteur r0035 et la valeur réelle de couple r0080 de l'objet entraînement "SERVO_3.3:2" et enregistrer les résultats respectifs à partir de l'indice de registre 10 :

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

8.3.2 Appel du sous-programme (CALL)

Description

La fonction CALL permet d'appeler un sous-programme chargé depuis n'importe quel emplacement d'une méthode. L'imbrication, c'est-à-dire l'appel d'un sous-programme par un sous-programme, est permise.

Programmation

Syntaxe :	**CALL**("Descripteur")	
Description :	Appel d'un sous-programme	
Paramètres :	Descripteur	Nom du sous-programme

Exemple

```
//M(MASQUE1)
DEF VAR1 = ...
DEF VAR2 = ...
CHANGE(VAR1)
  ...
  CALL("MY_UP1")              ; Appel et exécution du sous-programme
  ...
END_CHANGE

CHANGE(VAR2)
  ...
```

```
//M(MASQUE1)
  CALL("MY_UP1")              ; Appel et exécution du sous-programme
  ...
END_CHANGE

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

8.3.3 Définition de bloc (//B)

Description

Les sous-programmes sont désignés dans un fichier de programme par l'identifiant de bloc //B et se terminent par //END. Pour chaque identifiant de bloc, plusieurs sous-programmes peuvent être définis.

Remarque

Les variables utilisées dans les sous-programmes doivent être définies dans la boîte de dialogue à l'aide de laquelle le sous-programme a été appelé.

Programmation

Un bloc possède la structure suivante :

Syntaxe :	**//B**(Nom de bloc) **SUB**(Descripteur) END_SUB [SUB(Descripteur) ... END_SUB] ... //END	
Description :	Définir un sous-programme	
Paramètres :	Nom de bloc	Nom de l'identifiant de bloc
	Descripteur	Nom du sous-programme

Exemple

```
//B(PROG1)                    ; Début du bloc
```

```
SUB(UP1)                          ; Début du sous-programme
  ...
  REG[0] = 5                      ; Affectation de la valeur 5 au registre 0
  ...
END_SUB                           ; Fin du sous-programme
SUB(UP2)                          ; Début du sous-programme
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN
  ENDIF
  VAR1.val="Otto"
END_SUB                           ; Fin du sous-programme
//END                             ; Fin du bloc
```

8.3.4 Vérifier la variable (CVAR)

Description

A l'aide de la fonction CVAR (Check Variable), il est possible de demander si toutes ou certaines variables ou variables d'aide d'une boîte de dialogue sont correctes.

Une demande pour savoir si des variables contiennent une valeur correcte peut être utile par exemple avant de créer un code CN avec la fonction GC.

Une variable est parfaite lorsque l'état de la variable est descripteur .vld = 1

Programmation

Syntaxe :	**CVAR**(VarN)	
Description :	Vérifier que les variables ont un contenu correct	
Paramètres :	VarN	Liste des variables à vérifier. Il est possible de vérifier jusqu'à 29 variables séparées par des virgules. Il faut alors respecter la longueur de ligne maximale de 500. Le résultat de l'interrogation peut être :
	1 =	TRUE (toutes les variables ont un contenu correct)
	0 =	FALSE (une variable au moins n'a pas un contenu correct)

Exemple

```
IF CVAR == TRUE                   ; Vérification de toutes les variables
  VS8.SE = 1                      ; Si toutes les variables sont correctes,
                                  la touche logicielle VS8 est visible
```

```
ELSE
  VS8.SE = 2                                ; Si une variable contient une valeur in-
                                              correcte, la touche logicielle VS8 n'est
                                              pas activée
ENDIF

IF CVAR("VAR1", "VAR2") == TRUE
                                            ; Vérification des variables VAR1 et VAR2
  DLGL ("VAR1 et VAR2 sont OK")
                                            ; Si VAR1 et VAR2 sont renseignées correc-
                                              tement, la ligne de dialogue "VAR1 et
                                              VAR2 sont OK" s'affiche
ELSE
  DLGL ("VAR1 et VAR2 ne sont pas OK")
                                            ; Si VAR1 et VAR2 ont été renseignées de
                                              façon incorrecte, la ligne de boîte de
                                              dialogue "VAR1 et VAR2 ne sont pas OK"
                                              s'affiche
ENDIF
```

8.3.5 Fonction de fichier Copy Program (CP)

Description

La fonction CP (Copy Program) permet de copier des fichiers dans le système de fichiers IHM ou CN.

Programmation

Syntaxe :	**CP**("Fichier source", "Fichier cible")	
Description :	Copier fichier	
Paramètres :	Fichier source	Chemin d'accès complet du fichier source
	Fichier cible	Chemin d'accès complet du fichier cible

La valeur retournée (VAR1 définie en tant que variable auxiliaire) permet de savoir si la fonction a été exécutée avec succès :

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

Exemple

Cas d'application avec valeur en retour :

```
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 doit exister
```

Cas d'application sans valeur en retour :

```
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 doit exister
```

Voir aussi

Prise en charge de FILE_ERR : Variable FILE_ERR (Page 110)

8.3.6 Fonction de fichier Delete Program (DP)

Description

La fonction DP (Delete Program) supprime un fichier dans le système de fichiers IHM passif ou dans le système de fichiers CN actif.

Programmation

Syntaxe :	**DP**("Fichier")	
Description :	Effacer le fichier	
Paramètres :	Fichier	Chemin d'accès complet du fichier à supprimer

Exemple

La syntaxe suivante de la gestion des données est utilisée pour cette fonction :

- avec valeur en retour

  ```
  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 Le fichier a été supprimé.

  VAR1 = 1 Le fichier n'a pas été supprimé.
- sans valeur en retour :

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

8.3.7 Fonction de fichier Exist Program (EP)

Description

La fonction EP (Exist Program) vérifie si un programme CN particulier se trouve dans le système de fichiers CN ou IHM sous le chemin indiqué.

Programmation

Syntaxe :	**EP**("Fichier")	
Description :	Vérifier l'existence du programme CN	
Paramètres :	Fichier	Chemin d'accès complet du fichier dans le système de fichiers CN ou IHM
Valeur retournée :	Nom d'une variable à laquelle le résultat de la demande doit être affecté.	

La fonction EP gère la nouvelle syntaxe et l'ancienne logique (avec la syntaxe adaptée).

Le fichier est appelé directement avec un nom qualifié :

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

ou
`CF_CARD : /MPF.DIR/XYZ.MPF` (indique /user/sinumerik/data/prog)
ou
`LOC :` (correspond à CF_CARD)

Exemples de nouvelle syntaxe :

```
EP("//NC/WKS.DIR/TEST.WPD/XYZ.MPF",VAR1)
EP("CF_CARD:/mpf.dir/XYZ.MPF",VAR1)
EP("LOC:/mpf.dir/XYZ.MPF",VAR1)
; avec valeur en retour :
; VAR1 = 0 le fichier existe.
; VAR1 = 1 le fichier n'existe pas.
```

Exemples d'ancienne syntaxe :

```
EP("/MPF.DIR/CFI.MPF", VAR1)
; avec valeur en retour :
; VAR1 = M le fichier se trouve dans le système de fichiers IHM.
; VAR1 = N le fichier se trouve dans le système de fichiers CN.
; VAR1 = B le fichier se trouve dans les systèmes de fichiers IHM et CN.
```

Exemple

```
EP("/MPF.DIR/GROB.MPF",VAR1)                                    ; ancien - chemin d'accès maintenant
                                                                avec / au lieu de \


IF VAR1 == "M"
   DLGL("Le fichier se trouve dans le système de fichiers
IHM")
ELSE
   IF VAR1 == "N"
      DLGL("Le fichier se trouve dans le répertoire de fi-
chier CN")
   ELSE
     DLGL("Le fichier ne se trouve ni dans le système de
fichiers IHM, ni dans le système de fichiers CN")
   ENDIF
ENDIF
```

8.3.8 Fonction de fichier Move Program (MP)

Description

La fonction MP (Move Program) permet de copier des fichiers dans le système de fichiers IHM ou CN.

Programmation

Syntaxe :	**MP**("Source", "Cible")	
	`MP("CF_CARD:/MPF.DIR/MYPROG.MPF","//NC/MPF.DIR")`	
Description :	Déplacement du fichier	
Paramètres :	Fichier source	Chemin d'accès complet
	Fichier cible	Chemin d'accès complet
Valeur retournée	Résultat de l'interrogation	

Exemples

Avec valeur en retour :

```
MP("//NC/MPF.DIR/123.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)        ; dans CN
MP("//NC/MPF.DIR/123.MPF",VAR0,VAR1)                           ; Cible via la variable
MP(VAR4,VAR0,VAR1)                                             ; Source et cible via la variable
```

```
MP("CF_CARD:/mpf.dir/myprog.mpf","//NC/MPF.DIR/123.MPF",VAR1)     ; De la carte CF vers la CN
MP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/XYZ/123.mpf",VAR1)           ; De la CN vers la carte CF

; avec valeur en retour :
; VAR1 = 0 exécuté
; VAR1 = 1 non exécuté
```

8.3.9 Fonction de fichier Select Program (SP)

Description

La fonction SP (Select Program) sélectionne un fichier du système de fichiers CN actif afin de le traiter. Ainsi, le fichier doit être au préalable chargé dans la CN.

Programmation

Syntaxe :	**SP**("Fichier")	
Désignation :	Sélectionner un programme	
Paramètres :	"Fichier"	Chemin d'accès complet du fichier CN

Exemple

La syntaxe suivante de la gestion des données est utilisée pour cette fonction :

- avec valeur en retour
  `SP("//NC/MPF.DIR/MYPROG.MPF", VAR1)`
  VAR1 = 0 Le fichier a été chargé.
  VAR1 = 1 Le fichier n'a pas été chargé.
- sans valeur en retour :
  `SP("//NC/MPF.DIR/MYPROG.MPF")`

8.3.10 Accès fichier : RDFILE, WRFILE, RDLINEFILE, WRLINEFILE

Description

Les fonctions RDFILE et WRFILE sont disponibles pour accéder en lecture et en écriture aux fichiers avec une syntaxe INI.

Les fonctions RDLINEFILE et WRLINEFILE sont disponibles pour accéder en lecture et en écriture aux différentes lignes d'un fichier.

Programmation

Syntaxe :	**RDFILE**("Nom de fichier + chemin", "Section", "Key")	
Description :	Lecture à partir d'un fichier	
Paramètres :	Nom de fichier + chemin	Nom du fichier et chemin d'accès
	Section	Section dans le fichier INI
	Key	Clé dans le fichier INI

Syntaxe :	**WRFILE**(Value, "Nom de fichier + chemin", "Section", "Key")	
Description :	Ecriture dans un fichier	
Paramètres :	Value	Valeur à écrire
	Nom de fichier + chemin	Nom du fichier et chemin d'accès
	Section	Section dans le fichier INI
	Key	Clé dans le fichier INI

Syntaxe :	**RDLINEFILE**("Nom de fichier + chemin", numéro de ligne)	
Description :	Lecture d'une ligne à partir d'un fichier	
Paramètres :	Nom de fichier + chemin	Nom du fichier et chemin d'accès
	Numéro de ligne	Numéro de ligne La numérotation commence par 0 pour la première ligne.

Syntaxe :	**WRLINEFILE**(valeur, "Nom de fichier + chemin")	
Description :	Ecriture d'une ligne à la fin d'un fichier	
Paramètres :	Valeur	Valeur à écrire
	Nom de fichier + chemin	Nom du fichier et chemin d'accès

Remarque

- Les fichiers ne peuvent pas se trouver dans le système de fichiers de la CN (gestion des données).
- Si le fichier n'existe pas, si la fin du fichier est atteinte ou si une autre erreur quelconque se produit, les variables FILE_ERR et ERR sont définies en conséquence. Vous pouvez vérifier préalablement l'existence d'un fichier à l'aide de la fonction Exist Program (EP).
- Le fichier traité est généré avec le format de codage "UTF-8" (sans BOM (Byte Order Mask)). Lors de la lecture d'un fichier, celui-ci est attendu avec le format de codage "UTF-8".
- La fonction de fichiers Delete Program (DP) permet de supprimer explicitement le fichier en cas de besoin.

Exemples

Lecture à partir d'un fichier INI :

Condition/hypothèse :

Contenu du fichier "C:/tmp/myfile.ini" :

```
<...>
[MyData]
MyName=Daniel
<...>
```

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

Résultat :

MyVar contient maintenant la valeur "Daniel".

Ecriture dans un fichier INI :

Condition/hypothèse :

VARS=12

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

Résultat :

Contenu du fichier "C:/tmp/myfile.ini" :

```
<...>
[MySession]
NrOfSessions=12
<...>
```

Lecture de la 4e ligne d'un fichier :

Condition/hypothèse :

Contenu de 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)
```

Résultat :

MyVar contient maintenant la valeur " R[0]=R[0]+1"

Ecriture à la fin d'un fichier :

Condition/hypothèse :

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

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

Résultat :

Contenu de c:/tmp/mypp.mpf :

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

8.3.11 Dialog Line (DLGL)

Description

Dans la ligne de dialogue de la boîte de dialogue, des textes succincts peuvent être affichés dans certaines situations (messages ou aides).

Nombre de caractères possibles en taille de police standard : environ 50 caractères

Remarque

Les fonctions LS, LM, DLGL, EXIT et EXITLS ne doivent pas être configurées dans les méthodes SUSPEND ou RESUME. L'utilisation des fonctions dans ces méthodes empêchera l'exécution des fonctions.

Programmation

Syntaxe :	**DLGL**("Chaîne")	
Description :	Transférer le texte dans la ligne de boîte de dialogue	
Paramètres :	string	Texte qui figure dans la ligne de boîte de dialogue

Exemple

```
IF Var1 > Var2
                              ; Le texte "Valeur trop grande !" s'affiche dans la li-
                              gne de boîte de dialogue si Variable1 > Variable2.
  DLGL("Valeur trop gran-
de !")
ENDIF
```

8.3.12 DEBUG

Description

La fonction DEBUG met à votre disposition une aide à l'analyse lors de la phase de conception de masques utilisateur Run MyScreen. La fonction DEBUG permet d'évaluer en cours d'exécution une expression transmise entre parenthèses. Le résultat est ajouté sous la forme d'une entrée distincte dans le journal de bord "easyscreen_log.txt".

L'horodatage actuel est mis en préfixe entre crochets devant chaque entrée (voir exemple ci-dessous).

Il est recommandé de supprimer à nouveau les sorties DEBUG dans les parties à temps critique lorsque cela est possible. Il est recommandé notamment de supprimer les commentaires des sorties DEBUG après la fin de la phase de conception. Les accès en écriture au journal de bord "easyscreen_log.txt" grandissant sans cesse entraînent un ralentissement.

Programmation

Syntaxe :	**DEBUG**(Expression)
Description :	Effectuer une entrée dans le journal de bord "easyscreen_log.txt"
Paramètres :	Expression à évaluer à partir de laquelle une entrée doit être générée dans le journal de bord

Exemple

```
IF Var1 > Var2
  DEBUG("Value of ""Var1"" : " << Var1)
                                        ; Entrée dans "easyscreen_log_txt:
                                        [10:22:40.445] DEBUG : Value of "Var1":
                                        123,456
ENDIF
```

8.3.13 Quitter le dialogue (EXIT)

Description

La fonction EXIT permet de quitter une boîte de dialogue et de revenir à la boîte de dialogue principale. S'il n'existe pas de boîte de dialogue principale, vous quittez la nouvelle interface utilisateur et vous revenez dans l'application standard.

Remarque

Les fonctions LS, LM, DLGL, EXIT et EXITLS ne doivent pas être configurées dans les méthodes SUSPEND ou RESUME. L'utilisation des fonctions dans ces méthodes empêchera l'exécution des fonctions.

Programmation (sans paramètres)

Syntaxe :	**EXIT**
Description :	Quitter une boîte de dialogue
Paramètres :	- aucun -

Exemple

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

Description

Si la boîte de dialogue actuelle est appelée avec la variable de transfert, la valeur des variables peut être modifiée et être retournée dans la boîte de dialogue initiale.

Les valeurs des variables sont affectées aux variables transférées de la boîte de dialogue initiale à la boîte de dialogue consécutive à l'aide de la fonction "LM". Il est possible de transmettre jusqu'à 20 valeurs de variable séparées par des virgules.

Remarque

L'ordre des variables ou des valeurs de variables doit être effectué conformément à l'ordre des variables de transfert de la fonction LM afin que l'affectation soit sans équivoque. Si certaines valeurs de variable ne sont pas indiquées, ces variables de transfert ne sont pas modifiées. Les variables de transfert modifiées sont immédiatement valables dans la boîte de dialogue initiale dès que la fonction LM a été utilisée.

Programmation avec variable de transfert

Syntaxe :	**EXIT**[(VARx)]	
Description :	Quitter la boîte de dialogue avec transfert d'une ou plusieurs variables	
Paramètres :	VARx	Désignation des variables

Exemple

```
//M(Masque1)
...
PRESS(HS1)
  LM("MASQUE2","CFI.COM",1, POSX, POSY,
DIAMÈTRE)
                                        ; Interrompre Masque1 et afficher Mas-
                                        que2. Transférer les variables POSX, POSY
                                        et DIAMÈTRE.
  DLGL("Masque2 terminé")               ; Après le retour du Masque2, la ligne de
                                        boîte de dialogue de Masque1 affiche le
                                        texte : Masque2 terminé.
END_PRESS
...
//END

//M(Masque2)
...
PRESS(HS1)
  EXIT(5, , DIAMÈTRE_CALCULÉ)
                                        ; Sortie de Masque2 et retour à la ligne
                                        du Masque1 après LM. La valeur 5 est af-
                                        fectée à la variable POSX et la valeur de
                                        la variable DIAMÈTRE_CALCULÉ est affectée
                                        à la variable DIAMÈTRE. La variable POSY
                                        garde sa valeur actuelle.
END_PRESS
...
//END
```

8.3.14 Manipulation dynamique des listes des champs de basculement ou de liste

Description

Les fonctions LISTADDITEM, LISTINSERTITEM, LISTDELETEITEM et LISTCLEAR servent à manipuler de manière dynamique les listes des champs de basculement ou de liste.

Ces fonctions ne s'appliquent qu'aux variables ayant leur propre liste, comme p. ex. :

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

Si la variable présente un tableau, p. ex. `DEF VAR_AC3 = (I/* MYARRAY)`, ces fonctions ne sont pas disponibles, car autrement l'ensemble du tableau devrait être modifié.

Une variable doit contenir au moins une valeur définie dans la ligne DEF. Le type de liste "simple" ou "étendue" est ainsi défini.

Ensuite, il est toutefois autorisé de supprimer complètement la liste et le cas échéant de la reconstruire entièrement. Le type "simple" ou "étendu" doit cependant être conservé et ne peut pas être modifié de manière dynamique.

Programmation

Syntaxe :	**LISTINSERTITEM**(Nom de variable, Position, ItemValue[, ItemDispValue])	
Description :	Insertion d'un élément à une certaine position de lecture à partir d'un fichier	
Paramètres :	Nom de variable	
	Position	Position à laquelle un élément doit être ajouté à la liste
	ItemValue	Valeur de l'entrée de la liste
	ItemDispValue	Valeur telle qu'elle doit être représentée dans la liste

Syntaxe :	**LISTADDITEM**(Nom de variable, ItemValue[, ItemDispValue])	
Description :	Ajout d'un élément à la fin de la liste	
Paramètres :	Nom de variable	
	ItemValue	Valeur de l'entrée de la liste
	ItemDispValue	Valeur telle qu'elle doit être représentée dans la liste

Syntaxe :	**LISTDELETEITEM**(Nom de variable, Position)	
Description :	Suppression d'un certain élément	
Paramètres :	Nom de variable	
	Position	Position de l'élément à supprimer de la liste

Syntaxe :	**LISTCOUNT**(Nom de variable)	
Description :	Fournit le nombre actuel d'éléments de la liste	
Paramètres :	Nom de variable	

Syntaxe :	**LISTCLEAR**(Nom de variable)	
Description :	Suppression de la liste complète	
Paramètres :	Nom de variable	

Exemples

Remarque

Dans les exemples suivants, chaque exemple est basé sur l'exemple précédent. L'ordre des exemples est donc important pour comprendre les résultats respectifs.

Condition/hypothèse :

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

Ajout d'un élément -1="Undefined" à la fin de la liste :

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

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

Insertion d'un élément 99="Maybe" à la position 2 :

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

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

Détermination du nombre actuel d'éléments de la liste :

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

Résultat : REG[10] = 4

Suppression de l'élément à la position 1 :

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

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

Suppression de la liste complète :

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

Résultat : la liste est vide

8.3.15 Evaluate (EVAL)

Description

La fonction EVAL évalue une expression transmise et l'exécute ensuite. Ainsi, des expressions peuvent être créées en cours d'exécution. Cela peut être utile pour des accès indexés à des variables.

Programmation

Syntaxe :	**EVAL**(exp)	
Description :	Evaluer l'expression	
Paramètres :	exp	Expression logique

Exemple

```
VAR1=(S)
VAR2=(S)
VAR3=(S)
VAR4=(S)
CHANGE()
  REG[7] = EVAL("VAR"<<REG[5])  : L'expression entre parenthèses donne VAR3 si la
                                valeur de REG[5] est égale à 3. REG[7] se voit af-
                                fecter la valeur de VAR3.
  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)

Description

La fonction EXITLS permet de quitter l'interface utilisateur courante et de charger une barre de TL définie.

Remarque

Les fonctions LS, LM, DLGL, EXIT et EXITLS ne doivent pas être configurées dans les méthodes SUSPEND ou RESUME. L'utilisation des fonctions dans ces méthodes empêchera l'exécution des fonctions.

Programmation

Syntaxe :	**EXITLS**("Barre de TL"[, "Nom de chemin"])	
Description :	charger la barre de touches logicielles en quittant	
Paramètres :	Barre de touches logicielles	Nom de la barre de TL à charger
	Chemin	Chemin du répertoire de la barre de TL à charger

Exemple

```
PRESS(HS1)
    EXITLS( "Barre1", "AEDITOR.COM" )
END_PRESS
```

8.3.17 Function (FCT)

Description

Les fonctions externes sont placées dans un fichier DLL et déclarées par une entrée dans les lignes de définition du fichier de configuration.

Remarque

La fonction externe doit avoir au moins un paramètre de retour.

Programmation

Syntaxe :	**FCT**Nom de fonction = ("Fichier"/Type de retour/Types de paramètre fixe/Types de paramètre variable)	
	FCT InitConnection = ("c:\tmp\xyz.dll"/I/R,I,S/I,S)	
Description :	L'appel d'une fonction externe peut être effectué à partir de la méthode LOAD ou de la méthode PRESS, par exemple.	
Paramètres :	Nom de la fonction	Nom de la fonction externe
	Fichier	Chemin d'accès complet du fichier DLL
	Type de retour	Type de données de la valeur retournée
	Type de paramètre fixe	Paramètre Value
	Type de paramètre variable	Paramètres de référence
	Les types de données sont séparés par une virgule.	

L'appel de la fonction externe peut être effectué à partir de la méthode LOAD ou de la méthode PRESS, par exemple.

Exemple :

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

Structure de la fonction externe

La fonction externe doit respecter une signature prédéfinie :

Syntaxe :	extern "C" dllexport void InitConnection (ExtFctStructPtr FctRet, ExtFctStructPtr FctPar, char cNrFctPar)
Description :	Export DLL uniquement pour la mise en œuvre sous Windows Les qualificateurs et les paramètres de transfert sont prédéfinis. Les paramètres d'appel propres sont transmis via les structures transférées.

Paramètres :	cNrFctPar	Nombre de paramètres d'appel = nombre d'éléments structurels dans FctPar
	FctPar	Pointeur sur un champ d'éléments structurels qui contiennent les paramètres d'appel respectifs avec le type de données.
	FctRet	Pointeur sur une structure pour le retour de la valeur de la fonction avec le type de données.

Définition de la structure de transfert

```
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;
```

Si la fonction externe doit être développée indépendamment de la plate-forme (Windows, Linux), le mot-clé __declspec(dllexport) ne doit pas être utilisé. Ce mot-clé est exclusivement requis sous Windows. Sous Qt, on peut utiliser par exemple la macro suivante.

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

La déclaration de la fonction est la suivante :

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

Si les images configurées avec "Run MyScreens" sont utilisées sur NCU et PCU/PC, l'extension du fichier binaire doit être supprimée :

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

Lorsque l'information de chemin absolu est omise, "Run MyScreens" cherche le fichier binaire d'abord dans le répertoire configuré.

8.3.18 Generate Code (GC)

Description

La fonction GC (Generate Code) génère le code CN à partir de la méthode OUTPUT.

Programmation

Syntaxe :	**GC**("Descripteur"[,"Fichier cible"][,Opt],[Append])	
Description :	Générer un code CN (la fonction GC n'est possible que dans le CN.)	
Paramètres :	Descripteur	Nom de la méthode OUTPUT qui sert de base à la génération de code
	Fichier cible	Chemin d'accès du fichier cible pour le système de fichiers IHM ou CN. Si le fichier cible n'est pas indiqué (seulement possible dans l'aide à la programmation), le code est écrit à l'emplacement du curseur dans le fichier actuellement ouvert.
	opt	Option pour la génération de commentaire
	0 :	(Préréglage) Créer le code avec le commentaire pour permettre la décompilation (voir également Décompilation (Page 175)).
	1 :	Ne pas créer de commentaire avec le code généré. **Remarque :** Ce code ne peut être décompilé (voir aussi Décompilation (Page 175)).
	Append	Ce paramètre est pris en compte que si un fichier cible est indiqué.
	0 :	(préréglage) Si le fichier existe déjà, l'ancien contenu est supprimé.
	1 :	Si le fichier existe déjà, le nouveau code est écrit au début du fichier.
	2 :	Si le fichier existe déjà, le nouveau code est ajouté à la fin.

Exemple

```
//M(TestGC/"Génération de code :")
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)                ; Écriture du code CN dans le fichier \MPF.DIR
                                    \MESSEN.MPF à partir de la méthode OUTPUT :
                                    Cycle123(123, -6)
                                    M30
END_PRESS
```

Insertion du fichier cible

La fonction GC n'est possible que dans le CN. Pour déplacer les fichiers, utiliser les fonctions CP ou MP. Voir exemple ci-dessous :

```
; 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")
```

Décompilation

- **Aucune indication du fichier cible :**
  La fonction GC ne peut être utilisée que dans l'aide à la programmation et elle inscrit le code CN dans le fichier ouvert actuellement dans l'éditeur. La décompilation du code CN est possible. Si la fonction GC est configurée dans "Run MyScreens" sans indication du fichier cible, un message d'erreur s'affiche lors de l'exécution.
- **Indication du fichier cible :**
  Le code généré à partir de la méthode OUTPUT est entré dans le fichier cible. Si le fichier cible n'est pas disponible, il est créé dans le système de fichiers CN. Si le fichier cible est situé dans le système de fichiers IHM, le fichier est archivé sur le disque dur. Les lignes de commentaires utiles (informations utiles pour la décompilation) ne sont pas créées, c'est-à-dire qu'une décompilation n'est pas possible.

Particularités lors de la décompilation

La fonction GC ne peut pas être appelée dans les sous-dialogues ils peuvent contenir des variables provenant de la boîte de dialogue principale et qui ne seraient pas disponibles par un appel direct.

En cas modifications manuelles sur le code généré avec l'éditeur, le nombre de signes des valeurs créées par la génération de code, ne doit pas être modifié. Cela empêcherait une décompilation.

Remède :

1. Décompilation
2. Modification à l'aide de la boîte de dialogue configurée (p. ex. 99 → 101)
3. GC

8.3.19 Fonctions de mot de passe

Vue d'ensemble

Les fonctions de mot de passe suivantes sont disponibles :

- Définition du mot de passe
- Suppression du mot de passe
- Modification du mot de passe

Fonction HMI_LOGIN : définir un nouveau mot de passe

Syntaxe :	**HMI_LOGIN**(Passwd)	
Description :	La fonction HMI_LOGIN() permet d'envoyer un mot de passe au NCK avec lequel le niveau d'accès actuel est paramétré.	
Paramètres :	Passwd	Mot de passe

Exemple

```
REG[0] = HMI_LOGIN("CUSTOMER")          ; Résultat = TRUE, si le mot de passe est dé-
                                        fini avec succès, sinon FALSE
```

Fonction HMI_LOGOFF : supprimer le mot de passe

Syntaxe :	**HMI_LOGOFF**
Description :	La fonction HMI_LOGOFF permet de réinitialiser le niveau d'accès actuel.
Paramètres :	- aucun -

Exemple

```
REG[0] = HMI_LOGOFF                     ; Résultat = TRUE, si le mot de passe est sup-
                                        primé avec succès, sinon FALSE
```

Fonction HMI_SETPASSWD : modifier le mot de passe

Syntaxe :	**HMI_SETPASSWD**(AC, Passwd)	
Description :	La fonction HMI_SETPASSWD permet de modifier le mot de passe du niveau d'accès actuel ou d'un niveau inférieur.	
Paramètres :	AC	Niveau d'accès
	Passwd	Mot de passe

Exemple

```
REG[0] = HMI_SETPASSWD(4,"MYPWD")      ; Résultat = TRUE, si le mot de passe est modifié avec succès, sinon FALSE
```

8.3.20 Load Array (LA)

Description

La fonction LA (Load Array) permet de charger un tableau à partir d'un autre fichier.

Programmation

Syntaxe :	**LA**(descripteur [, fichier])	
Description :	Charger le tableau à partir du fichier	
Paramètres :	Descripteur	Nom du tableau à recharger
	Fichier	Fichier dans lequel le tableau est défini

Remarque

Si un tableau du fichier courant de configuration doit être remplacé par un tableau d'un autre fichier de configuration, les tableaux doivent porter le même nom.

Exemple

```
                                       ; Extrait du fichier   masque.com
DEF VAR2 = (S/*ARR5/"Off"/,"Champ bascule")
PRESS(HS5)
  LA("ARR5","arrayext.com")            ; Chargement du tableau ARR5 à partir du fichier arrayext.com
```

```
  VAR2 = ARR5[0]                              ; Au lieu de "Marche"/"Arrêt", le champ
                                              bascule de VAR2 affiche
                                              "Haut"/"Bas"/"Droite"/"Gauche"
END_PRESS
//A(ARR5)
("Off"/"On")
//END
                                              ; Extrait du fichier arrayext.com
//A(ARR5)
("Haut"/"Bas"/"Droite"/"Gauche")
//END
```

Remarque

Notez qu'une valeur valide doit être attribuée à une variable lorsqu'un autre tableau a été affecté au champ bascule de la variable avec la fonction LA.

8.3.21 Load Block (LB)

Description

La fonction LB (Load Block) permet de charger des blocs dans les sous-programmes en cours d'exécution. Il est préférable de configurer LB dans une méthode LOAD pour que les sous-programmes chargés puissent être appelés à tout moment.

Remarque

Les sous-programmes peuvent également être définis directement dans une boîte de dialogue et ils n'ont alors pas besoin d'être chargés.

Programmation

Syntaxe :	**LB**("Nom de bloc"[,"Fichier"])	
Description :	Charger le sous-programme en cours d'exécution	
Paramètres :	Nom de bloc	Nom de l'identifiant de bloc
	Fichier	Indication de chemin du fichier de configuration par défaut = fichier de configuration actuel

Exemple

```
LOAD
  LB("PROG1")              ; Le bloc "PROG1" est recherché dans le fichier de confi-
                           guration actuel puis chargé.
  LB("PROG2","XY.COM")     ; Le bloc "PROG2" est recherché dans le fichier de confi-
                           guration XY.COM puis chargé.
END_LOAD
```

8.3.22 Load Mask (LM)

Description

La fonction LM permet de charger une nouvelle boîte de dialogue. Selon le mode de changement de boîte de dialogue (voir le tableau ci-dessous), cette fonction permet de réaliser une boîte de dialogue de message.

Remarque

Les fonctions LS, LM, DLGL, EXIT et EXITLS ne doivent pas être configurées dans les méthodes SUSPEND ou RESUME. L'utilisation des fonctions dans ces méthodes empêchera l'exécution des fonctions.

Boîte de dialogue principale / boîte de dialogue secondaire

Une boîte de dialogue qui appelle une autre boîte de dialogue et qui ne se ferme pas, est appelée boîte de dialogue principale. Une boîte de dialogue qui est appelée à partir d'une boîte de dialogue principale, est appelée boîte de dialogue secondaire.

Programmation

Syntaxe :	**LM**("Descripteur"[,"Fichier"] [,MSx [, VARx]])
Description :	Charger la boîte de dialogue

Paramètres :	Descripteur	Nom de la boîte de dialogue à charger
	Fichier	Chemin d'accès (système de fichiers IHM ou CN) du fichier de configuration ; réglage par défaut : fichier de configuration actuel
	MSx	Mode du changement de boîte de dialogue
	0 :	(préréglage) La boîte de dialogue actuelle est déplacée et la nouvelle boîte de dialogue est chargée et affichée. EXIT permet de revenir à l'application standard. Le paramètre MSx permet de déterminer si la boîte de dialogue actuelle doit être fermée ou non en cas de changement de boîte de dialogue. Si la boîte de dialogue actuelle est conservée, les variables peuvent être reprises dans la nouvelle boîte de dialogue. L'avantage du paramètre MSx est que les boîtes de dialogue n'ont pas besoin d'être en permanence réinitialisées en cas de changement ainsi les données et la structure de la boîte de dialogue actuelle sont conservées et le transfert de données est facilité.
	1 :	La boîte de dialogue principale actuelle est interrompue à partir de la fonction LM et la nouvelle boîte de dialogue secondaire est chargée et affichée (p. ex. pour la réalisation d'une boîte de dialogue de message). En appuyant sur EXIT, la boîte de dialogue secondaire est fermée et l'utilisateur revient à la boîte de dialogue principale dans l'état où elle se trouvait au moment de l'interruption. En cas d'interruption du bloc UNLOAD, le traitement n'est plus effectué dans la boîte de dialogue principale.
	VARx	Condition : MS1 Liste des variables pouvant être reprises de la boîte de dialogue principale vers la boîte de dialogue secondaire. Il est possible de transmettre jusqu'à 20 variables séparées par des virgules.

Remarque

Le paramètre VARx ne fait que de transférer la valeur des variables, c'est-à-dire que les variables peuvent être lues et écrites dans la boîte de dialogue secondaire, mais n'y sont pas visibles. La restitution des variables de la boîte de dialogue secondaire vers la boîte de dialogue principale est possible à l'aide de la fonction EXIT.

Exemple

MASQUE1 (boîte de dialogue principale) : débranchement dans la boîte de dialogue secondaire MASQUE2 avec transfert de variable

```
PRESS(HS1)

  LM("MASQUE2","CFI.COM",1, POSX, POSY, DIAMÈTRE)
                                    ; Interrompre Masque1 et afficher Masque2 : Les
                                    variables POSX, POSY et DIAMÈTRE sont ainsi
                                    transférées.
```

MASQUE1 (boîte de dialogue principale) : débranchement dans la boîte de dialogue secondaire MASQUE2 avec transfert de variable

```
  DLGL("Masque2 terminé")            ; Après le retour du Masque2, la ligne de boîte
                                     de dialogue de Masque1 affiche le texte : Mas-
                                     que2 terminé.
END_PRESS
```

MASQUE2 (boîte de dialogue secondaire) : Retour vers la boîte de dialogue principale MASQUE1 avec transfert des contenus de variable

```
PRESS(VS8)


  EXIT(POSX, POSY, DIAMÈTRE)
                                     ; Fermer masque2 et continuer masque1 : Les va-
                                     riables POSX, POSY et DIAMÈTRE modifiées sont
                                     alors transférées.
END_PRESS
```

8.3.23 Load Softkey (LS)

Description

La fonction LS permet d'afficher une autre barre de touches logicielles.

Remarque

Les fonctions LS, LM, DLGL, EXIT et EXITLS ne doivent pas être configurées dans les méthodes SUSPEND ou RESUME. L'utilisation des fonctions dans ces méthodes empêchera l'exécution des fonctions.

Programmation

Syntaxe :	**LS**("Descripteur"[, "Fichier"][, Merge])	
Description :	Afficher la barre de touches logicielles	
Paramètres :	Descripteur	Nom de la barre de touches logicielles
	Fichier	Chemin d'accès (système de fichiers IHM ou CN) du fichier de configuration Préréglage : fichier de configuration actuel
	Merge	
	0 :	Toutes les touches logicielles existantes sont supprimées, les touches logicielles nouvellement configurées sont entrées.
	1 :	Préréglage Seules les touches logicielles nouvellement configurées écrasent les touches logicielles existantes. Les autres touches logicielles (= touches logicielles de l'application IHM) conservent leur fonctionnalité et leur texte.

Exemple

```
PRESS(HS4)
  LS("Barre2",,0)     ; Barre2 écrase la barre de touches logicielles existante, les
                      touches logicielles affichées sont supprimées.
END_PRESS
```

Remarque

Tant que l'interpréteur n'a pas encore ouvert de boîte de dialogue (c'est-à-dire qu'aucune fonction LM n'a été utilisée), la seule action possible est de configurer une instruction LS ou LM dans les méthodes PRESS du bloc de description de touches logicielles d'accès et de la barre d'affichage.

Les fonctions LS et LM ne peuvent être appelées qu'au sein d'une méthode PRESS des touches logicielles et ne peuvent servir en tant que réaction aux touches de navigation (PU, PD, SL, SR, SU, SD)

8.3.24 Load Grid (LG)

Description

La description de tableau (grille) peut être mise à disposition de manière dynamique dans des méthodes (p. ex. LOAD) au moyen de la méthode LG.

Afin qu'un tableau puisse être affecté à l'aide de la méthode LG, la variable doit déjà être définie comme variable de grille et renvoyer à un tableau existant et valide.

Programmation

Syntaxe :	**LG**(nom de grille, nom de variable [,nom de fichier])	
Description :	Charger un tableau	
Paramètres :	Nom de grille	Nom du tableau (grille) entre guillemets (")
	Nom de variable	Nom de la variable à laquelle le tableau doit être affecté, entre guillemets (")
	Nom du fichier	Nom du fichier dans lequel le tableau (grille) est défini, entre guillemets (") Il ne doit être indiqué que si le tableau n'est pas défini dans le fichier dans lequel la variable est également définie.

Exemple

```
//M(MyGridSample/"My Grid Sample")
DEF MyGridVar=(R/% MyGrid1///WR2////100,,351,100)
```

```
LOAD
   LG("MyGrid1","MyGridVar","mygrids.com")
END_LOAD
```

Contenu de 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 Sélection multiple SWITCH

Description

Une instruction SWITCH permet de vérifier plusieurs valeurs d'une variable.

L'expression configurée dans l'instruction SWITCH est comparée successivement avec toutes les valeurs configurées dans les instructions CASE suivantes.

En présence d'une différence, une comparaison avec l'instruction CASE suivante est réalisée. Si une instruction DEFAULT suit et si aucune instruction CASE n'était jusqu'ici égale à l'expression configurée dans l'instruction SWITCH, les instructions qui suivent l'instruction DEFAULT sont exécutées.

En l'absence de différence, les instructions suivantes sont exécutées jusqu'à ce qu'une instruction CASE, DEFAULT ou END_SWITCH suive.

Exemple

```
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)

Description

L'instruction MRNP permet de transmettre plusieurs variables CN/AP avec un accès au registre. Cet accès est nettement plus rapide que la lecture par accès individuel. Les variables CN/AP doivent provenir de la même zone dans une même instruction MRNP.

Les zones des variables CN/AP sont structurés de la façon suivante :

- Données générales CN ($MN..., $SN.., /nck/...)
- Données CN spécifiques à un canal ($MC..., $SC.., /channel/...)
- Données AP (DB..., MB.., /plc/...)
- Données CN spécifiques au même axe ($MA..., $SA..)

Programmation

Syntaxe :	**MRNP**(nom de variable1*nom de variable2[* ...], index registre)
Description :	Lire plusieurs variables
Paramètres :	Pour les noms de variable, "*" sert de séparateur. Les valeurs sont reprises dans le registre REG[Registerindex] dans l'ordre d'apparition des noms de variables dans l'instruction. Dans ce contexte : La valeur de la première variable se trouve dans REG[indice de registre]. La valeur de la deuxième variable se trouve dans REG[indice de registre + 1] etc.

Remarque

Important : la liste de variables est limitée à 500 signes et le nombre de registres est limité.

Exemple

```
MRNP("$R[0]*$R[1]*$R[2]*$R[3]",1)      ;REG[1] à REG[4] est décrit avec la valeur
                                        des variables $R[0] à $R[3].
```

Variable CN

Tous les paramètres machine et les données de réglage ainsi que les paramètres R sont disponibles ainsi que certaines variables système (voir aussi : AUTOHOTSPOT).

Toutes les variables utilisateur globales et spécifiques au canal (GUD) sont accessibles. Les variables utilisateur locales et globales ne peuvent pas être traitées.

Paramètres machine	
Paramètre machine global	$MN_...
Paramètre machine spécifique à l'axe	$MA_...
Paramètre machine spécifique au canal	$MC_...

Données de réglage	
Paramètre de réglage global	$SN_...
Paramètre de réglage spécifique à l'axe	$SA_...
Paramètre de réglage spécifique au canal	$SC_...

Variables système	
Paramètres R 1	$R[1]

Variable AP

Toutes les données AP sont disponibles.

Données AP	
Octet y bit z du bloc de données x	DBx.DBXy.z
Octet y du bloc de données x	DBx.DBBy
Mot y du bloc de données x	DBx.DBWy
Double mot y du bloc de données x	DBx.DBDy
Real y du bloc de données x	DBx.DBRy
Octet de mémentos x bit y	Mx.y
Octet de mémentos x	MBx
Mot de mémentos x	MWx
Double mot de mémentos x	MDx
Octet d'entrée x bit y	Ix.y ou Ex.y
Octet d'entrée x	IBx ou EBx
Mot d'entrée x	IWx ou EWx
Double mot d'entrée x	IDx ou EDx
Octet de sortie x bit y	Qx.y ou Ax.y
Octet de sortie x	QBx ou ABx
Mot de sortie x	QWx ou AWx
Double mot de sortie x	QDx ou ADx
String y avec longueur z à partir du bloc de données x	DBx.DBSy.z

8.3.27 P_LOGICAL_FILEPATH

Description

La fonction P_LOGICAL_FILEPATH renvoie le nom et le chemin logique du programme pièce en cours de traitement dans l'éditeur.

Remarque

Cette fonction n'est disponible que dans le contexte des masques de cycle.

Programmation

Syntaxe :	P_LOGICAL_FILEPATH()	
Description :	Déterminer le nom et le chemin logique du programme pièce en cours de traitement dans l'éditeur. p. ex. "//NC/MPF.DIR/HUGO.MPF"	
Paramètres :	- aucun -	

Voir aussi

P_REAL_FILEPATH (Page 169)

8.3.28 P_REAL_FILEPATH

Description

La fonction P_REAL_FILEPATH renvoie le nom et le chemin réel du programme pièce en cours de traitement dans l'éditeur.

Remarque

Cette fonction n'est disponible que dans le contexte des masques de cycle.

Programmation

Syntaxe :	P_REAL_FILEPATH()	
Description :	Déterminer le nom et le chemin réel du programme pièce en cours de traitement dans l'éditeur. p. ex. "/_N_MPF_DIR/_N_HUGO_MPF"	
Paramètres :	- aucun -	

Voir aussi

P_LOGICAL_FILEPATH (Page 169)

8.3.29 Services PI

Description

La fonction PI_START permet de démarrer des services PI (services d'instance de programme) de l'AP dans la zone CN.

Remarque

Vous trouverez une liste des services PI disponibles dans la description fonctionnelle Fonctions de base.

Programmation

Syntaxe :	**PI_START**("Chaîne de transfert")	
Description :	Exécuter un service PI	
Paramètres :	"Chaîne de transfert"	La chaîne de transfert doit être placée entre guillemets ("), contrairement à la documentation OEM.

Exemple

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

Remarque

Les services PI dépendant du canal se réfèrent toujours au canal actuel.

Les services PI des fonctions d'outils (zone TO) se rapportent toujours à la zone TO à laquelle le canal actuel est attribué.

8.3.30 Lecture (RNP) et écriture (WNP) de variable système ou utilisateur

Description

La commande RNP (Read NC PLC) permet de lire des variables CN ou AP ou des paramètres machines.

Programmation

Syntaxe :	**RNP** ("Variable système ou utilisateur", Valeur)	
Description :	Lire des variables CN ou AP ou des paramètres machine	
Paramètres :	Variable système ou utilisateur	Nom des variables CN ou AP
	Valeur	Valeur qui doit être écrite dans la variable système ou utilisateur. Si la valeur est de type String, elle doit être inscrite entre doubles quotes.

Exemple

```
VAR2=RNP("$AA_IN[2]")         ; Lecture de la variable CN
```

Description

La commande WNP (Write NC PLC) permet d'écrire des variables CN ou AP ou des paramètres machines.

Les accès aux variables CN et AP sont réexécutés pour chaque traitement de la fonction WNP. C'est-à-dire qu'un accès CN ou AP est toujours exécuté dans une méthode CHANGE. Ceci présente de l'intérêt lorsqu'une variable système ou utilisateur change souvent de valeur. Si un accès CN/AP ne doit être effectué qu'une seule fois, il faut le configurer dans la méthode LOAD ou UNLOAD.

Programmation

Syntaxe :	**WNP**("Variable système ou utilisateur", Valeur)	
Description :	Ecrire des variables CN ou AP ou des paramètres machine	
Paramètres :	Variable système ou utilisateur	Nom des variables CN ou AP
	Valeur	Valeur qui doit être écrite dans la variable système ou utilisateur. Si la valeur est de type String, elle doit être inscrite entre doubles quotes.

Exemple

```
WNP("DB20.DBB1",1)        ; Ecriture de la variable AP
```

8.3.31 RESIZE_VAR_IO et RESIZE_VAR_TXT

Description

Les instructions `RESIZE_VAR_IO()` et `RESIZE_VAR_TXT()` permettent de modifier la géométrie de la partie saisie et visualisation ou de la partie texte d'une variable.

Une fois la nouvelle géométrie définie, tous les champs du masque sont positionnés comme si le masque avait été configuré avec ces positions depuis le début. Ainsi, tous les champs sont correctement orientés les uns par rapport aux autres en fonction de la configuration.

Programmation

Syntaxe :	**RESIZE_VAR_IO**(Nom de variable, [X], [Y], [largeur], [hauteur]) **RESIZE_VAR_TXT**(Nom de variable, [X], [Y], [largeur], [hauteur])	
Description :	Modification de la géométrie de la partie saisie et visualisation ou de la partie texte d'une variable.	
Paramètres :	Nom de variable	Nom de la variable dont la géométrie de la partie saisie et visualisation ou de la partie texte doit être modifiée.
	X	Coordonnée X gauche/haut
	Y	Coordonnée Y gauche/haut
	Largeur	Largeur
	Hauteur	Hauteur

Remarque

Pour chaque valeur X, Y, largeur ou hauteur non indiquée, la valeur précédente est conservée. Si l'on indique -1 pour l'une de ces valeurs, la partie prescrite par "Run MyScreens" de la position par défaut est définie.

Exemple

```
RESIZE_VAR_IO("MyVar1", 200, , 100)    ; La partie E/S de la variable "MyVar1" est
                                       décalée de 200 pixels au niveau de la position
                                       X, la largeur est définie à 100 pixels. La po-
                                       sition Y et la hauteur de la valeur précédente
                                       sont conservées.
```

8.3.32 Registre (REG)

Description du registre

Les registres sont nécessaires pour échanger des données entre différentes boîtes de dialogue. Les registres sont affectés à une boîte de dialogue. Ils sont générés lors du chargement de la première boîte de dialogue, contant par défaut la valeur 0 ou une chaîne vide.

Remarque

Les registres ne doivent pas être directement utilisés dans une méthode OUTPUT pour générer du code CN.

Programmation

Syntaxe :	**REG**[x]	
Description :	Définir le registre	
Paramètres :	x	Indice de registre où x = 0...19 ; Type : REAL ou STRING = VARIANT Les registres avec x ≥ 20 sont déjà utilisés par Siemens.

Description de la valeur de registre

L'affectation des valeurs dans les registres est configurée dans une méthode.

Remarque

Si une nouvelle boîte de dialogue est créée à partir d'une boîte de dialogue avec la fonction LM, le contenu des registres est repris automatiquement dans la nouvelle boîte de dialogue et est disponible pour de nouveaux calculs dans la seconde boîte de dialogue.

Programmation

Syntaxe :	Descripteur.val = Valeur de registre ou Descripteur = valeur du registre	
Description :		
Paramètres :	Descripteur	Nom de registre
	Valeur de registre	Valeur de registre

Exemple

```
UNLOAD
```

```
  REG[0] = VAR1          ; Affectation de la valeur de Variable1 au registre 0
END_UNLOAD

UNLOAD
  REG[9].VAL = 84        ; Affectation de la valeur 84 au registre 9
END_UNLOAD
                         ; Dans les boîtes de dialogue suivantes, ces registres peu-
                         vent être de nouveau affectés à des variables locales dans
                         une méthode.
LOAD
  VAR2 = REG[0]
END_LOAD
```

Description de l'état du registre

La caractéristique Etat permet de demander au cours de la configuration si un registre contient une valeur valide.

L'interrogation de l'état de registre peut servir notamment à écrire une valeur dans un registre si une boîte de dialogue est utilisée comme boîte de dialogue principale.

Programmation

Syntaxe :	Descripteur.vld	
Description :	Cette caractéristique n'est accessible qu'en lecture.	
Paramètres :	Descripteur	Nom de registre
Valeur retournée :		Le résultat de l'interrogation peut être :
	FALSE =	valeur non valide
	TRUE =	valeur valide

Exemple

```
IF REG[15].VLD == FALSE    ; Interrogation de la validité de la valeur de registre
  REG[15] = 84
ENDIF
VAR1 = REG[9].VLD          ; Affectation à Var1 de la valeur de l'interrogation de
                           l'état de REG[9].
```

8.3.33 RETURN

Description

La fonction RETURN permet d'annuler le traitement d'un sous-programme en cours et de revenir à l'emplacement de la dernière instruction CALL.

Si RETURN n'est pas configuré dans le sous-programme, le sous-programme est exécuté jusqu'à la fin et l'annulation revient ensuite à l'emplacement d'appel.

Programmation

Syntaxe :	**RETURN**	
Description :	revenir à l'emplacement d'appel	
Paramètres :	- aucun -	

Exemple

```
//B(PROG1)                ; Début du bloc
SUB(UP2)                  ; Début du sous-programme
  IF VAR1.val=="Paul"
    VAR1.val="Jules"
RETURN                    ; Si la valeur de la variable est = Paul, la valeur "Jules"
                          est affectée à la variable et le sous-programme se termine
                          à cet endroit.
  ENDIF
  VAR1.val="Paul"         ; Si la valeur de variable est ≠ Paul, la valeur "Paul" est
                          affectée à la variable.
END_SUB                   ; Fin du sous-programme
//END                     ; Fin du bloc
```

8.3.34 Décompilation

Description

L'aide à la programmation permet de **décompiler** le code CN créé avec la fonction GC et d'afficher de nouveau les valeurs de variable dans le champ de saisie et de visualisation de la boîte de dialogue de saisie correspondante.

Programmation

Les variables à partir du code CN sont reprises dans la boîte de dialogue. Les valeurs de variable à partir du code CN sont comparées aux valeurs de variables calculées à partir du fichier de configuration S'il n'y a pas de concordance, un message d'erreur est consigné dans le journal de bord car les valeurs ont été modifiées dans le code CN généré.

Si une variable existe plusieurs fois en code CN, c'est toujours la dernière occurrence de cette variable qui est exploitée lors de la décompilation. Une alarme est consignée dans le journal de bord.

Les variables qui n'ont pas été utilisées en code CN lors de la génération de code, sont enregistrées comme commentaire utile. Ce commentaire utile permet de décrire toutes les informations nécessaires à la décompilation. Le commentaire utile ne doit pas être modifié.

Remarque

Le bloc du code CN et du commentaire utile peut être décompilé uniquement s'il commence au début d'une ligne.

Exemples :

Dans le programme figure le code CN suivant :

```
DEF VAR1 = (I//101)
OUTPUT(CODE1)
  "X" VAR1 " Y200"
  "X" VAR1 " Y0"
END_OUTPUT
```

Dans le programme pièce, le code suivant est créé :

```
;NCG#TestGC#\cus.dir\aeditor.com#CODE1#1#3#
X101 Y200
X101 Y0
;#END#
```

L'éditeur lit lors de la décompilation :

```
X101 Y200
X222 Y0          ; La valeur pour X a été modifiée dans le programme pièce (X101 →
                 X222)
```

Dans la boîte de dialogue de saisie, la valeur suivante est indiquée pour VAR1 : `VAR1 = 222`

Voir aussi

Generate Code (GC) (Page 157)

8.3.35 Décompilation sans commentaire utile

Description

L'aide à la programmation permet de **décompiler sans commentaire utile** le code CN créé avec la fonction GC et d'afficher de nouveau les valeurs de variable dans le champ de saisie et de visualisation de la boîte de dialogue de saisie correspondante.

Programmation

Pour contenir les lignes de commentaires créées lors de la génération du code normal, l'instruction GC peut être exécutée de la manière suivante :

```
CN("CODE1",D_NAME,1)
```

Le code créé ne peut normalement pas être décompilé. Pour pouvoir ainsi décompiler malgré tout les appels de cycle générés, les étapes suivantes sont nécessaires :

- **Étendre le fichier "easyscreen.ini"**
  Dans le fichier "easyscreen.ini", la section `[RECOMPILE_INFO_FILES]` est introduite. Tous les fichiers ini contenant des descriptions pour les cycles à décompiler sans commentaire utile sont répertoriés dans cette section :
  ```
  [RECOMPILE_INFO_FILES]
  IniFile01 = cycles1.ini
  IniFile02 = cycles2.ini
  ```
  Plusieurs fichiers ini peuvent être indiqués, dont le nom peut être choisi librement.
- **Création d'un fichier Ini pour la description du cycle**
  Enregistrer le fichier ini avec les descriptions de cycles dans le répertoire suivant :
  [Répertoire système user]/cfg
  [Répertoire système oem]/cfg
  [Répertoire système addon]/cfg
  Une section individuelle est nécessaire pour chaque cycle. Le nom de section correspond au nom du cycle :
  ```
  [Cycle123]
  Mname = TestGC
  Dname = testgc.com
  OUTPUT = Code1
  Anzp = 3
  Version = 0
  Code_typ = 1
  Icône = cycle123.png
  Desc_Text = This is describing text
  ```

Mname	Nom du masque
Dname	Nom du fichier dans lequel le masque est défini
OUTPUT	Nom de la méthode OUTPUT concernée
Anzp	Nombre de paramètres du masque décompilé (toutes les variables créées avec DEF, également variables auxiliaires)
Version	(en option) indication de version pour le cycle

Icône	(en option) Icône pour l'affichage dans le programme de chaîne logistique, format *.png Taille d'image pour résolution correspondante : 640 x 480 mm → 16 x 16 pixels 800 x 600 mm → 20 x 20 pixels 1024 x 768 mm → 26 x 26 pixels 1280 x 1 024 mm → 26 x 26 pixels 1280 x 768 mm → 26 x 26 pixels Archivage : [Répertoire système user]/ico/ico<résolution> **Remarques :** • Pour les résolutions 1280, le dossier est utilisé pour 1024 x 768 mm (adapté uniquement pour les programmes de chaîne logistique). • La taille de l'image ne dépend pas uniquement de la résolution mais également de la taille de police réglée dans l'éditeur.
Desc_Text	(en option) Texte explicatif pour l'affichage dans le programme de chaîne logistique, max. 17 signes longueur de string (adapté uniquement pour les programmes de chaîne logistique)
Param_Text	(en option) Texte pour la liste de paramètres T=%1 D=%2

Remarque

Remarques relatives à Icon, Desc_text et Param_Text :

1. L'icône de l'utilisateur est toujours affichée dans l'éditeur de code G. L'icône en code G s'affiche toujours dans l'éditeur ShopMill/ShopTurn. Ceci permet de mieux différentier les étapes ShopMill/ShopTurn des étapes non ShopMill/ShopTurn.
2. L'étape Run MyScreens dépend du réglage de "Afficher les cycles en tant qu'étape de travail" dans l'éditeur. Ceci est vrai pour le code G et pour l'éditeur JobShop
   Ainsi, si "Afficher les cycles en tant qu'étape de travail" = "OUI", le "Desc_Text" de la configuration Run MyScreens sera utilisé, par contre si la valeur est "NON", le cycle sera affiché.
3. Desc_Text et Param_Text
   – peuvent être indiqué en fonction de la langue avec la notation $xxxxx
   – via %1,%2, ... il est possible d'accéder aux paramètres de l'appel de cycle généré. Le caractère générique est alors remplacé par le paramètre. Dans la liste de paramètres générée, le paramètre se trouve derrière le '%'.

Exemple

```
//M(TestGC/"Génération de code :")
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)                          ; Ecriture du code CN dans le fichier
                                              \MPF.DIR\MESSEN.MPF à partir de la mé-
                                              thode OUTPUT :
                                              Cycle123(123, -6)
                                              M30

END_PRESS
```

Autres conditions à prendre en compte

- La fonctionnalité "Décompilation sans commentaire utile" ne dispose pas de toutes les fonctionnalités de la "Décompilation avec commentaire utile".
  Lors d'une "décompilation sans commentaire utile", les appels cycliques typiques tels que MYCYCLE(PAR1, PAR2, PAR3...) sont pris en charge. Cependant, aucun commentaire utile ne peut figurer dans la ligne d'appel de fonction. Les paramètres facultatifs qui ne sont pas transmis lors de l'appel de fonction et qui sont de type String S doivent toutefois être au moins indiqués par des guillemets vides, par exemple**.** "". Sinon "Run MyScreens" essaie de remplir ces paramètres avec des virgules pour ensuite décompiler "l'appel cyclique rempli".
- Les paramètres de type String ne doivent contenir aucune virgule et aucun point-virgule dans la chaîne de caractères à transmettre.
- Lors d'une "décompilation sans commentaire utile", toutes les variables contenues dans la méthode OUTPUT doivent figurer entre parenthèses afin que la fonctionnalité "remplir les paramètres cycliques manquants avec des virgules" puisse prendre effet.

Exemple :

Autorisé :

```
OUTPUT
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")"
END_OUTPUT
```

Non autorisé (la variable MYCOMMENT est placée après la parenthèse de fermeture) :

```
OUTPUT
```

```
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")" MYCOMMENT
END_OUTPUT
```

8.3.36 Search Forward, Search Backward (SF, SB)

Description

La fonction **SF, SB (Search Forward, Search Backward)** permet de rechercher dans le programme CN actuel de l'éditeur une chaîne (string) à partir de la position actuelle du curseur. La valeur de cette chaîne est ensuite éditée.

Programmation

Syntaxe :	**SF**("Chaîne")	
Désignation :	**S**earch **F**orward : recherche vers le bas à partir de la position actuelle du curseur	
Syntaxe :	**SB**("Chaîne")	
Désignation :	**S**earch **B**ackward : recherche vers le haut à partir de la position actuelle du curseur	
Paramètres :	string	texte à rechercher

Règles lors de la recherche

- Dans le programme CN actuel, il faut placer un espace avant et après l'unité à partir de laquelle la chaîne doit être recherchée ainsi que sa valeur.
- Le terme n'est pas recherché dans les commentaires ni au sein d'une chaîne.
- La valeur à éditer doit être numérique car les expressions de type "X1=4+5" ne sont pas reconnues.
- Les constantes hexadécimales de type X1='HFFFF', les constantes binaires de type X1='B10010' et les constantes exponentielles de type X1='-.5EX-4' sont reconnues.
- La valeur d'une chaîne peut être éditée à condition qu'entre la chaîne et la valeur figure :
  - rien
  - un espace
  - un signe d'égalité (=)

Exemple

Les notations suivantes sont possibles :

```
X100 Y200               ; La variable Abc obtient la valeur 200
Abc = SB("Y")
X100 Y 200              ; La variable Abc obtient la valeur 200
Abc = SB("Y")
```

```
X100 Y=200              ; La variable Abc obtient la valeur 200
Abc = SB("Y")
```

8.3.37 SL_HMI_INSTALL_DIR

Description

La fonction SL_HMI_INSTALL_DIR renvoie le chemin du répertoire d'installation de SINUMERIK Operate.

Programmation

Syntaxe :	SL_HMI_INSTALL_DIR()	
Description :	Déterminer le répertoire d'installation de SINUMERIK Operate	
Paramètres :	- aucun -	

8.3.38 SL_TEMP_DIR

Description

La fonction SL_TEMP_DIR renvoie un chemin pour les fichiers temporaires dans SINUMERIK Operate.

Remarque

Le contenu de ce répertoire n'est pas persistant, c'est-à-dire qu'il est supprimé à chaque démarrage de SINUMERIK Operate.

Programmation

Syntaxe :	SL_TEMP_DIR()	
Description :	Déterminer le chemin du répertoire non persistant (volatile) de SINUMERIK Operate	
Paramètres :	- aucun -	

8.3.39 Fonctions STRING

Vue d'ensemble

Les fonctions suivantes permettent les traitements des chaînes de caractère :

- Détermination des longueurs de chaînes
- Recherche d'un caractère dans une chaîne
- Extraire une partie de chaîne depuis la gauche
- Extraire une partie de chaîne depuis la droite
- Extraire une partie de chaîne à partir du milieu de chaîne
- Remplacement de parties de chaîne
- Comparaison de chaînes de caractères
- Insertion d'une chaîne dans une chaîne
- Suppression d'une chaîne dans une chaîne
- Suppression d'espaces (depuis la gauche ou la droite)
- Insertion de valeurs ou de chaînes avec identifiants de mise en forme

Fonction LEN : longueur d'une chaîne de caractères

Syntaxe :	**LEN**(string | varname)	
Description :	Déterminer le nombre de caractères d'une chaîne	
Paramètres :	string	Toute expression de chaîne valide. Pour une chaîne vide, NULL est retourné.
	varname	Tout nom de variable valide et déclaré
	Seul un des deux paramètres possibles est autorisé.	

Exemple

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="BONJOUR"
  VAR02=LEN(VAR01)          ; Résultat = 5
END_LOAD
```

Fonction INSTR : rechercher des caractères dans une chaîne

Syntaxe :	**INSTR**(Start, String1, String2 [,sens])
Description :	Rechercher des caractères

Paramètres :	Start	La position de départ, à partir de laquelle la recherche de string1 s'effectue dans string2. Si la recherche doit commencer au début de string2, il convient de saisir 0.
	String1	Caractère recherché.
	String2	Chaîne de caractères dans laquelle la recherche est effectuée
	Sens (facultatif)	Sens dans lequel la recherche est effectuée 0 : de gauche à droite (préréglage) 1 : de droite à gauche
	Si string1 n'est pas contenu dans string2, le résultat retourné est 0.	

Exemple

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="BONJOUR/MONDE"
  VAR02=INST(1,"/",VAR01)        ; Résultat = 6
END_LOAD
```

Fonction LEFT : traiter la chaîne à partir de la gauche

Syntaxe :	**LEFT**(string, longueur)	
Description :	LEFT retourne une chaîne de caractères qui doit contenir le nombre de caractères indiqué en partant de la gauche d'une chaîne.	
Paramètres :	string	Chaîne de caractères ou variable avec la chaîne de caractères à traiter
	longueur	Nombre de caractères à lire (extraire)

Exemple

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="BONJOUR/MONDE"
  VAR02=LEFT(VAR01,5)           ; Résultat = "BONJOUR"
END_LOAD
```

Fonction RIGHT : traiter la chaîne à partir de la droite

Syntaxe :	**RIGHT**(string, longueur)	
Description :	RIGHT retourne une chaîne de caractères qui doit contenir le nombre de caractères indiqué en partant de la droite d'une chaîne.	
Paramètres :	string	Chaîne de caractères ou variable avec la chaîne de caractères à traiter
	longueur	Nombre de caractères à lire (extraire)

Exemple

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="BONJOUR/MONDE"
  VAR02=LEFT(VAR01,4)         ;Résultat = "MONDE"
END_LOAD
```

Fonction MIDS : traiter la chaîne à partir du milieu

Syntaxe :	**MIDS**(string, start [, longueur])	
Description :	MIDS retourne une chaîne de caractères qui doit contenir le nombre de caractères indiqué à partir du point indiqué d'une chaîne.	
Paramètres :	string	Chaîne de caractères ou variable avec la chaîne de caractères à traiter
	start	Point à partir duquel une chaîne de caractères doit être lue
	longueur	Nombre de caractères à lire (extraire)

Exemple

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="BONJOUR/MONDE"
  VAR02=LEFT(VAR01,4,4)           ; Résultat = "UR/M"
END_LOAD
```

Fonction REPLACE : remplacement de caractères

Syntaxe :	**REPLACE**(string, FindString, ReplaceString [, start [, count]])
Description :	La fonction REPLACE permet de remplacer un caractère / une chaîne de caractères dans une chaîne par un autre caractère / une autre chaîne de caractères.

Paramètres :	string	Chaîne, dans laquelle FindString doit être remplacé par ReplaceString.
	FindString	Chaîne à remplacer
	ReplaceString	Chaîne de remplacement (remplace FindString)
	start	Position de départ à partir de laquelle sont effectués la recherche et le remplacement
	count	Nombre de caractères dans lesquels la recherche de FindString doit être effectuée à partir de la position de départ
Valeur retournée :		
	string = chaîne vide	Copie de String
	FindString = chaîne vide	Copie de String
	ReplaceString = chaîne vide	Copie de String, dans laquelle toutes les occurrences de FindString sont supprimées
	start > Len(String)	Chaîne vide
	count = 0	Copie de String

Fonction STRCMP : comparer des chaînes de caractères

Syntaxe :	**STRCMP**(string1, string2 [, CaseInsensitive])	
Description :	STRCMP compare une chaîne de caractères à une autre.	
Paramètres :	string1	Chaînes de caractères à comparer avec une deuxième chaîne de caractères
	string2	Chaîne de caractères à comparer avec la première chaîne de caractères
	CaseInsensitive	Comparer en tenant compte / sans tenir compte de la casse : sans indication ou FALSE = la casse est respectée TRUE = la casse n'est pas respectée

Exemple

```
REG[0]=STRCMP("Hugo", "HUGO")              ; Résultat = 32
                                           ; (<> 0, les chaînes ne sont pas identiques)

REG[0]=STRCMP("Hugo", "HUGO", TRUE)        ; Résultat = 0
                                           ; (== 0, les chaînes sont identiques
```

Fonction STRINSERT : insérer une chaîne de caractères

Syntaxe :	**STRINSERT**(string1, string2 , insert position)
Description :	STRINSERT insère une chaîne de caractères à une certaine position dans une chaîne de caractères.

Paramètres :	string1	Chaîne de caractères dans laquelle une chaîne de caractères doit être insérée
	string2	Chaîne de caractères à insérer
	insert position	Position à laquelle la chaîne de caractères doit être insérée

Exemple

```
REG[0]=STRINSERT("Hello!", " world", 5)    ; Résultat = "Hello world!")
```

Fonction STRREMOVE : supprimer une chaîne de caractères

Syntaxe :	**STRREMOVE**(string1, remove positon , count)	
Description :	STRREMOVE supprime un certain nombre de caractères à une certaine position dans une chaîne de caractères.	
Paramètres :	string1	Chaîne de caractères dans laquelle des caractères doivent être supprimés
	remove position	Position à partir de laquelle des caractères doivent être supprimés de la chaîne de caractères
	count	Nombre de caractères à supprimer

Exemple

```
REG[0]=STRREMOVE("Hello world!", 5, 6)     ; Résultat = "Hello!")
```

Fonction TRIMLEFT : supprimer les espaces à gauche de la chaîne de caractères

Syntaxe :	**TRIMLEFT**(string1)	
Description :	TRIMLEFT supprime les espaces à gauche de la chaîne de caractères.	
Paramètres :	string1	Chaîne de caractères dans laquelle les espaces doivent être supprimés à gauche de la chaîne de caractères

Exemple

```
REG[0]=TRIMLEFT("    Hello!")              ; Résultat = "Hello!")
```

Fonction TRIMRIGHT : supprimer les espaces à droite de la chaîne de caractères

Syntaxe :	**TRIMRIGHT**(string1)	
Description :	TRIMRIGHT supprime les espaces à droite de la chaîne de caractères.	
Paramètres :	string1	Chaîne de caractères dans laquelle les espaces doivent être supprimés à droite de la chaîne de caractères

Exemple

```
REG[0]=TRIMRIGHT("Hello!    ")            ; Résultat = "Hello!")
```

Fonction FORMAT : insérer des valeurs ou une chaîne de caractères avec identifiant de mise en forme

Syntaxe :	**FORMAT**(Texte avec identifiant de mise en forme [, valeur1] ... [, valeur28])	
Description :	La fonction FORMAT permet d'insérer jusqu'à 28 valeurs ou chaînes de caractères à des certains endroits dans un texte spécifié en utilisant des identifiants de mise en forme.	
Identifiants de mise en forme :	Syntaxe	%[Mémentos] [Largeur] [.Chiffres après la virgule] Type
	Mémentos	Caractère facultatif pour définir la mise en forme d'édition : • Justifié à gauche ou à droite ("-" pour justifié à gauche) • Ajouter des zéros de tête ("0") • Par défaut : remplir avec des espaces si la valeur à éditer possède moins de caractères que ce qui a été indiqué avec [Largeur]
	Largeur	L'argument définit la largeur minimale d'affichage d'un nombre non négatif. Si la valeur possède moins de caractères que ce qui a été défini par l'argument, les caractères manquants sont remplis par des espaces.
	Chiffres après la virgule	Pour un nombre à virgule flottante, ce paramètre facultatif définit le nombre de chiffres après la virgule.
	Type	Le caractère de type définit les formats de données qui sont transmis à l'instruction d'impression. Ces caractères doivent être indiqués. • d : valeur entière • f : nombre à virgule flottante • s : chaîne de caractères • o : octal • x : hexadécimal • b : binaire

Exemple

```
DEF VAR1
DEF VAR2
LOAD
VAR1 = 123
VAR2 = FORMAT("Hello %08b %.2f %s!", VAR1 + 1, 987.654321, "world")
                                    ; Résultat = "Hello 01111100 987.65 world!"
END_LOAD
```

Voir aussi

Utilisation de chaînes de caractères (Page 106)

8.3.40 Boucles WHILE/UNTIL

Description

Les instructions DO-LOOP permettent de réaliser une boucle. Selon la configuration, celle-ci est exécutée tant qu'une condition est remplie (WHILE) ou jusqu'à ce qu'une condition soit remplie (UNTIL).

Les boucles pouvant entraver les performances du système selon la configuration, il convient de les utiliser avec circonspection et de renoncer aux actions chronophages dans les boucles.

Il est recommandé d'utiliser p. ex. un registre (REG[]) comme variable d'exécution, car les variables d'affichage normales (en particulier celles avec une connexion à des variables système ou utilisateur) peuvent également entraver les performances du système en raison de l'extrême fréquence des mises à jour et des processus d'écriture.

La fonction DEBUG (voir chapitre DEBUG (Page 148)) permet de déterminer le temps d'exécution des méthodes "Run MyScreens". Cela permet d'identifier, le cas échéant, les problèmes générés par les boucles (charge CPU élevée, réactivité réduite).

Remarque

Etant donné que chaque boucle FOR peut être remplacée par une boucle WHILE, la syntaxe de formulation d'une boucle FOR n'est pas prise en charge dans EasyScreen.

Programmation

```
DO
   <Instructions>
LOOP_WHILE <Condition de poursuite de la boucle>
```

```
DO
   <Instructions>
LOOP_UNTIL <Condition d'arrêt de la boucle>

DO_WHILE <Condition de poursuite de la boucle>
   <Instructions>
LOOP

DO_UNTIL <Condition d'arrêt de la boucle>
   <Instructions>
LOOP
```

Exemple

```
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 Exécution cyclique de scripts : START_TIMER, STOP_TIMER

Description

Des méthodes SUB peuvent être appelées de manière cyclique à l'aide de temporisations. Les fonctions START_TIMER() et STOP_TIMER() servent à cette fin.

Remarque

Une seule temporisation peut être configurée par méthode SUB.

Programmation

Syntaxe :	**START_TIMER**("nom SUB", intervalle)	
Description :	Lancer l'exécution cyclique d'une méthode SUB	
Paramètres :	Nom SUB :	Nom de la méthode SUB à appeler de manière cyclique
	Intervalle :	Intervalle en millisecondes

Syntaxe :	**STOP_TIMER**("nom SUB")	
Description :	Arrêter l'exécution cyclique d'une méthode SUB	
Paramètres :	Nom SUB :	Nom de la méthode SUB dont la temporisation doit être arrêtée

Exemple

```
//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)
   ; Appelle la SUB "MyTimerSub" toutes les 1000 millisecondes
   START_TIMER("MyTimerSub", 1000)
END_PRESS

PRESS(VS2)
   STOP_TIMER("MyTimerSub")
END_PRESS
```

Si START_TIMER est de nouveau appelée pour une temporisation déjà affectée à une méthode SUB, le nouvel intervalle est appliqué s'il est différent du précédent. Autrement, ce deuxième appel est ignoré.

Le système impose une longueur de 100 millisecondes pour le plus petit intervalle.

Si STOP_TIMER est appelée pour une méthode SUB pour laquelle aucune temporisation n'est actuellement en cours, cet appel est ignoré.

Eléments graphiques et logiques

9

9.1 Trait, ligne de séparation, rectangle, cercle et ellipse

9.1.1 Définition des éléments graphiques

Description

Les traits, lignes de séparation, rectangles et ellipses sont configurés dans la méthode LOAD :

- Les rectangles transparents sont obtenus en définissant la couleur de remplissage sur la couleur d'arrière-plan du système.
- Avec l'élément ELLIPSE, on obtient un cercle en réglant la hauteur et la largeur sur une valeur identique.
- Les lignes de séparation horizontales et verticales ont toujours exactement la même largeur ou hauteur de fenêtre. Les lignes de séparation ne provoquent pas l'apparition de barres de défilement.
  Si jusqu'ici vous avez utilisé l'élément RECT pour représenter des lignes de séparation, p. ex. RECT(305,0,1,370,0,0,1), celui-ci est reconnu par l'aide à la programmation et converti automatiquement en V_SEPARATOR(305,1,"#87a5cd", 1). Cela s'applique aussi bien aux lignes de séparation horizontales que verticales.

Elément LINE

Programmation :

Syntaxe :	LINE (x1, y1, x2, y2, color, style, tag, visible)
Description :	Définition de trait

Paramètres :	x1	Coordonnée x du point de départ
	y1	Coordonnée y du point de départ
	x2	Coordonnée x du point final
	y2	Coordonnée y du point final
	color	Couleur du trait
	style	Style de trait : 1 = ininterrompu 2 = interrompu 3 = pointillé 4 = en trait mixte
	tag	Ce paramètre tag peut être utilisé pour afficher/masquer et supprimer un élément graphique ou un groupe d'éléments graphiques avec les fonctions SHOW_GRAPHIC, HIDE_GRAPHIC et DELETE_GRAPHIC.
	visible	Élément graphique visible (TRUE/FALSE)

Elément RECT

Programmation :

Syntaxe :	RECT (x, y, w, h, frame color, fill color, style, tag, visible)	
Description :	Définition de rectangle	
Paramètres :	x	Coordonnée x gauche/haut
	y	Coordonnée y gauche/haut
	w	Largeur
	h	Hauteur
	frame color	Couleur du cadre
	fill color	Couleur de remplissage
	style	Style du cadre : 1 = ininterrompu 2 = interrompu 3 = pointillé 4 = en trait mixte
	tag	Ce paramètre tag peut être utilisé pour afficher/masquer et supprimer un élément graphique ou un groupe d'éléments graphiques avec les fonctions SHOW_GRAPHIC, HIDE_GRAPHIC et DELETE_GRAPHIC.
	visible	Élément graphique visible (TRUE/FALSE)

Elément ELLIPSE

Programmation :

Syntaxe :	ELLIPSE(x, y, w, h, frame color, fill color, style, tag, visible)
Description :	Définir un cercle ou une ellipse

Paramètres :	x	Coordonnée x gauche/haut
	y	Coordonnée y gauche/haut
	w	Largeur
	h	Hauteur
	frame color	Couleur du cadre
	fill color	Couleur de remplissage
	style	Style du cadre : 1 = ininterrompu 2 = interrompu 3 = pointillé 4 = en trait mixte
	tag	Ce paramètre tag peut être utilisé pour afficher/masquer et supprimer un élément graphique ou un groupe d'éléments graphiques avec les fonctions SHOW_GRAPHIC, HIDE_GRAPHIC et DELETE_GRAPHIC.
	visible	Élément graphique visible (TRUE/FALSE)

Si la valeur de la largeur et celle de la hauteur sont identiques, on obtient un cercle.

Elément V_SEPARATOR

Programmation :

Syntaxe :	V_SEPARATOR (x,w,color,pen)	
Description :	Définir une ligne de séparation verticale	
Paramètres :	x	Position X
	pen width	Epaisseur du trait
	color	Couleur
	style	Style de trait : 1 = ininterrompu 2 = interrompu 3 = pointillé 4 = en trait mixte
	tag	Ce paramètre tag peut être utilisé pour afficher/masquer et supprimer un élément graphique ou un groupe d'éléments graphiques avec les fonctions SHOW_GRAPHIC, HIDE_GRAPHIC et DELETE_GRAPHIC.
	visible	Élément graphique visible (TRUE/FALSE)

Elément H_SEPARATOR

Programmation :

Syntaxe :	H_SEPARATOR (y,h,color,pen)
Description :	Définir une ligne de séparation horizontale

Paramètres :	y	Position Y
	pen width	Epaisseur du trait
	color	Couleur
	style	Style de trait : 1 = ininterrompu 2 = interrompu 3 = pointillé 4 = en trait mixte
	tag	Ce paramètre tag peut être utilisé pour afficher/masquer et supprimer un élément graphique ou un groupe d'éléments graphiques avec les fonctions SHOW_GRAPHIC, HIDE_GRAPHIC et DELETE_GRAPHIC.
	visible	Élément graphique visible (TRUE/FALSE)

Plus d'informations

Les fonctions SHOW_GRAPHIC, HIDE_GRAPHIC, CLEAR_BACKGROUND et DELETE_GRAPHIC sont décrites dans la section Fonctions graphiques (Page 196). Ces fonctions permettent de commander l'affichage des éléments graphiques.

9.1.2 Fonctions graphiques

Vue d'ensemble

Les fonctions graphiques suivantes sont disponibles :

- CLEAR_BACKGROUND
- DELETE_GRAPHIC
- HIDE_GRAPHIC
- SHOW_GRAPHIC

Les fonctions peuvent être appliquées aux éléments graphiques LINE, RECT, ELLIPSE, V_SEPARATOR et H_SEPARATOR (Page 193).

Fonction CLEAR_BACKGROUND : Suppression des éléments graphiques

La fonction CLEAR_BACKGROUND permet de supprimer les éléments graphiques.

Syntaxe :	CLEAR_BACKGROUND()	
Description :	Suppression des éléments graphiques. La mémoire occupée par les objets graphiques est libérée.	
Paramètres :	- aucun -	

Fonction DELETE_GRAPHIC : Suppression d'un élément graphique ou d'un groupe d'éléments graphiques

La fonction DELETE_GRAPHIC permet de supprimer un élément graphique spécifique ou un groupe d'éléments graphiques. Pour les éléments graphiques correspondants, une valeur doit être définie dans le paramètre "tag".

- Par rapport à la fonction CLEAR_BACKGROUND, il est possible de supprimer des éléments graphiques spécifiques.
- Par rapport à la fonction HIDE_GRAPHIC, la mémoire occupée par les objets graphiques est libérée.

Syntaxe :	**DELETE_GRAPHIC**(tag)	
Description :	Suppression d'un élément graphique ou d'un groupe d'éléments graphiques dont la valeur du paramètre "tag" est identique.	
Paramètres :	tag	Le paramètre "tag" peut être défini dans les éléments graphiques. Il est également possible de définir plusieurs éléments graphiques avec la même valeur et de former ainsi un groupe d'éléments graphiques.

Exemple

```
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)                                    ;les 3 éléments graphi-
                                                     ques de type LINE avec
                                                     tag=5 sont supprimés
```

Fonction HIDE_GRAPHIC : Masquage d'un élément graphique ou d'un groupe d'éléments graphiques

La fonction HIDE_GRAPHIC permet de masquer un élément graphique spécifique ou un groupe d'éléments graphiques. Pour les éléments graphiques correspondants, une valeur doit être définie dans le paramètre "tag".

Syntaxe :	HIDE_GRAPHIC(tag)	
Description :	Masquage d'un élément graphique ou d'un groupe d'éléments graphiques dont la valeur du paramètre "tag" est identique.	
Paramètres :	tag	Le paramètre "tag" peut être défini dans les éléments graphiques. Il est également possible de définir plusieurs éléments graphiques avec la même valeur et de former ainsi un groupe d'éléments graphiques.

Exemple

```
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)                                          ;les 3 éléments graphi-
                                                         ques de type LINE avec
                                                         tag=5 sont masqués
```

Fonction SHOW_GRAPHIC : Affichage d'un élément graphique ou d'un groupe d'éléments graphiques

La fonction SHOW_GRAPHIC permet d'afficher un élément graphique spécifique ou un groupe d'éléments graphiques. Pour les éléments graphiques correspondants, une valeur doit être définie dans le paramètre "tag".

Syntaxe :	SHOW_GRAPHIC(tag)	
Description :	Affichage d'un élément graphique ou d'un groupe d'éléments graphiques dont la valeur du paramètre "tag" est identique.	
Paramètres :	tag	Le paramètre "tag" peut être défini dans les éléments graphiques. Il est également possible de définir plusieurs éléments graphiques avec la même valeur et de former ainsi un groupe d'éléments graphiques.

Exemple

```
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)                                          ;les 3 éléments graphi-
                                                         ques de type LINE avec
                                                         tag=5 sont affichés
```

9.2 Définition d'un array

Définition

Un tableau permet de sauvegarder des données de type standard dans la mémoire de manière à pouvoir y accéder à l'aide d'un index.

Description

Les tableaux peuvent être unidimensionnels ou bidimensionnels. Un tableau unidimensionnel est considéré comme un tableau bidimensionnel avec une ligne ou une colonne.

Les tableaux sont définis avec l'identifiant //A et se terminent par //END. Le nombre de lignes et de colonnes est librement paramétrable. Un tableau possède la structure suivante :

Programmation

Syntaxe :	**//A**(Descripteur) (a/b...) (c/d...) ... //END	
Description :	Définir le tableau	
Paramètres :	Descripteur	Nom du tableau
	a, b, c, d	Valeur du tableau Les valeurs de type STRING doivent être indiquées entre guillemets (").

Exemple

```
//A(Filetage)                    ; Dimensions/Pas du filetage/Diamètre à fond de filet
(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 Accéder à la valeur d'un élément de l'array

Description

La propriété valeur (descripteur.val) permet de transmettre la valeur d'un accès à un tableau.

Toujours commencer l'indice de ligne (numéro de ligne du tableau) et l'indice de colonne (numéro de colonne du tableau) à la valeur 0. Si un indice de ligne ou indice de colonne pointe

en dehors du tableau, la valeur 0 ou une chaîne de caractères vide s'affiche et la variable ERR a la valeur TRUE. La variable ERR est également TRUE lorsque le terme recherché n'a pas été trouvé.

Programmation

Syntaxe :	Descripteur [Z,[M[,C]]].val ou Descripteur [Z,[M[,C]]]		
Description :	Accès à un tableau unidimensionnel avec une seule colonne		
Syntaxe :	Descripteur [S,[M[,C]]].val ou Descripteur [S,[M[,C]]]		
Description :	Accès à un tableau unidimensionnel avec une seule ligne		
Syntaxe :	Descripteur [Z,S,[M[,C]]].val ou Descripteur [Z,S,[M[,C]]]		
Description :	Accès à un tableau bidimensionnel		
Paramètres :	Descripteur :	Nom du tableau	
	D :	Valeur de ligne (indice de ligne ou terme recherché)	
	S :	Valeur de colonne (indice de colonne ou terme recherché)	
	M :	Mode d'accès	
		0	direct
		1	recherche par ligne, accès direct à la colonne
		2	accès direct à la ligne, recherche par colonne
		3	recherche
		4	recherche de l'indice de ligne
		5	recherche de l'indice de colonne
	C :	Mode de comparaison	
		0	le terme recherché doit être situé dans la plage de valeurs de la ligne ou de la colonne
		1	le terme recherché doit être trouvé de façon exacte

Exemple :	`VAR1 = MET_G[REG[3],1,0].VAL`	`;Affecter à Var1 une valeur à partir du tableau MET_G`

Mode d'accès

- **Mode d'accès "direct"**
  En mode d'accès "direct" (M = 0), l'accès est effectué sur le tableau avec l'indice de ligne dans Z et l'indice de colonne dans S. Le mode de comparaison C n'est pas évalué.
- **Mode d'accès "recherche"**
  En mode d'accès M = 1, 2 ou 3, la recherche s'effectue toujours dans la ligne 0 ou la colonne 0.

Mode M	Valeur de ligne Z	Valeur de colonne S	Valeur de sortie
0	Indice de ligne	Indice de colonne	Valeur de la ligne Z et de la colonne S
1	Terme recherché : recherche dans la colonne 0	Indice de la colonne à partir de laquelle la valeur est lue	Valeur de la ligne trouvée et de la colonne S

Mode M	Valeur de ligne Z	Valeur de colonne S	Valeur de sortie
2	Indice de la ligne à partir de laquelle la valeur de retour est lue	Terme recherché : recherche dans la ligne 0	Valeur de la ligne Z et de la colonne trouvée
3	Terme recherché : recherche dans la colonne 0	Terme recherché : recherche dans la ligne 0	Valeur de la ligne trouvée et de la colonne trouvée
4	Terme recherché : recherche dans la colonne S	Indice de la colonne dans laquelle la recherché est effectuée	Indice de ligne
5	Indice de la ligne dans laquelle la recherché est effectuée.	Terme recherché : recherche dans la ligne Z	Indice de colonne

Mode de comparaison

En utilisant le mode de comparaison C = 0, le contenu de la ligne de recherche ou de la colonne de recherche est trié par ordre croissant. Si le terme de recherche est inférieur au premier élément ou supérieur au dernier, la recherche fournit la valeur 0 ou un string vide et la variable d'erreur ERR est TRUE.

En utilisant le mode de comparaison C = 1, le terme recherché doit être trouvé dans la ligne de recherche ou la colonne de recherche. Si le terme recherché n'est pas trouvé, la recherche fournit la valeur 0 ou un string vide et la variable d'erreur ERR est TRUE.

9.2.2 Exemple : Accès à un élément de l'array

Condition préalable

Deux arrays sont définis ci-dessous qui sont la condition pour les exemples suivants :

```
//A(Filetage)
                (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
```

Exemples

- **Mode d'accès, exemple 1 :**
  Le terme recherché se trouve dans Z. Ce terme est toujours recherché dans la colonne 0. La valeur de la colonne S est indiquée avec l'indice de ligne du terme trouvé :
  ```
  VAR1 = Filetage[0.5,1,1]   ;VAR1 vaut 0.125
  ```
  Signification des paramètres :
  Recherche de la valeur 0.5 dans la colonne 0 de l'array "Filet" et indication de la valeur trouvée dans la colonne 1 de la même ligne.
- **Mode d'accès, exemple 2 :**
  Le terme recherché se trouve dans S. Ce terme est toujours recherché dans la ligne 0. La valeur de la ligne Z est indiquée avec l'indice de colonne du terme trouvé :
  ```
  VAR1 = ARRAY2[3,"STG",2]   ;VAR1 vaut 0.125
  ```
  Signification des paramètres :
  Recherche de la colonne avec le contenu "STG" dans la ligne 0 de l'array "Array2". Indication de la valeur à partir de la colonne trouvée et de la ligne avec l'indice 3.
- **Mode d'accès, exemple 3 :**
  Z et S contiennent chacun un terme recherché. La recherche est effectuée dans la colonne 0 de l'indice de ligne pour le terme situé dans Z et dans la colonne 0 de l'indice de colonne pour le terme situé dans S. La valeur de l'array est indiquée avec l'indice de ligne et l'indice de colonne trouvés :
  ```
  VAR1 = ARRAY2[0.6,"STG",3]   ;VAR1 vaut 0.15
  ```
  Signification des paramètres :
  Recherche de la ligne avec le contenu 0.6 dans la colonne 0 de l'array "Array2", recherche de la colonne avec le contenu "STG" dans la ligne 0 de l'array2. Indication de la valeur à partir de la ligne et de la colonne trouvées d'après VAR1.

- **Mode d'accès, exemple 4 :**
  Le terme recherché se trouve dans Z. L'indice de colonne dans laquelle la recherche est effectuée, est situé dans S. L'indice de ligne du terme trouvé est indiqué :
  ```
  VAR1 = Filetage[0.125,1,4]   ;VAR1 vaut 2
  ```
  Signification des paramètres :
  Recherche de la valeur 0.125 dans la colonne 1 de l'array "Filet" et indication de l'indice de ligne de la valeur trouvée selon VAR1.
- **Mode d'accès, exemple 5 :**
  L'indice de ligne dans laquelle la recherché est effectuée, est situé dans Z. Le terme recherché est situé dans S. L'indice de colonne du terme trouvé est indiqué :
  ```
  VAR1 = Filetage[4,0.2,5,1]  ;VAR1 vaut 1
  ```
  Signification des paramètres :
  Recherche de la valeur 0.2 dans la ligne 4 de l'array "Filet" et indication de l'indice de colonne de la valeur trouvée selon VAR1. Le mode de comparaison 1 a été sélectionné car les valeurs de la ligne 4 de sont pas triées par ordre croissant.

9.2.3 Interrogation de l'état d'un élément de l'array

Description

La caractéristique Etat permet de demander si un accès à un tableau fournit une valeur valide.

Programmation

Syntaxe :	Descripteur [Z, S, [M[,C]]]**.vld**
Description :	Cette caractéristique n'est accessible qu'en lecture.
Paramètres :	Descripteur Nom du tableau
Valeur retournée :	FALSE = valeur non valide
	TRUE = valeur valide

Exemple

```
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 Description d'un tableau (grid)

Définition

Les valeurs d'un tableau de type grille (grid) sont actualisées en continu, contrairement à un tableau de type array. Il s'agit d'une représentation sous forme de tableau de valeurs, qui peuvent par exemple provenir d'une CN ou d'un AP. Le tableau est organisé en fonction des colonnes et contient donc toujours des variables de même type dans chaque colonne.

Affectation

Le renvoi à une description de tableau est défini dans la description de la variable :

- La définition de variable détermine les valeurs à afficher et la définition des éléments du tableau détermine l'apparence et l'agencement à l'écran. Le tableau reprend les propriétés du champ de saisie et de visualisation à partir de la ligne de définition de la variable.
- La zone visible du tableau est définie par la largeur et la hauteur du champ de saisie et de visualisation. Lorsque le nombre de lignes ou de colonnes dépasse la capacité de visualisation de la zone visible, un défilement vertical ou horizontal est possible.

Descripteur de tableau

Des descripteurs d'un tableau de mêmes valeurs de NCK ou AP qui peuvent être adressées via un module d'un canal. Le descripteur de tableau est différencié des valeurs limites ou du champ Toggle par le préfixe %. Le descripteur de tableau peut être suivi d'un nom de fichier séparé par une virgule, qui indique le fichier dans lequel la description de tableau est définie.

Descripteur d'un tableau de valeurs de même type, p. ex. à partir du NCK ou de l'AP. Le descripteur de tableau est indiqué à la position à laquelle l'indication de valeur limite ou de champ bascule est configurée. Le descripteur de tableau est introduit par un signe % en préfixe. Ensuite, un nom de fichier peut suivre, séparé par une virgule. Ce nom indique le fichier dans lequel est définie la description du tableau. Par défaut, le tableau est recherché dans le fichier de configuration où le masque lui-même est configuré.

Une description du tableau peut être chargée de manière dynamique, p. ex. dans la méthode LOAD, au moyen de la méthode Load Grid (LG).

Variable système ou utilisateur

Ce paramètre reste vide pour les tableaux, car les variables à afficher sont détaillées dans les lignes de définition de colonne. La description de tableau peut être effectuée de façon dynamique.

Exemple

Définition d'une variable MyGridVar qui représente une Grid "MyGrid1" dans son champ de saisie et de visualisation, (marge à gauche : 100, marge en haut : par défaut, largeur : 350, hauteur : 100).

```
DEF MyGridVar=(V/% MyGrid1///////100,,350,100)
```

Voir aussi

Paramètres de variables (Page 93)

Load Grid (LG) (Page 165)

9.3.1 Définition d'un tableau (grid)

Description

Le bloc de tableau se compose de :

- Description de l'en-tête
- 1 à n descriptions de colonne

Programmation

Syntaxe :	**//G**(Descripteur de tableau / Type de tableau / Nombre de lignes / [Attribut ligne fixe],[Attribut colonne fixe])		
Description :	Définition d'un tableau (grid)		
Paramètres :	Descripteur de tableau	Le descripteur de tableau est utilisé ici sans préfixe %. Il ne peut être utilisé qu'une seule fois dans une boîte de dialogue.	
	Type de tableau	0 (préréglages)	Tableau pour des données AP ou utilisateur (données spécifiques NCK et spécifiques au canal)
		1	et d'autres réservées
	Nombre de lignes	Nombre de lignes y compris la ligne d'en-tête	
		Il n'est pas possible de faire défiler la ligne fixe ou la colonne fixe. Le nombre de colonnes est celui des colonnes configurées.	
	Affichage ligne de titre de colonne	-1:	La ligne de titre de colonne n'est pas affichée
		0 :	La ligne de titre de colonne est affichée (par défaut)
	Nombre de colonnes fixes	0 :	aucune colonne fixe
		1 ... n :	Nombre de colonnes devant toujours être visibles, c.-à-d. qui ne défilent pas à l'horizontale

Exemples

`//G(grid1/0/5/-1,2)`	Ligne de titre de colonne masquée et 2 colonnes fixes
`//G(grid1/0/5/,1)`	Ligne de titre de colonne affichée et 1 colonne fixe
`//G(grid1/0/5)`	Ligne de titre de colonne affichée et aucune colonne fixe

9.3.2 Définition des colonnes

Description

Pour les tableaux (grid), il peut être utile d'utiliser des variables avec indice. Le numéro d'indice est significatif pour les variables AP ou CN avec un ou plusieurs indices.

Les valeurs affichées dans un tableau peuvent être directement modifiées dans le cadre des droits définis par les attributs et des limites éventuellement définies.

Programmation

Syntaxe :	(Type / Valeurs limites / Vide / Texte long, titre de la colonne / Attributs / Image d'aide / Variable système ou utilisateur / Largeur de colonne / Offset1, offset2, offset3) Voir aussi Syntaxe de configuration étendue (Page 47).	
Description :	Définition des colonnes	
Paramètres :	similaires aux variables	
	Type	Type de données
	Valeurs limites	Valeur limite MIN, valeur limite MAX
	Texte long, Titre de colonne	
	Attributs	
	Image d'aide	
	Variable système ou utilisateur	Il faut entrer la variable AP ou CN comme variable, entre guillemets.
	Largeur de colonne	Indication en pixels
	Offset	Les incréments avec lesquels l'indice doit être augmenté afin de remplir les colonnes, sont indiqués dans le paramètre Offset affecté : • Offset1 : Incrément pour le 1er indice • Offset2 : Incrément pour le 2ème indice • Offset3 : Incrément pour le 3ème indice

Titre de la colonne à partir du fichier texte

Le titre de la colonne peut être indiqué sous forme de texte ou de numéro de texte ($8xxxx) ; il n'est pas possible de le faire défiler.

Modifier les caractéristiques de colonne

Les caractéristiques de colonne modifiables (inscriptibles) de façon dynamique se nomment :

- Valeurs limites (min,max)
- Titre de colonne (st)
- Attributs (wr, ac et li)
- Vue d'aide (hlp)

La modification d'une propriété de colonne s'effectue par le descripteur des variables à partir de la ligne de définition et par l'indice de colonne (commençant par 1).

Exemple : `VAR1[1].st="Colonne 1"`

Les attributs wr, ac et li peuvent être indiqués pour la définition de colonne.

Voir aussi

Load Grid (LG) (Page 165)

9.3.3 Contrôle du focus dans le tableau (grid)

Description

Les caractéristiques Row et Col permettent de définir le focus dans un tableau :

- descripteur.**Row**
- descripteur.**Col**

Programmation

Chaque cellule d'un tableau possède les caractéristiques Val et Vld.

Pour l'écriture et la lecture des propriétés de cellule, il faut indiquer en plus du descripteur des variables de la ligne de définition, un indice de ligne et de colonne.

Syntaxe :	Descripteur[indice de ligne, indice de colonne].Val ou Descripteur[indice de ligne, indice de colonne]
Description :	Caractéristiques Val
Syntaxe :	Descripteur[indice de ligne, indice de colonne].Vld
Description :	Caractéristiques Vld

Exemple

```
Var1[2,3].val=1.203
```

Si aucun indice de ligne ou de colonne n'est donné, les indices de la cellule ayant le focus sont valables, ce qui correspond à :

```
Var1.Row =2
Var1.Col=3
Var1.val=1.203
```

9.4 Custom Widgets

9.4.1 Définir le Custom Widgets

Description

A l'aide d'un Custom Widget, des éléments d'affichage spécifiques à l'utilisateur sont configurés dans la boîte de dialogue.

Option logicielle

Pour utiliser des Custom Widgets dans les boîtes de dialogue, vous devez disposer des options logicielles suivantes :

"SINUMERIK Integrate Run MyHMI /3GL" (6FC5800-0AP60-0YB0)

Programmation

Définition :	**DEF** (nom)	
Syntaxe :	**(W///"","**(nom de bibliothèque)**.**(nom de classe)**"/////a,**b**,**c**,**d**);**	
Description :	W	Définir le Custom Widget
Paramètres :	Nom	Nom de Custom Widget, librement choisi
	Nom de bibliothèque	Librement choisi, nom du fichier de bibliothèque dll (Windows) ou so (Linux)
	Nom de classe	Librement choisi, nom de la fonction de classe issu de la bibliothèque nommée auparavant
	a, b, c, d	Position et taille de la configuration

Exemple

Un Custom Widget est défini de la manière suivante dans la configuration de la boîte de dialogue :

```
DEF Cus = (W///"","slestestcustomwidget.SlEsTestCustomWidget"/////
20,20,250,100);
```

9.4.2 Structure de la bibliothèque Custom Widget

Description

La bibliothèque Custom Widget contient en substance une classe définie. Le nom de cette classe doit être indiqué dans la configuration de la boîte de dialogue à côté du nom de bibliothèque. A partir du nom de bibliothèque, "Run MyScreens" accède au fichier dll du même nom, par ex. :

```
slestestcustomwidget.dll
```

Programmation

La définition de classe du fichier dll doit ressembler à cela :

```
#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 Structure de l'interface Custom Widget

Description

Pour pouvoir afficher le Custom Widget dans la boîte de dialogue, la bibliothèque est complétée par une interface. Celle-ci contient des macrodéfinitions avec lesquelles "Run MyScreens" initie le Custom Widget. L'interface est présente sous forme d'un fichier cpp. Le nom de fichier est librement choisi, par ex. :
`sleswidgetfactory.cpp`

Programmation

L'interface est définie comme suit :

```
#include "slestestcustomwidget.h"     ; Le fichier d'en-tête du Custom Widget con-
                                      cerné est inséré au début du fichier
....
//Makros                              ; Les macrodéfinitions ne sont pas modifiées
....
```

```
WIDGET_CLASS_EXPORT(SlEsTestCustom-Widget)    ; Le Custom Widget concerné est déclaré à la fin du fichier
```

Exemple

Contenu du fichier sleswidgetfactory.cpp pour un Custom Widget avec le nom de classe "SlEsTestCustomWidget" :

```
#include <Qt/qglobal.h>
3#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 Interaction entre Custom Widget et boîte de dialogue – échange automatique de données

Les Custom Widgets interagissent avec des boîtes de dialogue et peuvent afficher ou manipuler des valeurs.

Conditions

Un échange automatique de données a lieu dans les conditions suivantes :

Condition	**Sens**
Au démarrage ou à la décompilation d'une boîte de dialogue	Boîte de dialogue → Custom Widget
A l'exécution de l'ordre CN pour la génération d'appels de cycles	Custom Widget → boîte de dialogue

Programmation

Les définitions suivantes sont nécessaires pour les interactions :

Extension de la configuration de la boîte de dialogue

Définition :	**DEF** (variable)	
Syntaxe :	**(**(type)**//5/"","**(variable)**","/wr2/)**	
Type de variables :	type	Champ de saisie standard (pas de Grid ou Toggle) avec un type de données quelconque (pas de W)
Paramètres :	Variable	Désignation quelconque d'une variable pour l'échange de données
Mode de saisie :	wr2	Lecture et écriture

Exemple

```
DEF CUSVAR1 = (R//5/"","CUSVAR1",""/wr2/)
```

Extension de la définition de la classe

Dans la définition de la classe du Custom Widget, un QProperty doit être créé, dont le nom est identique à celui de la variable sélectionnée de la configuration de la boîte de dialogue, par ex. : `Q_PROPERTY(double CUSVAR1 READ cusVar1 WRITE setCusVar1);`

Exemple

La définition de classe du fichier dll doit ressembler à cela :

```
#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 Interaction entre Custom Widget et boîte de dialogue – échange manuel de données

Outre l'échange automatique de données, un échange manuel est également possible. Celui-ci a lieu de manière dynamique, c'est-à-dire lors de l'exécution de la boîte de dialogue. Les actions suivantes sont possibles :

- Les propriétés du Custom Widget peuvent être lues et écrites.
- Les méthodes du Custom Widget peuvent être appelées à partir de la configuration Run MyScreens.
- Il est possible de réagir à un certain signal du Custom Widget et, par conséquent, d'appeler des sous-programmes (SUB) dans la configuration Run MyScreens.

9.4.5.1 Lecture et écriture de propriétés

Description

Les fonctions ReadCWProperties et WriteCWProperties sont disponibles dans la configuration Run MyScreens pour pouvoir lire et écrire des propriétés du Custom Widget.

Programmation

Syntaxe :	**ReadCWProperty**("Nom de variable", "Nom de propriété")	
Description :	Lecture d'une propriété d'un Custom Widget	
Paramètres :	Nom de variable	Nom de la variable de boîte de dialogue à laquelle est affecté un Custom Widget
	Nom de propriété	Nom de la propriété du Custom Widget à lire
Valeur retournée :	Valeur actuelle de la propriété du Custom Widget	

Syntaxe :	**WriteCWProperty**("Nom de variable", "Nom de propriété", "Valeur")	
Description :	Ecriture d'une propriété d'un Custom Widget	
Paramètres :	Nom de variable	Nom de la variable de boîte de dialogue à laquelle est affecté un Custom Widget
	Nom de propriété	Nom de la propriété du Custom Widget à écrire
	Valeur	Valeur qui doit être écrite dans la propriété du Custom Widget

Exemples

Exemple 1 :

Lecture de la propriété "MyStringVar" du Custom Widget qui est reliée à la variable de boîte de dialogue "MyCWVar1" et affectation de la valeur dans le registre 7.

Déclaration de la classe CustomWidget :

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(QString MyStringVar
              READ myStringVar
              WRITE setMyStringVar);
   ...
```

Configuration de la boîte de dialogue :

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   REG[7]=ReadCWProperty("MyCWVar1", "MyStringVar")
END_PRESS
```

Exemple 2 :

Ecriture du résultat du calcul "3 + sin(123.456)" dans la propriété "MyRealVar" du Custom Widget qui est reliée à la variable de boîte de dialogue "MyCWVar1".

Déclaration de la classe CustomWidget :

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(double MyRealVar
              READ myRealVar
              WRITE setMyRealVar);
   ...
```

Configuration de la boîte de dialogue :

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   WriteCWProperty("MyCWVar1", "MyRealVar", 3 + sin(123.456))
END_PRESS
```

9.4.5.2 Exécution d'une méthode du Custom Widget

Description

La fonction CallCWMethod est disponible dans la configuration Run MyScreens pour exécuter les méthodes du Custom Widget.

La méthode du Custom Widget à appeler doit comporter 10 paramètres de transfert au maximum.

Les formats de données des paramètres de transfert pris en charge sont les suivants :

- bool
- uint
- int
- double
- QString
- QByteArray

Programmation

Syntaxe :	**CallCWMethod**("Nom de variable", "Nom de méthode[, Argument 0][, Argument 1 ... [,Argument 9]")
Description :	Appel d'une méthode CustomWidget

Paramètres :	Nom de variable	Nom de la variable de boîte de dialogue à laquelle est affecté un Custom Widget
	Nom de méthode	Nom de la méthode Custom Widget à appeler
	Argument 0 - 9	Paramètre de transfert pour la méthode CustomWidget Formats de données pris en charge : voir ci-dessus **Remarque :** Les paramètres de transfert sont toujours transmis "ByVal", c'est-à-dire que c'est toujours uniquement la valeur qui est transférée et non la référence à une variable, par exemple.
Valeur retournée :	Valeur en retour de la méthode Custom Widget Les formats de données des paramètres de transfert pris en charge sont les suivants : • void • bool • uint • int • double • QString • QByteArray **Remarque :** Même si le format de données de la valeur en retour de la méthode Custom Widget est "void", celle-ci doit être affectée de manière formelle, par exemple à une variable.	

Exemple

Déclaration de la classe CustomWidget :

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
   Q_OBJECT
   public slots:
       void myFunc1(int nValue, const QString& szString, double dValue);
...
```

Configuration de la boîte de dialogue :

```
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
```

Remarque

Le Custom Widget doit implémenter la méthode "serialize". Cette méthode permet d'écrire les données internes d'un Custom Widget
dans un fichier spécifié ou de rétablir ces données à partir de ce fichier. Cela s'avère nécessaire surtout lors du passage à un autre groupe fonctionnel, puis du retour au premier, alors que le masque "Run MyScreens" est ouvert. Dans le cas contraire, les données internes sont perdues lors de la restauration de l'affichage.

Syntaxe :	public slots: bool **serialize**(const QString& szFilePath, bool bIsStoring);	
Description :	Lecture ou écriture des données internes et des états depuis ou dans un fichier	
Paramètres :	szFilePath	Nom, avec chemin d'accès complet, du fichier dans lequel les données internes et les états du Custom Widget doivent être écrites ou à partir duquel les données et états doivent être lus. Le fichier doit, le cas échéant, créer lui-même le Custom Widget.
	bIsStoring	TRUE = écrire FALSE = lire

Exemple

```
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)
```

```
bool SlEsTestCustomWidget::serialize(const QString& szFilePath, bool bIsStoring)
        {
           streamData << m_nDataCount << m_dValueX;
        }
        else
        {
           streamData >> m_nDataCount >> m_dValueX;
        }

        streamData.setDevice(0);
           fileData.flush();
           fileData.close();
        bReturn = true;
     }
  }
  return bReturn;
}
```

9.4.5.3 Réaction à un signal Custom Widget

Description

Dans Run MyScreens, il est possible de réagir à un signal donné (invokeSub()) du Custom Widget et ainsi d'appeler un sous-programme (SUB).

Pour le transfert de valeur (signal Custom Widget -> SUB), il existe 10 variables globales appelées SIGARG, dont la configuration est comparable aux registres (REG). Les valeurs transmises avec le signal Custom Widget y sont mémorisées.

Les formats de données des paramètres de transfert pris en charge sont les suivants :

- bool
- uint
- int
- double
- QString
- QByteArray

Programmation

Appel du sous-programme :

Syntaxe :	`void invokeSub(const QString& rszSignalName, const QVariantList& rvntList);`
Description :	Signal Custom Widget avec lequel est appelé un sous-programme Run My-Screens.

Paramètres :	rszSignalName	Nom du sous-programme Run MyScreens à appeler
	rvntList	Tableau QVariantList pour le transfert de paramètres mémorisés dans les paramètres globaux SIGARG et disponibles dans la configuration. Taille maximale : 10 éléments Formats de données pris en charge : voir ci-dessus **Remarque :** Les paramètres de transfert sont toujours transmis "ByVal", c'est-à-dire que c'est toujours uniquement la valeur qui est transférée et non la référence à une variable, par exemple.

Sous-programme à appeler :

Syntaxe :	`SUB(on_<Nom de variable>_<Nom de signal>)`
	`…`
	`END_SUB`	

Description :	Réaction à un signal Custom Widget	
Paramètres :	Nom de variable	Nom de la variable de boîte de dialogue à laquelle est affecté un Custom Widget
	Nom de signal	Nom du signal Custom Widget
	SIGARG 0 - 9	Paramètre de transfert pour la méthode Custom Widget Formats de données pris en charge : voir ci-dessus **Remarque :** Les paramètres de transfert sont toujours transmis "ByVal", c'est-à-dire que c'est toujours uniquement la valeur qui est transférée et non la référence à une variable, par exemple.

Exemple

Déclaration de la classe Custom Widget :

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
   Q_OBJECT
signals:
   void invokeSub(const QString& szSubName, const QVariantList& vntList);
…
}
```

Classe CustomWidget :

```
QVariantList vntList;
vntList << 123.456;
emit invokeSub("MySub", vntList);
```

Configuration de la boîte de dialogue :

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
SUB(on_MyCWVar1_MySub)
   DEBUG("SUB(on_MyCWVar1_MySub) was called with parameter: """ << SIGARG[0] <<
""""")
END_SUB
```

Résultat "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

Généralités

SlEsGraphCustomWidget vous permet de représenter des objets géométriques (point, ligne, rectangle, rectangle avec coins arrondis, ellipse, arc de cercle, texte) et des courbes à partir de points d'interpolation (p. ex. valeurs de mesure, courbe).

Les objets sont organisés en un ou plusieurs éléments appelés "contours". Ces derniers peuvent alors être affichés ou vidés, sélectionnés et supprimés, individuellement ou de manière groupée.

Les objets sont ajoutés à l'aide des fonctions du contour actuellement sélectionné et sont aussi tracés dans cet ordre. Si vous souhaitez tracer les objets dans une couleur particulière, vous devez définir la couleur correspondante du tracé avant de l'insérer. Tous les objets insérés par la suite sont ensuite tracés dans cette couleur. Outre la couleur du tracé, vous pouvez aussi influencer la largeur et le style du tracé. Pour les objets fermés que sont le rectangle, le rectangle avec coins arrondis et l'ellipse, vous pouvez définir une couleur de remplissage avant d'insérer les objets.

Chaque contour peut être formé selon différents modes :

- Affichage normal de tous les types d'objets.
- Les points peuvent être reliés pour former ce qu'on appelle une polyligne, et ainsi être représentés sous forme de courbe/graphique.
- La zone entre les points, les lignes ou les arcs de cercle jusqu'à l'axe X peut être colorée avec la couleur de remplissage actuelle. Vous pouvez utiliser cette possibilité par exemple pour visualiser la matière restante lors du tournage.
  Dans ce mode, si les points sont également affichés sous forme de polyligne, l'ensemble de la zone sous la courbe jusqu'à l'axe X est remplie. Les points, les lignes et les arcs de cercle peuvent se trouver dans chacun des quatre quadrants.

Avec un mode de curseur spécial, vous pouvez "parcourir" un ou plusieurs contours. Pour cela, le curseur est placé sur le premier ou le dernier objet point d'un contour, ou un objet point est déplacé vers l'avant ou l'arrière sur le contour en partant de la position actuelle du curseur.

Si nécessaire, un deuxième axe Y (droite) avec une échelle propre peut être affiché. Celui-ci est couplé au premier axe Y (gauche) par un offset et un facteur.

Grâce à la fonction de recherche, vous pouvez trouver au moyen d'une coordonnée X donnée un point ajouté précédemment à un contour et, si vous le souhaitez, le curseur peut être placé sur ce point.

Vous pouvez également configurer les contours en tant que tampon FIFO avec taille réglable.

Avec la sérialisation, l'état actuel du SlEsGraphCustomWidget peut être enregistré sous forme binaire dans un fichier et restauré à partir de ce fichier.

Le SlEsGraphCustomWidget peut être commandé par le geste "Déplacer" ("Pan", déplacement de la vue) et les gestes "Pincer"/"Etirer" ("Pinch"/"Spread", zoom avant/arrière dans la vue).

La souris permet de déplacer la vue (bouton gauche de la souris + Move) et d'effectuer un zoom (molette de la souris).

L'affichage n'est pas actualisé automatiquement pour des raisons de performance. Selon le cas d'application, vous pouvez lancer l'actualisation en appelant la fonction correspondante.

Exemple

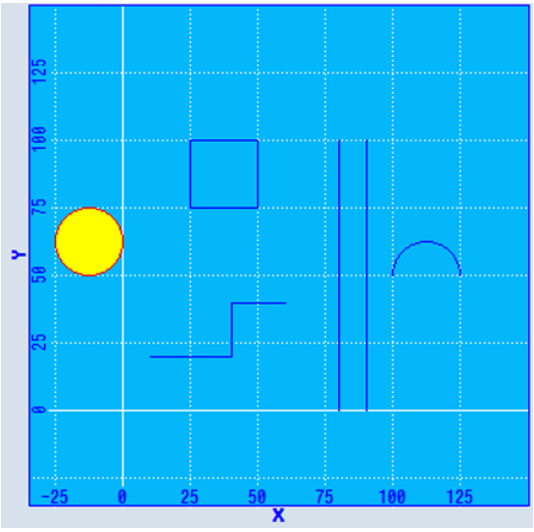


Figure 9-1 SlEsGraphCustomWidget - Exemple

```
//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)
```

```
//M(MyGraphSampleMask/"SlEsGraphCustomWidget Sample")

   REG[0]= CALLCWMETHOD("MyGraphVar", "addContour", "MyContour", TRUE)
   REG[0]= CALLCWMETHOD("MyGraphVar", "showContour", "MyContour")
   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 Remarques concernant les performances

En fonction du matériel utilisé et du taux d'utilisation de base du système, vous devez prendre en compte les valeurs de référence suivantes lors de l'utilisation du SlEsGraphCustomWidget. Outre le matériel utilisé et le taux d'utilisation de base, les valeurs varient en fonction de la configuration des SIEsGraphCustomWidgets.

Tenez compte de ces valeurs de référence afin de ne pas entraver la réactivité et la stabilité de l'ensemble du système.

- Nombre de SlEsGraphCustomWidgets configurés en même temps à un instant donné (p. ex. 1 au maximum)
- Nombre de contours configurés (p. ex. 6 au maximum)
- Nombre d'objets graphiques configurés par contour (p. ex. 1000 au maximum)
- Fréquence des actualisations demandées (p. ex. 500 ms au maximum)

Remarque

L'affichage n'est pas conçu pour fonctionner en temps réel.

9.5.3 Lecture et écriture de propriétés

Description

Les propriétés présentées dans le chapitre suivant sont lues avec la fonction ReadCWProperty() et définies avec WriteCWProperty().

Exemples

Lecture des propriétés "CursorX" du SlEsGraphCustomWidget lié par la variable d'affichage "MyGraphVar". Le résultat est écrit dans le registre 0.

```
REG[0] = ReadCWProperty("MyGraphVar", "CursorX")
```

Ecriture de la valeur "MyFirstContour" dans la propriété "SelectedContour" du SlEsGraphCustomWidget lié par la variable d'affichage "MyGraphVar".

```
WriteCWProperty("MyGraphVar ", "SelectedContour", "MyFirstContour")
```

9.5.4 Properties

Vue d'ensemble

Property	Description
AxisNameX	Désignation de l'axe X
AxisNameY	Désignation de l'axe Y
AxisNameY2	Désignation du deuxième axe Y (droite)
AxisY2Visible	Afficher/masquer le deuxième axe Y (droite)
AxisY2Offset	Offset du deuxième axe Y (droite)
AxisY2Factor	Facteur du deuxième axe Y (droite)
ScaleTextEmbedded	Positionnement des textes d'échelle
ScaleTextOrientationYAxis	Orientation des textes d'échelle de l'axe Y
KeepAspectRatio	Conserver les proportions
SelectedContour	Nom du contour actuellement sélectionné
BackColor	Couleur d'arrière-plan
ChartBackColor	Couleur d'arrière-plan de la zone de tracé
ForeColor	Couleur d'avant-plan pour les textes et couleur par défaut du tracé
ForeColorY2	Couleur d'avant-plan pour les textes du deuxième axe Y (droite)
GridColor	Couleur des lignes de la grille
GridColorY2	Couleur des lignes horizontales de la grille du deuxième axe Y (droite)
CursorColor	Couleur du réticule du curseur
ShowCursor	Afficher/masquer le curseur
CursorX	Position X du curseur
CursorY	Position Y du curseur

Property	Description
CursorY2	Position Y du curseur par rapport au deuxième axe Y (droite)
CursorStyle	Type de représentation du curseur
ViewMoveZoomMode	Comportement lors du zoom et du déplacement avec des gestes

AxisNameX – Désignation de l'axe X

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**AxisNameX**") WriteCWProperty(GraphVarName, "**AxisNameX**", Value)	
Description :	Lit ou définit le descripteur de l'axe X. Si aucun descripteur n'est indiqué, l'espace disponible est utilisé pour la zone de tracé.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (QString)
	Value	Valeur à définir (QString)

AxisNameY – Désignation de l'axe Y

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**AxisNameY**") WriteCWProperty(GraphVarName, "**AxisNameY**", Value)	
Description :	Lit ou définit le descripteur de l'axe Y. Si aucun descripteur n'est indiqué, l'espace disponible est utilisé pour la zone de tracé.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (QString)
	Value	Valeur à définir (QString)

AxisY2Visible – Afficher/masquer le deuxième axe Y (droite)

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Visible**") WriteCWProperty(GraphVarName, "**AxisY2Visible**", Value)	
Description :	Afficher/masquer le deuxième axe Y (droite)	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE ou FALSE

AxisY2Offset – Offset du deuxième axe Y (droite)

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Offset**") WriteCWProperty(GraphVarName, "**AxisY2Offset**", Value)
Description :	Offset du deuxième axe Y (droite) par rapport au premier axe Y (gauche). Voir l'exemple pour la propriété AxisY2Factor.

Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (double)
	Value	Valeur à définir (double)

AxisY2Factor – Facteur du deuxième axe Y (droite)

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Factor**") WriteCWProperty(GraphVarName, "**AxisY2Factor**", Value)	
Description :	Facteur du deuxième axe Y (droite) par rapport au premier axe Y (gauche).	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (double)
	Value	Valeur à définir (double)

En association avec la propriété "AxisY2Offset", il est possible d'afficher un deuxième axe Y (droite) avec une échelle propre. L'échelle est couplée au premier axe Y (gauche) par l'offset et le facteur.

Formule pour la conversion de Y2 à Y :

Y = Y2 / facteur - offset

Formule pour la conversion de Y à Y2 :

Y2 = (Y + offset) * facteur

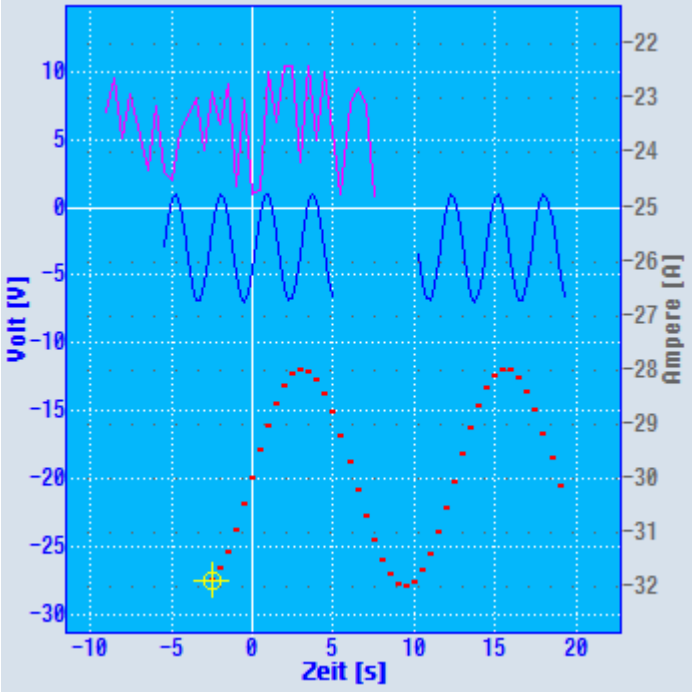


Figure 9-2 Exemple avec le deuxième axe Y avec échelle propre

Exemple

Y : 0 à 200 doit correspondre à Y2 : -25 à 25.

- Offset : -100
- Facteur : 0,25

Exemple de calcul :

Y2 = -30

Y = -30 / 0,25 – (-100) = -20

Exemple de calcul :

Y = 0

Y2 = (0 + (-100)) * 0,25 = -25

L'axe X est identique pour les deux systèmes de coordonnées.

Vous pouvez définir les couleurs avec setForeColorY2() et setGridColorY2() (voir Couleurs).

Le libellé des axes est défini avec setAxisNameY2() (voir Désignation des axes).

ScaleTextEmbedded – Positionnement des textes d'échelle

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**ScaleTextEmbedded**") WriteCWProperty(GraphVarName, "**ScaleTextEmbedded**", Value)	
Description :	Cette propriété vous permet de déterminer si les textes d'échelle doivent être positionnés à l'intérieur ou à l'extérieur de la zone de tracé.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE ou FALSE

Exemple

Textes d'échelle à l'extérieur de la zone de tracé :

ScaleTextEmbedded = FALSE

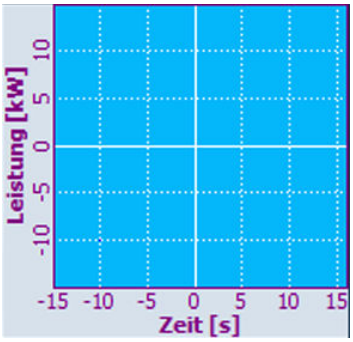


Figure 9-3 Textes d'échelle à l'extérieur de la zone de tracé

Textes d'échelle à l'intérieur de la zone de tracé :

ScaleTextEmbedded = TRUE

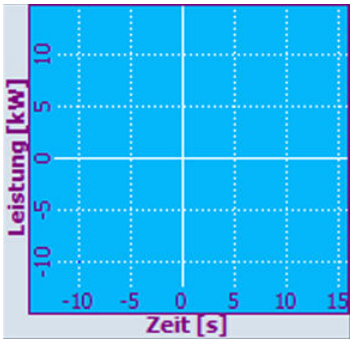


Figure 9-4 Textes d'échelle à l'intérieur de la zone de tracé

ScaleTextOrientationYAxis – Orientation des textes d'échelle de l'axe Y

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**ScaleTextOrientationYAxis**") WriteCWProperty(GraphVarName, "**ScaleTextOrientationYAxis**", Value)	
Description :	Cette fonction permet d'orienter verticalement les textes d'échelle de l'axe Y.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (int)
	Value	Valeur à définir (int) : 1 (= horizontalement) ou 2 (= verticalement)

Exemple

Textes d'échelle à l'intérieur de la zone de tracé :

- ScaleTextEmbedded = TRUE

Orientation horizontale du texte :

- ScaleTextOrientationYAxis = 1

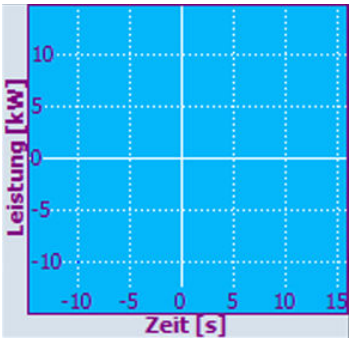


Figure 9-5 Textes d'échelle à l'intérieur de la zone de tracé, orientation horizontale du texte

Orientation verticale du texte :

- ScaleTextOrientationYAxis = 2

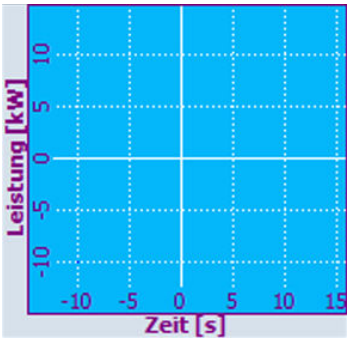


Figure 9-6 Textes d'échelle à l'intérieur de la zone de tracé, orientation verticale du texte

Textes d'échelle à l'extérieur de la zone de tracé :

- ScaleTextEmbedded = FALSE

Orientation horizontale du texte :

- ScaleTextOrientationYAxis = 1

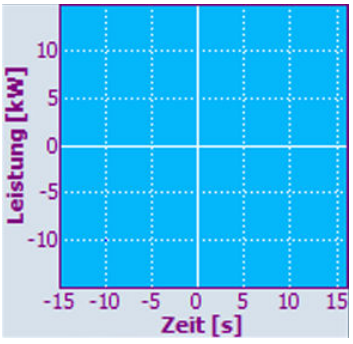


Figure 9-7 Textes d'échelle à l'extérieur de la zone de tracé, orientation horizontale du texte

Orientation verticale du texte :

- ScaleTextOrientationYAxis = 2

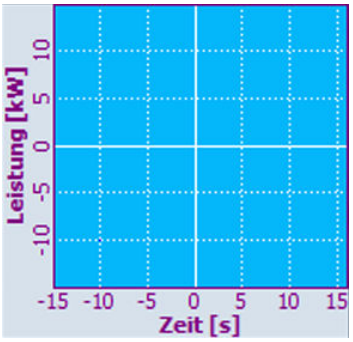


Figure 9-8 Textes d'échelle à l'extérieur de la zone de tracé, orientation verticale du texte

KeepAspectRatio – Conserver les proportions

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**KeepAspectRatio**") WriteCWProperty(GraphVarName, "**KeepAspectRatio**", Value)	
Description :	Cette propriété vous permet de déterminer si la vue définie avec setView() doit être automatiquement orientée, adaptée ou agrandie de manière à ce que les proportions de l'axe X par rapport à l'axe Y restent toujours les mêmes. Cette propriété est pertinente en particulier lors de l'affichage de figures géométriques. Ainsi, par exemple, un cercle ou un carré est représenté en tant que tel et non de façon déformée sous forme d'ellipse ou de rectangle.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (bool)
	Return Value	Valeur à définir (bool) : TRUE ou FALSE

Exemple

```
setView(-15, -15, 15, 15)
setFillColor("#A0A0A4")
addCircle(0, 0, 5)
KeepAspectRatio = TRUE
```

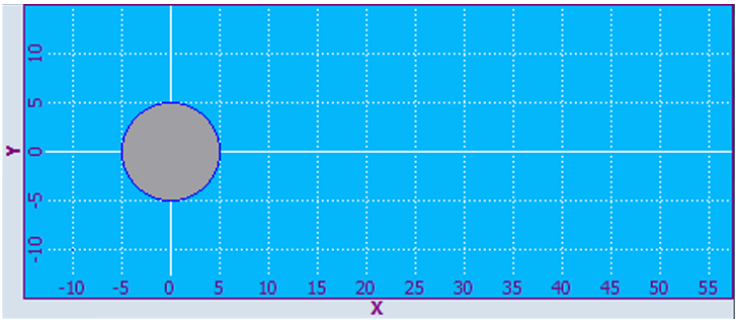


Figure 9-9 Les proportions de l'axe X par rapport à l'axe Y restent les mêmes

```
KeepAspectRatio = FALSE
```

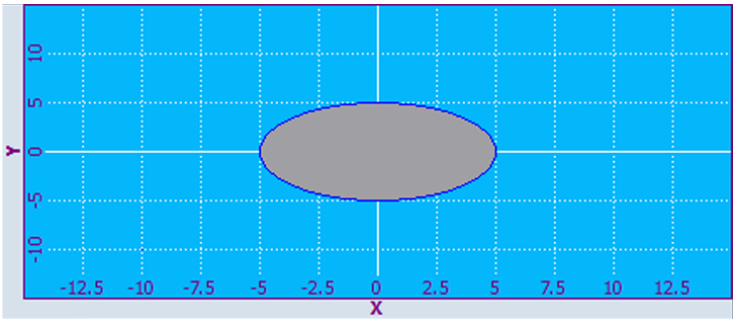


Figure 9-10 Les proportions de l'axe X par rapport à l'axe Y sont déformées

SelectedContour – Nom du contour actuellement sélectionné

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, " **SelectedContour**") WriteCWProperty(GraphVarName, " **SelectedContour**", Value)	
Description :	Lit ou définit la sélection du contour actuel.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (QString)
	Value	Valeur à définir (QString)

Remarque

Avant de pouvoir effectuer des opérations sur un contour, p. ex. pour insérer des objets graphiques, vous devez d'abord les sélectionner.

BackColor – Couleur d'arrière-plan

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, " **BackColor**") WriteCWProperty(GraphVarName, " **BackColor**", Value)	
Description :	Couleur d'arrière-plan pour le libellé des axes, en fonction de la propriété Scale-TextInsideChartArea des textes d'échelle également.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Couleur lue de la propriété (QString)
	Value	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

ChartBackColor – Couleur d'arrière-plan de la zone de tracé

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**ChartBackColor**") WriteCWProperty(GraphVarName, "**ChartBackColor**", Value)
Description :	Couleur d'arrière-plan pour la zone de tracé.

Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Couleur lue de la propriété (QString)
	Value	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

ForeColor – Couleur d'avant-plan pour le libellé et couleur par défaut du tracé

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**ForeColor**") WriteCWProperty(GraphVarName, "**ForeColor**", Value)	
Description :	Couleur d'avant-plan pour les textes et couleur par défaut du tracé.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Couleur lue de la propriété (QString)
	Value	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

ForeColorY2 – Couleur d'avant-plan pour le libellé du deuxième axe Y (droite)

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**ForeColorY2**") WriteCWProperty(GraphVarName, "**ForeColorY2**", Value)	
Description :	Couleur d'avant-plan pour les textes du deuxième axe Y (droite).	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Couleur lue de la propriété (QString)
	Value	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

GridColor – Couleur des lignes de la grille

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**GridColor**") WriteCWProperty(GraphVarName, "**GridColor**", Value)	
Description :	Couleur des lignes de la grille.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Couleur lue de la propriété (QString)
	Value	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

GridColorY2 – Couleur des lignes horizontales de la grille du deuxième axe Y (droite)

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**GridColorY2**") WriteCWProperty(GraphVarName, "**GridColorY2**", Value)
Description :	Couleur des lignes horizontales de la grille du deuxième axe Y (droite).

Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Couleur lue de la propriété (QString)
	Value	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

CursorColor – Couleur du curseur

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**CursorColor**") WriteCWProperty(GraphVarName, "**CursorColor**", Value)	
Description :	Couleur du curseur.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Couleur lue de la propriété (QString)
	Value	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

ShowCursor – Afficher/masquer le curseur

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, " **ShowCursor**") WriteCWProperty(GraphVarName, " **ShowCursor**", Value)	
Description :	Afficher/masquer le curseur.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE ou FALSE

Remarque

Cette fonction actualise automatiquement l'affichage.

CursorX – Position X du curseur

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**CursorX**") WriteCWProperty(GraphVarName, "**CursorX**", Value)	
Description :	Position X du curseur.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (double)
	Value	Valeur à définir (double)

Remarque

Cette fonction actualise automatiquement l'affichage.

CursorY – Position Y du curseur

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**CursorY**") WriteCWProperty(GraphVarName, "**CursorY**", Value)	
Description :	Position Y du curseur.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (double)
	Value	Valeur à définir (double)

Remarque

Cette fonction actualise automatiquement l'affichage.

CursorY2 – Position Y du curseur par rapport au deuxième axe Y (droite)

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**CursorY2**") WriteCWProperty(GraphVarName, "**CursorY2**", Value)	
Description :	Position Y du curseur par rapport au deuxième axe Y (droite).	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (double)
	Value	Valeur à définir (double)

Remarque

Cette fonction actualise automatiquement l'affichage.

CursorStyle – Type de représentation du curseur

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**CursorStyle**") WriteCWProperty(GraphVarName, "**CursorStyle**", Value)	
Description :	Type de représentation du curseur.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Valeur lue de la propriété (int)
	Value	Valeur à définir (int) : 0 (= croix) 1 (= réticule) 2 (= ligne verticale) 3 (=ligne horizontale) 4 (= lignes horizontale et verticale)

ViewMoveZoomMode – Comportement lors du zoom et du déplacement avec des gestes

Syntaxe :	ReturnValue = ReadCWProperty(GraphVarName, "**ViewMoveZoomMode**") WriteCWProperty(GraphVarName, "**ViewMoveZoomMode**", Value)	
Description :	La vue affichée du SlEsGraphCustomWidget peut être modifiée par des gestes. Cette propriété vous permet de déterminer de quelle manière cela est autorisé. Par exemple, il peut s'avérer utile dans certains cas d'application d'autoriser uniquement le déplacement horizontal et le zoom, p. ex. pour l'affichage de la courbe des valeurs de mesure.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Valeur lue de la propriété (int)
	Value	Valeur à définir (int) : 0 (= désactivé) 1 (= horizontalement uniquement) 2 (= verticalement uniquement) 3 (= horizontalement et verticalement)

9.5.5 Fonctions

Vous pouvez appeler les fonctions indiquées ci-dessous avec la fonction CallCWMethod().

Exemple

Définition de la vue du SlEsGraphCustomWidget lié par la variable d'affichage "MyGraphVar".

Le résultat est écrit dans le registre 0.

```
REG[0] = CallCWMethod("MyGraphVar", "setView", -8, -5, 11- 20)
```

Vue d'ensemble

Fonction	Description
setView	Définir un système de coordonnées
setMaxContourObjects	Définir le nombre max. d'objets d'un contour (tampon FIFO)
addContour	Ajouter un contour
showContour	Afficher le contour
hideContour	Rendre le contour invisible
hideAllContours	Rendre tous les contours invisibles
removeContour	Supprimer le contour
clearContour	Effacer les objets graphiques d'un contour
fitViewToContours	Adaptation automatique de la vue
fitViewToContour	Adaptation automatique de la vue
findX	Recherche dans un contour
setPolylineMode	Représenter les points d'un contour en tant que polyligne/courbe
setIntegralFillMode	Représenter les points d'un contour en tant que polyligne/courbe

Fonction	Description
repaint, update	Mettre à jour la vue
addPoint	Ajouter un point d'un contour
addLine	Ajouter une ligne d'un contour
addRect	Ajouter un rectangle d'un contour
addRoundedRect	Ajouter un rectangle avec coins arrondis d'un contour
addEllipse	Ajouter une ellipse d'un contour
addCircle	Ajouter un cercle d'un contour
addArc	Ajouter un arc de cercle d'un contour
addText	Ajouter un texte d'un contour
setPenWidth	Définir la largeur du tracé
setPenStyle	Définir le style du tracé
setPenColor	Définir la couleur du tracé
setFillColor	Définir la couleur de remplissage
setCursorPosition	Positionnement du curseur
setCursorPositionY2	Positionner le curseur par rapport au deuxième axe Y (droite)
setCursorOnContour	Positionner le curseur sur le contour
moveCursorOnContourBegin	Positionner le curseur sur le premier objet graphique d'un contour
moveCursorOnContourEnd	Positionner le curseur sur le dernier objet graphique d'un contour
moveCursorOnContourNext	Positionner le curseur sur l'objet graphique suivant d'un contour
serialize	Enregistrement/restauration de l'état actuel

setView – Définir un système de coordonnées

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setView**", x1, y1, x2, y2)	
Description :	Cette fonction vous permet de définir la taille du système de coordonnées qui doit être affiché dans le SlEsGraphCustomWidget. Voir aussi à ce sujet la propriété KeepAspectRatio.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x1	Bord gauche (double)
	y1	Bord supérieur (double)
	x2	Bord droit (double)
	y2	Bord inférieur (double)

Remarque

La fonction actualise automatiquement l'affichage.

Exemple

```
setView(-8, -5, 11, 20)
```

```
AxisNameX = "X"
AxisNameY = "Y"
ScaleTextOrientationYAxis = 1
ScaleTextEmbedded = false
KeepAspectRatio = false
update
```

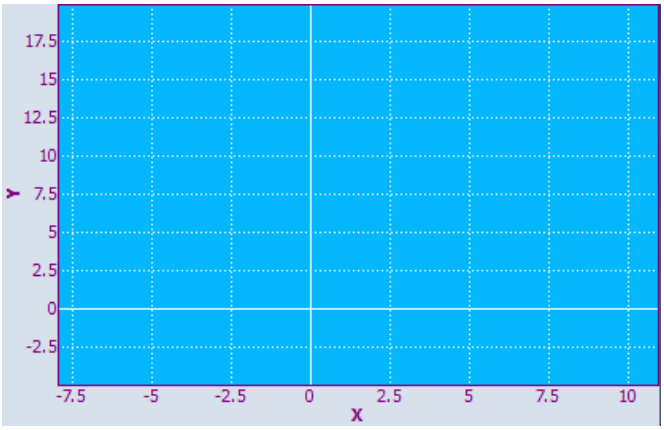


Figure 9-11 Exemple - setView

setMaxContourObjects – Définir le nombre max. d'objets d'un contour (tampon FIFO)

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setMaxContourObjects**", Value) ReturnValue = CallCWMethod(GraphVarName, "**setMaxContourObjects**", Value, ContourName)	
Description :	Cette fonction vous permet de déterminer la taille du tampon FIFO d'un contour. Si le nombre d'objets ajoutés au contour dépasse cette limite, l'objet le plus ancien est effacé.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	Value	Nombre maximum d'objets graphiques (uint)
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

Remarque

La fonction actualise automatiquement l'affichage.

addContour – Ajouter un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName) ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName, Selected) ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName, Selected, AssignedY2)	
Description :	Cette fonction vous permet de créer un nouveau contour. En option, vous pouvez affecter le contour au système de coordonnées du deuxième axe Y (droite).	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.
	Selected	Le contour doit être sélectionné (bool) : TRUE ou FALSE
	AssignedY2	Le contour doit être affecté au deuxième axe Y (droite) (bool): TRUE ou FALSE

showContour – Afficher le contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**showContour**") ReturnValue = CallCWMethod(GraphVarName, "**showContour**", ContourName)	
Description :	Cette fonction rend un contour visible.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

hideContour – Rendre le contour invisible

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**hideContour**") ReturnValue = CallCWMethod(GraphVarName, "**hideContour**", ContourName)	
Description :	Cette fonction rend un contour invisible.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

showAllContours – Rendre tous les contours visibles

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**showAllContours**")	
Description :	Cette fonction rend tous les contours visibles.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi

hideAllContours – Rendre tous les contours invisibles

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**hideAllContours**")	
Description :	Cette fonction rend tous les contours invisibles.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi

removeContour – Rendre le contour invisible

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**removeContour**") ReturnValue = CallCWMethod(GraphVarName, "**removeContour**", ContourName)	
Description :	Supprime un contour.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

clearContour – Effacer les objets graphiques d'un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**clearContour**") ReturnValue = CallCWMethod(GraphVarName, "**clearContour**", ContourName)	
Description :	Efface tous les objets graphiques contenus dans un contour.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

fitViewToContours – Adaptation automatique de la vue

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContours**") ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContours**", OnlyVisible)	
Description :	Cette fonction permet d'adapter automatiquement la vue aux contours. En fonction du paramètre de transfert vous définissez si tous les contours sont rendus visibles ou seulement tous les contours visibles.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	OnlyVisible	Valeur à définir (bool) : TRUE ou FALSE

fitViewToContour – Adaptation automatique de la vue

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContour**", ContourName)	
Description :	Cette fonction permet d'adapter automatiquement la vue de sorte à ne rendre visible qu'un seul contour donné.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString)

findX – Recherche dans un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**findX**", x) ReturnValue = CallCWMethod(GraphVarName, "**findX**", x, ContourName) ReturnValue = CallCWMethod(GraphVarName, "**findX**", x, ContourName, SetCursor)	
Description :	Cette fonction vous permet de rechercher un point défini précédemment dans un contour avec une coordonnée X donnée. Le résultat obtenu est la coordonnée Y. En option, vous pouvez aussi positionner immédiatement le curseur sur ce point.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Errorcode (QString): Lorsque la recherche a abouti, la valeur Y associée est retournée
	x	Valeur X à rechercher (double)
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.
	SetCursor	Définir le curseur (bool) : TRUE ou FALSE

Remarque

Lorsque vous définissez le curseur avec cette fonction, l'affichage est actualisé automatiquement.

setPolylineMode – Représenter les points d'un contour en tant que polyligne/courbe

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setPolylineMode**", SetPoly)	
Description :	Tous les points du contour actuel ajoutés après cet appel de fonction sont reliés pour former une polyligne/courbe. Cette fonction est spécifique au contour et peut être activée et désactivée par tronçons.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	SetPoly	PolylineMode (bool) : TRUE = activé

Exemple

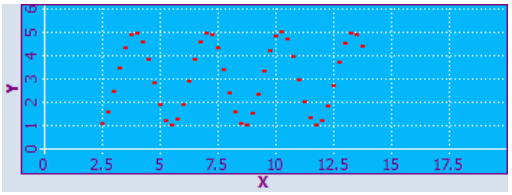


Figure 9-12 Exemple - PolylineMode : désactivé

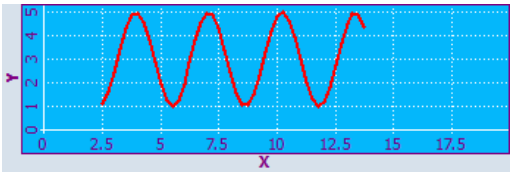


Figure 9-13 Exemple - PolylineMode : activé

setIntegralFillMode – Remplissage des zones entre les points, les lignes et les arcs de cercle et l'axe X

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setIntegralFillMode**", SetIntFill)	
Description :	Les zones entre les points, les lignes et les arcs de cercle et l'axe X sont remplies avec la couleur de remplissage actuelle (que les points soient +Y ou –Y). Cette fonction est spécifique au contour et peut être activée et désactivée par tronçons.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	SetIntFill	IntegralFillMode (bool): TRUE = activé

Exemple

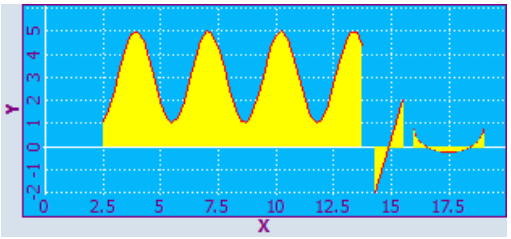


Figure 9-14 Exemple - setIntegralFillMode : activé

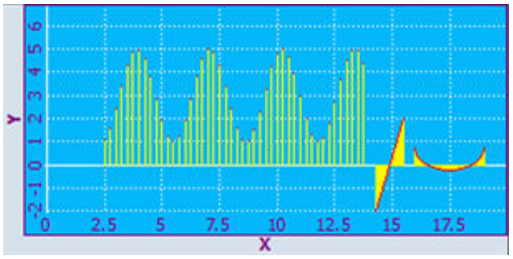


Figure 9-15 Exemple - setIntegralFillMode : désactivé

repaint, update – Actualiser la vue

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**repaint**") ReturnValue = CallCWMethod(GraphVarName, "**update**")	
Description :	Cette fonction vous permet d'actualiser manuellement l'affichage.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SIEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi

- update()
  La fonction actualise la vue, à condition que l'actualisation du Widget ne soit pas désactivée et que le Widget ne soit pas caché. La fonction est optimisée de manière à ce que les performances de l'application soient préservées et que la vue ne scintille pas en raison d'actualisations trop fréquentes.
- repaint()
  La fonction actualise la vue immédiatement après l'appel de fonction. Ainsi, il est préférable d'utiliser la fonction repaint uniquement lorsqu'une actualisation immédiate est absolument nécessaire, p. ex. pour les animations.

Il est recommandé de toujours utiliser la fonction update(). En interne, le SIEsGraphCustomWidget utilise toujours la fonction update(), p. ex. avec setView().

addPoint – Ajouter un point à un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addPoint**", x, y) ReturnValue = CallCWMethod(GraphVarName, "**addPoint**", x, y, DummyPoint)	
Description :	Ajouter un point au contour actuellement sélectionné. En outre, vous pouvez définir un point factice qui n'est pas tracé mais sur lequel le curseur peut être positionné.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x	Coordonnée x (double)
	y	Coordonnée y (double)
	DummyPoint	Le point est traité comme invisible (bool) : TRUE = oui

addLine – Ajouter une ligne à un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addLine**", x1, y1, x2, y2)	
Description :	Ajouter une ligne au contour actuellement sélectionné.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x1	Coordonnée x du point de départ (double)
	y1	Coordonnée y du point de départ (double)
	x2	Coordonnée x du point final (double)
	y2	Coordonnée y du point final (double)

addRect – Ajouter un rectangle à un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addRect**", x1, y1, x2, y2)	
Description :	Ajouter un rectangle au contour actuellement sélectionné.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x1	Bord gauche (double)
	y1	Bord gauche (double)
	x2	Bord droit (double)
	y2	Bord inférieur (double)

addRoundedRect – Ajouter un rectangle avec coins arrondis à un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addRoundedRect**", x1, y1, x2, y2, r)
Description :	Ajouter un rectangle avec coins arrondis au contour actuellement sélectionné.

Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x1	Bord gauche (double)
	y1	Bord supérieur (double)
	x2	Bord supérieur (double)
	y2	Bord supérieur (double)
	r	Rayon (double)

addEllipse – Ajouter une ellipse à un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addEllipse**", x1, y1, x2, y2, radius)	
Description :	Ajouter une ellipse au contour actuellement sélectionné.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x1	Bord gauche (double)
	y1	Bord supérieur (double)
	x2	Bord droit (double)
	y2	Bord inférieur (double)
	radius	Rayon (double)

addCircle – Ajouter un cercle à un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addCircle**", x, y, radius)	
Description :	Ajouter un cercle au contour actuellement sélectionné.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x	Bord gauche (double)
	y	Bord supérieur (double)
	radius	Rayon (double)

addArc – Ajouter un arc de cercle à un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addArc**", x1, y1, x2, y2, StartAngle, SpanAngle
Description :	Ajouter un arc de cercle au contour actuellement sélectionné.

Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x1	Bord gauche (double)
	y1	Bord supérieur (double)
	x2	Bord droit (double)
	y2	Bord inférieur (double)
	StartAngle	Angle de départ en degrés (double)
	SpanAngle	Angle d'ouverture en degrés (double)

addText – Ajouter un texte à un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**addText**", x, y, Text)	
Description :	Ajouter un texte au contour actuellement sélectionné.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x	Bord gauche (double)
	y	Bord supérieur (double)
	Texte	Texte (QString)

setPenWidth – Définir la largeur du tracé

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setPenWidth**") ReturnValue = CallCWMethod(GraphVarName, "**setPenWidth**", width)	
Description :	Détermine la largeur du tracé à partir de l'appel de fonction pour les objets du contour actuel ajoutés par la suite.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	width	Largeur de tracé (double). Si aucune largeur de tracé n'est indiquée, la largeur de tracé par défaut est définie.

setPenStyle – Définir le style du tracé

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setPenStyle**") ReturnValue = CallCWMethod(GraphVarName, "**setPenStyle**", style)
Description :	Détermine le style du tracé à partir de l'appel de fonction pour les objets du contour actuel ajoutés par la suite.

Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	style	1 : ligne continue (___) 2 : ligne discontinue (_ _) 3 : ligne pointillée (...) 4 : ligne-point-ligne (_._) 5 : ligne-point-point (_..)

setPenColor – Définir la couleur du tracé

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setPenColor**") ReturnValue = CallCWMethod(GraphVarName, "**setPenColor**", Color)	
Description :	Détermine la couleur du tracé à partir de l'appel de fonction pour les objets du contour actuel ajoutés par la suite.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	Color	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

setFillColor – Définir la couleur de remplissage

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setFillColor**") ReturnValue = CallCWMethod(GraphVarName, "**setFillColor**", Color)	
Description :	Détermine la couleur de remplissage à partir de l'appel de fonction pour les objets du contour actuel ajoutés par la suite.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	Color	Valeur à définir (QString) en tant que valeur RVB au format "#RRVVBB", p. ex. "#04B7FB"

setCursorPosition – Positionner le curseur

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setCursorPosition**", x, y)	
Description :	Place le curseur à la position indiquée.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x	Coordonnée x (double)
	y	Coordonnée y (double)

Remarque

Cette fonction actualise automatiquement l'affichage.

setCursorPositionY2Cursor – Positionner le curseur par rapport au deuxième axe Y (droite)

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setCursorPositionY2**", x, y2)	
Description :	Place le curseur à la position indiquée par rapport au deuxième axe Y (droite).	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	x	Coordonnée x (double)
	y2	Coordonnée Y par rapport au deuxième axe Y (droite) (double)

Remarque

Cette fonction actualise automatiquement l'affichage.

setCursorOnContour – Positionner le curseur sur le contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**setCursorOnContour**") ReturnValue = CallCWMethod(GraphVarName, "**setCursorOnContour**", Contour-Name)	
Description :	Place le curseur à la dernière position de curseur au sein d'un contour.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

Remarque

Cette fonction actualise automatiquement l'affichage.

Exemple

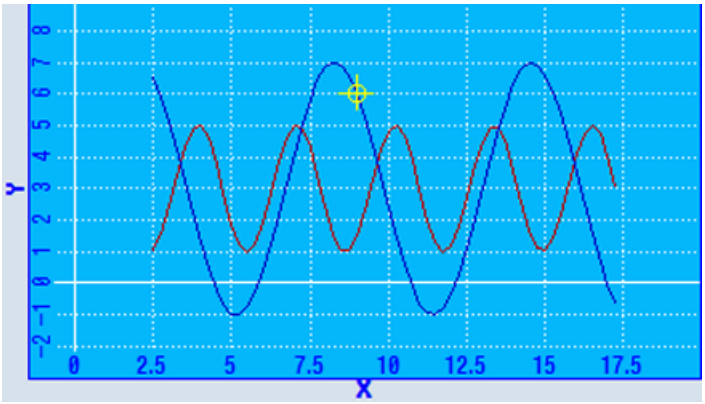


Figure 9-16 Exemple - setCursorOnContour

moveCursorOnContourBegin – Positionner le curseur sur le premier objet graphique d'un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBegin**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBegin**", ContourName)	
Description :	Cette fonction vous permet de déplacer le curseur sur le premier objet point d'un contour en partant de la position actuelle du curseur sur le contour.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

Remarque

Cette fonction actualise automatiquement l'affichage.

moveCursorOnContourEnd – Positionner le curseur sur le dernier objet graphique d'un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourEnd**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourEnd**", ContourName)	
Description :	Cette fonction vous permet de déplacer le curseur sur le dernier objet point d'un contour en partant de la position actuelle du curseur sur le contour.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

Remarque

Cette fonction actualise automatiquement l'affichage.

moveCursorOnContourNext – Positionner le curseur sur l'objet graphique suivant d'un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourNext**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourNext**", ContourName)	
Description :	Cette fonction vous permet de déplacer le curseur sur l'objet point suivant d'un contour en partant de la position actuelle du curseur sur le contour.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

Remarque

Cette fonction actualise automatiquement l'affichage.

moveCursorOnContourBack – Positionner le curseur sur l'objet graphique précédent d'un contour

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBack**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBack**", ContourName)	
Description :	Cette fonction vous permet de déplacer le curseur sur l'objet point précédent d'un contour en partant de la position actuelle du curseur sur le contour.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	ContourName	Nom du contour (QString). Si aucun contour n'est indiqué, l'appel se rapporte automatiquement au contour actuellement sélectionné.

Remarque

Cette fonction actualise automatiquement l'affichage.

serialize – Enregistrement/restauration de l'état actuel

Syntaxe :	ReturnValue = CallCWMethod(GraphVarName, "**serialize**", FilePath, IsStoring)	
Description :	Cette fonction vous permet, si nécessaire, d'écrire l'état actuel et le contenu du SlEsGraphWidget sous forme binaire dans un fichier ou DataStream, puis de le restaurer à partir du fichier.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	Return Value	Errorcode (bool): TRUE = réussi
	FilePath	Chemin d'accès complet avec nom de fichier (QString)
	IsStoring	Enregistrement/restauration (bool) : TRUE = enregistrer

Remarque

Cette fonction actualise automatiquement l'affichage.

9.5.6 Signaux

Le signal ViewChanged décrit ci-dessous peut être récupéré dans la configuration et il est possible d'y réagir en conséquence.

Vue d'ensemble

Fonction	Description
ViewChanged	Changement de la vue

ViewChanged – Changement de la vue

Syntaxe :	SUB(on_<GraphVarName>**_ViewChanged**) ... END_SUB	
Description :	Si la vue change, le signal "ViewChanged" est envoyé. Il est possible de réagir à ce signal dans la configuration via une certaine méthode SUB. Les paramètres SIGARG sont définis en conséquence.	
Paramètres :	GraphVarName	Nom de la variable d'affichage contenant un SlEsGraph-CustomWidget
	SIGARG[0]	Bord gauche (double)
	SIGARG[1]	Bord supérieur (double)
	SIGARG[2]	Bord droit (double)
	SIGARG[3]	Bord inférieur (double)

Exemple

```
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

Généralités

Run MyScreens vous permet de créer simplement des applications complexes sur le plan fonctionnel. La commande tactile en particulier vous permet de configurer des boutons que vous pouvez positionner librement (boutons tactiles). Les propriétés suivantes s'appliquent à la configuration des boutons tactiles :

- Les deux changements d'état possibles du bouton "enfoncé" et "coché" peuvent être enregistrés dans la configuration et les actions correspondantes déclenchées.
- Les boutons tactiles peuvent être commandés d'une simple pression ou actionnés par commande multitactile, par la souris et le clavier.
- Le bouton tactile est représenté conformément à la résolution actuelle. Cela s'applique entre autres aussi pour la police et l'image d'affichage.
- Le bouton tactile a deux styles de représentation : "Touche logicielle Look&Feel" et "Spécifique à l'utilisateur". En ce qui concerne le style de représentation "Touche logicielle Look&Feel", la représentation du bouton tactile correspond aux touches logicielles d'Operate. Des fonctions telles que la mise à l'échelle des images par exemple sont disponibles pour les deux styles de représentation.
- Le bouton tactile est conçu et mis à disposition en tant que CustomWidget.

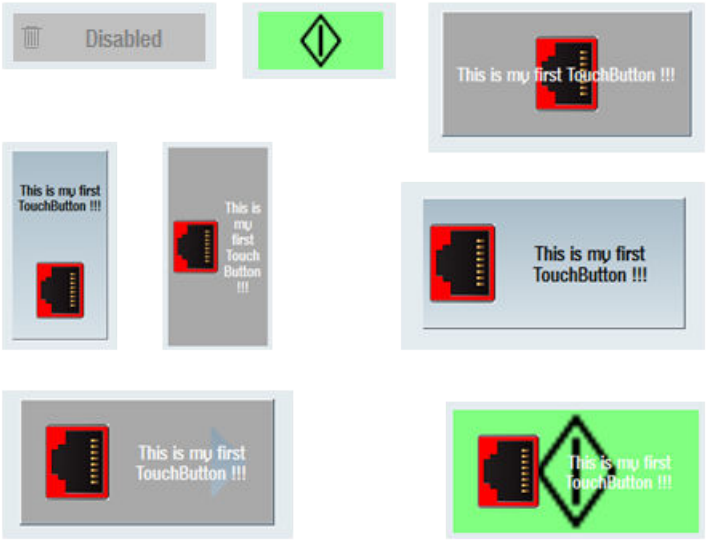


Figure 9-17 Exemples pour le bouton tactile

Exemple

```
//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
```

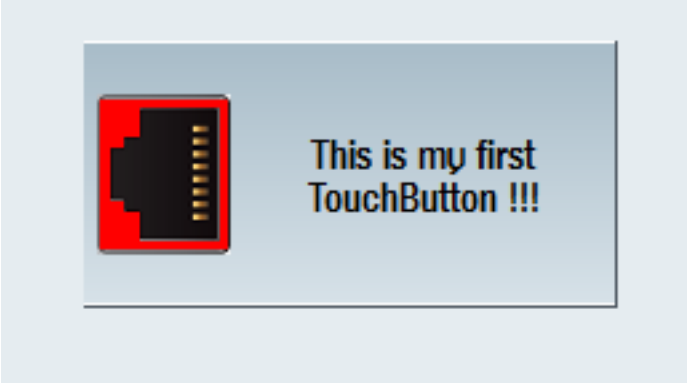


Figure 9-18 Exemple "This is my first TouchButton !!!"

9.6.2 Lecture et écriture de propriétés

Description

Les propriétés présentées dans le chapitre suivant sont lues avec la fonction ReadCWProperty() et définies avec WriteCWProperty().

Exemples

Lecture de la propriété "Text" du SlEsTouchButton lié par la variable d'affichage "MyTouchButton". Le résultat est écrit dans le registre 0.

```
REG[0] = ReadCWProperty("MyTouchButton", "Text")
```

Écriture de la valeur "sk_ok.png" dans la propriété "Picture" du SlEsTouchButton lié par la variable d'affichage "MyTouchButton".

```
WriteCWProperty("MyTouchButton", "Picture", "sk_ok.png")
```

9.6.3 Properties

Vue d'ensemble

Property	Description
ButtonStyle	Style de représentation du bouton tactile
Flat	Représentation plate ou en 3D
Enabled	Activation/désactivation de l'utilisation
Checkable	Activation/désactivation de la fonction de basculement
Checked	Le bouton tactile est enclenché droit (checked)
ShowFocusRect	Afficher le rectangle de focus lorsque le bouton tactile a le focus de saisie
Picture	Image à afficher lorsque le bouton tactile est à l'état normal non enfoncé
PicturePressed	Image à afficher lorsque le bouton tactile est représenté enfoncé ou enclenché (checked)
PictureAlignment PictureAlignmentString	Orientation de l'image
ScalePicture	L'image est mise à l'échelle (étirée ou réduite)
PictureKeepAspectRatio	Comportement d'étirement de l'image
Text	Texte d'affichage normal
TextPressed	Texte d'affichage lorsque le bouton tactile est enfoncé
textAlignment textAlignmentString	Orientation du texte
TextAlignedToPicture	Le texte est orienté par rapport à l'image - activation/désactivation

Property	Description
BackgroundPicture	Image d'arrière-plan à afficher
BackgroundPictureAlignment BackgroundPictureAlignmentString	Orientation de l'image d'arrière-plan
ScaleBackgroundPicture	L'image d'arrière-plan est mise à l'échelle (étirée ou réduite)
BackgroundPictureKeepAspectRatio	Comportement d'étirement de l'image
BackColor	Couleur d'arrière-plan lorsque le bouton tactile est à l'état normal non enfoncé
BackColorChecked	Couleur d'arrière-plan lorsque le bouton tactile est enclenché (checked)
BackColorPressed	Couleur d'arrière-plan lorsque le bouton tactile est enfoncé temporairement
BackColorDisabled	Couleur d'arrière-plan lorsque le bouton tactile n'est pas utilisable
TextColor	Couleur de texte lorsque le bouton tactile est à l'état normal non enfoncé
TextColorChecked	Couleur de texte lorsque le bouton tactile est enclenché (checked)
TextColorPressed	Couleur de texte lorsque le bouton tactile est enfoncé temporairement
TextColorDisabled	Couleur de texte lorsque le bouton tactile n'est pas utilisable

ButtonStyle – Style de représentation

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ButtonStyle**") WriteCWProperty(TouchButtonVarName, "**ButtonStyle**", Value)	
Description :	Lit/définit le style de représentation du bouton tactile.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouchButton
	Return Value	Valeur lue de la propriété (int)
	Value	Valeur à définir (int) 0 = Touche logicielle Look&Feel (default) 1 = Spécifique à l'utilisateur

Avec le réglage "0 = Touche logicielle Look&Feel", le bouton tactile se présente et se comporte tout comme une touche logicielle. L'apparence actuellement réglée est également prise en compte – voir le paramètre machine d'affichage 9112 = $MM_HMI_SKIN.

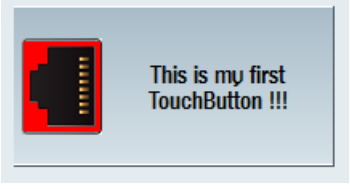


Figure 9-19 ButtonStyle - Touche logicielle Look&Feel

Avec le réglage "1 = spécifique à l'utilisateur", la couleur de texte et d'arrière-plan peut être définie en fonction de l'état du bouton tactile. Par ailleurs, un effet 3D, la mise à l'échelle automatique des images et l'incidence des marges par rapport aux bords sont possibles.

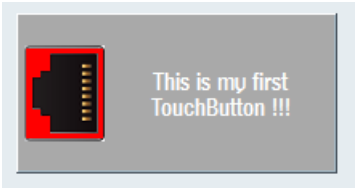


Figure 9-20 ButtonStyle - Spécifique à l'utilisateur

Flat - Mode de représentation

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Flat**") WriteCWProperty(TouchButtonVarName, "**Flat**", Value)	
Description :	Lit/définit si le bouton tactile est représenté à plat (default) ou en 3D **Remarque :** cette propriété n'est disponible qu'avec le style de représentation actif (ButtonStyle) "1 = spécifique à l'utilisateur".	
	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEs-TouchButton
Paramètres :	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE (default) ou FALSE

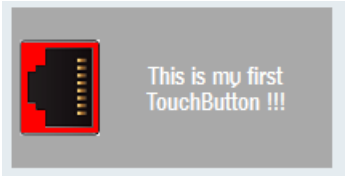


Figure 9-21 Mode de représentation à plat

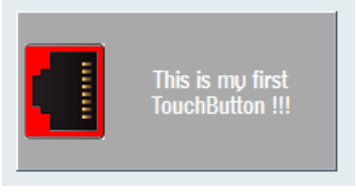


Figure 9-22 Mode de représentation 3D

Enabled – Utilisation du bouton tactile

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Enabled**") WriteCWProperty(TouchButtonVarName, "**Enabled**", Value)	
Description :	Lit/définit si le bouton tactile doit être utilisable (= TRUE) ou non utilisable (= FALSE).	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE (default) ou FALSE

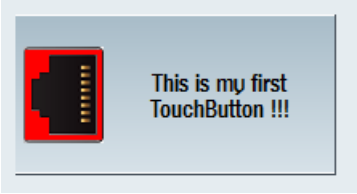


Figure 9-23 Bouton tactile utilisable

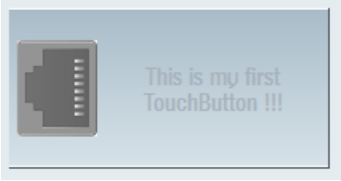


Figure 9-24 Bouton tactile non utilisable

Remarque

Si l'on clique sur un bouton tactile non utilisable, le signal "clickedDisabled" est émis. Le signal peut être évalué et p. ex. un message correspondant indiquant la raison pour laquelle le bouton tactile n'est pas utilisable pour le moment et la fonction associée est affiché. L'image d'affichage ou d'arrière-plan est représentée à l'état désactivé automatiquement en niveaux de gris.

Checkable - Activation/désactivation de la fonction de basculement

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Checkable**") WriteCWProperty(TouchButtonVarName, "**Checkable**", Value)	
Description :	Lit/définit la fonctionnalité de basculement du bouton tactile.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE ou FALSE (default)

Remarque

Généralement, le bouton tactile revient à sont état normal lorsqu'il est relâché (voir touche). Si, en revanche, le fonctionnalité de basculement est activée (TRUE), celui-ci reste en position enfoncée après l'appui. S'il est de nouveau actionné, celui-ci revient à l'état normal (voir commutateur).

Voir aussi la propriété "Checked".

Checked – État de basculement actuel

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Checked**") WriteCWProperty(TouchButtonVarName, "**Checked**", Value)
Description :	Lit/définit l'état de basculement actuel du bouton tactile.

Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE ou FALSE (default)

Si la fonction de basculement est activée par la propriété "Checkable", l'état de basculement actuel peut être lu ou défini avec la propriété "Checked".

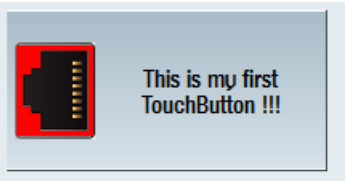


Figure 9-25 Unchecked

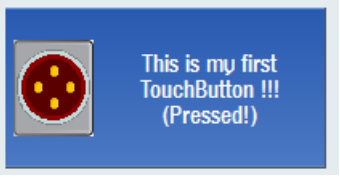


Figure 9-26 Checked

Un texte et une image peuvent être affichés spécifiquement pour les deux états.

Remarque

Voir aussi la propriété "Checkable".

ShowFocusRect – Représentation du rectangle de focus

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ShowFocusRect**") WriteCWProperty(TouchButtonVarName, "**ShowFocusRect**", Value)	
Description :	Lit/définit si le bouton tactile doit représenter un rectangle de focus dès qu'il obtient le focus de saisie.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE ou FALSE (default)

La couleur du cadre de focus est définie automatiquement avec le réglage de couleur Operate, "orange" dans l'exemple suivant.

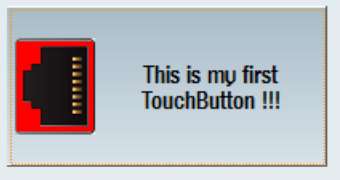


Figure 9-27 ShowFocusRect

Picture – Image d'affichage

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Picture**") WriteCWProperty(TouchButtonVarName, "**Picture**", Value)	
Description :	Lit/définit l'image à afficher lorsque le bouton tactile se trouve dans sa position neutre (non enfoncé).	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	valeur lue de la propriété (String)
	Value	valeur à définir (String)

Le nom de fichier est donné comme valeur, par exemple "sk_ok.png". Le bouton tactile détermine automatiquement le fichier d'image correct dans le répertoire de résolution actuel (voir chapitre Utiliser des images ou des graphiques (Page 66)).

L'image d'affichage est automatiquement représentée à l'état désactivé en niveaux de gris.

Remarque

Voir aussi les propriétés "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PicturePressed – Image d'affichage à l'état enfoncé

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PicturePressed**") WriteCWProperty(TouchButtonVarName, "**PicturePressed**", Value)	
Description :	Lit/définit l'image à afficher lorsque le bouton tactile est à l'état enfoncé ou enclenché.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	valeur lue de la propriété (String)
	Value	valeur à définir (String)

Si aucune valeur n'est indiquée (chaîne vide " "), le bouton tactile affiche dans cet état l'image prédéfinie par la propriété "Picture".

Le nom de fichier est donné comme valeur, par exemple "sk_ok.png". Le bouton tactile détermine automatiquement le fichier d'image correct dans le répertoire de résolution actuel (voir chapitre Utiliser des images ou des graphiques (Page 66)).

L'image d'affichage est automatiquement représentée à l'état désactivé en niveaux de gris.

Remarque

Voir aussi les propriétés "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PictureAlignment – Orientation de l'image

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureAlignment**") WriteCWProperty(TouchButtonVarName, "**PictureAlignment**", Value)	
Description :	Lit/définit l'orientation de l'image d'affichage sur le bouton tactile.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (int)
	Value	Valeur à définir (int) : 1 = gauche (default) 2 = droite 32 = haut 64 = bas 128 = centrée

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).
Voir aussi les propriétés "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PictureAlignmentString – Orientation de l'image

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureAlignmentString**") WriteCWProperty(TouchButtonVarName, "**PictureAlignmentString**", Value)	
Description :	Lit/définit l'orientation de l'image d'affichage. Cette propriété a la même signification que la propriété "PictureAlignment". La valeur n'est cependant pas indiquée ici en chiffres (int), mais sous forme de chaîne.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	valeur lue de la propriété (String)
	Value	valeur à définir (String) : "Left" = gauche (default) "Right" = droite "Top" = haut "Bottom" = bas "Center" = centrée

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

Voir aussi les propriétés "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

ScalePicture – Mise à l'échelle de l'image d'affichage

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ScalePicture**") WriteCWProperty(TouchButtonVarName, "**ScalePicture**", Value)	
Description :	Lit/définit si le bouton tactile doit mettre à l'échelle l'image à afficher en fonction de l'orientation.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE ou FALSE (default)

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

PictureKeepAspectRatio – Mise à l'échelle de l'image d'affichage en fonction du rapport largeur/hauteur

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureKeepAspectRatio**") WriteCWProperty(TouchButtonVarName, "**PictureKeepAspectRatio**", Value)	
Description :	Lit/définit si, le rapport largeur/hauteur de l'image d'affichage doit être conservé ou non lors de la mise à l'échelle (propriété "ScalePicture" = "TRUE").	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE (default) ou FALSE

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

Text – Texte affiché

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Text**") WriteCWProperty(TouchButtonVarName, "**Text**", Value)	
Description :	Lit/définit le texte affiché lorsque le bouton tactile se trouve en position de repos (non enfoncé).	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	valeur lue de la propriété (String)
	Value	valeur à définir (String)

Les textes sont automatiquement coupés en limite de mot dans le rectangle mis à disposition.

Les retours à la ligne ne peuvent être forcés par le système que lorsque le texte affiché provient d'un fichier de langue.

Remarque

Voir chapitre Textes localisés (Page 274).

TextPressed – Texte affiché à l'état enfoncé

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextPressed**") WriteCWProperty(TouchButtonVarName, "**TextPressed**", Value)	
Description :	Lit/définit le texte affiché lorsque le bouton tactile est à l'état enfoncé ou enclenché.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouchButton
	Return Value	valeur lue de la propriété (String)
	Value	valeur à définir (String)

Les textes sont automatiquement coupés en limite de mot dans le rectangle mis à disposition.

Les retours à la ligne ne peuvent être forcés par le système que lorsque le texte affiché provient d'un fichier de langue.

Remarque

Voir chapitre Textes localisés (Page 274).

TextAlignment – Orientation du texte

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextAlignment**") WriteCWProperty(TouchButtonVarName, "**TextAlignment**", Value)	
Description :	Lit/définit l'orientation du texte sur le bouton tactile.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouchButton
	Return Value	Valeur lue de la propriété (int)
	Value	Valeur à définir (int) : 1 = gauche 2 = droite 32 = haut 64 = bas 128 = centrée (default) (voir les exemples ci-dessous)

Figure 9-28 TextAlignment - gauche

Figure 9-29 TextAlignment - droite

Figure 9-30 TextAlignment - haut

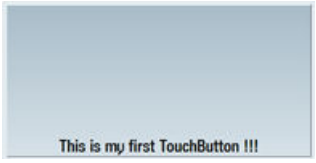


Figure 9-31 TextAlignment - bas

Figure 9-32 TextAlignment - centré

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

Voir aussi les propriétés "Text", "TextAlignmentString", "TextAlignedToPicture".

TextAlignmentString – Orientation du texte

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextAlignmentString**") WriteCWProperty(TouchButtonVarName, "**TextAlignmentString**", Value)
Description :	Lit/définit l'orientation du texte. Cette propriété a la même signification que la propriété "TextAlignment". La valeur n'est cependant pas indiquée en chiffres (int), mais sous forme de chaîne.

Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	valeur lue de la propriété (String)
	Value	valeur à définir (String) : "Left" = gauche "Right" = droite "Top" = haut "Bottom" = bas "Center" = centrée (default)

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

Voir aussi les propriétés "Text", "TextAlignment", "TextAlignedToPicture".

TextAlignedToPicture – Orientation du texte par rapport à l'image

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "TextAlignedToPicture") WriteCWProperty(TouchButtonVarName, "**TextAlignedToPicture**", Value)	
Description :	Lit/définit si le texte affiché doit être positionné par rapport à l'image. Si FALSE est défini ici, le texte est représenté centré sur le bouton tactile.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE (default) ou FALSE

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).
Voir aussi les propriétés "Text", "TextPressed", "Picture", "PicturePressed".

BackColor – Couleur d'arrière-plan

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, " **BackColor**") WriteCWProperty(TouchButtonVarName, " **BackColor**", Value)	
Description :	Lit/définit la couleur d'arrière-plan lorsque le bouton tactile est dans sa position neutre (non enfoncé).	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	couleur lue de la propriété (String)
	Value	valeur à définir (String) comme valeur RVB de la forme "#RRGGBB", par ex. "#04B7FB"

Remarque

Cette propriété n'est disponible qu'avec le style de représentation actif "1 = spécifique à l'utilisateur".

BackColorChecked – Couleur d'arrière-plan à l'état enclenché

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorChecked**") WriteCWProperty(TouchButtonVarName, " **BackColorChecked**", Value)	
Description :	Lit/définit la couleur d'arrière-plan lorsque le bouton tactile est à l'état enclenché (fonction de basculement). Propriétés "Checked", "Checkable".	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	couleur lue de la propriété (String)
	Value	valeur à définir (String) comme valeur RVB de la forme "#RRGGBB", par ex. "#04B7FB"

Remarque

Cette propriété n'est disponible qu'avec le style de représentation actif "1 = spécifique à l'utilisateur".

Remarque

Voir aussi les propriétés "Checked", "Checkable".

BackColorPressed – Couleur d'arrière-plan à l'état enfoncé

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorPressed**") WriteCWProperty(TouchButtonVarName, " **BackColorPressed**", Value)	
Description :	Lit/définit la couleur d'arrière-plan lorsque le bouton tactile est à l'état enfoncé.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	couleur lue de la propriété (String)
	Value	valeur à définir (String) comme valeur RVB de la forme "#RRGGBB" par ex. "#04B7FB"

Remarque

Cette propriété n'est disponible qu'avec le style de représentation actif "1 = spécifique à l'utilisateur".

BackColorDisabled – Couleur d'arrière-plan à l'état désactivé

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorDisabled**") WriteCWProperty(TouchButtonVarName, " **BackColorDisabled**", Value)	
Description :	Lit/définit la couleur d'arrière-plan lorsque le bouton tactile est à l'état désactivé.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	couleur lue de la propriété (String)
	Value	valeur à définir (String) comme valeur RVB de la forme "#RRGGBB", par ex. "#04B7FB"

Remarque

Cette propriété n'est disponible qu'avec le style de représentation actif "1 = spécifique à l'utilisateur".

Remarque

Voir aussi la propriété "Enabled".

TextColor – Couleur du texte

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColor**") WriteCWProperty(TouchButtonVarName, " **TextColor**", Value)	
Description :	Lit/définit la couleur du texte lorsque le bouton tactile est dans sa position neutre (non enfoncé).	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	couleur lue de la propriété (String)
	Value	valeur à définir (String) comme valeur RVB de la forme "#RRGGBB", par ex. "#04B7FB"

Remarque

Cette propriété n'est disponible qu'avec le style de représentation actif "1 = spécifique à l'utilisateur".

TextColorChecked – Couleur du texte à l'état enclenché

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorChecked**") WriteCWProperty(TouchButtonVarName, " **TextColorChecked**", Value)
Description :	Lit/définit la couleur du texte lorsque le bouton tactile est à l'état enclenché (fonction de basculement).

Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	couleur lue de la propriété (String)
	Value	valeur à définir (String) comme valeur RVB de la forme "#RRGGBB", par ex. "#04B7FB"

Remarque

Cette propriété n'est disponible qu'avec le style de représentation actif "1 = spécifique à l'utilisateur".

Remarque

Voir aussi les propriétés "Checked", "Checkable".

TextColorPressed – Couleur du texte à l'état enfoncé

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorPressed**") WriteCWProperty(TouchButtonVarName, "**TextColorPressed**", Value)	
Description :	Lit/définit la couleur du texte lorsque le bouton tactile est à l'état enfoncé.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	couleur lue de la propriété (String)
	Value	valeur à définir (String) comme valeur RVB de la forme "#RRGGBB", par ex. "#04B7FB"

Remarque

Cette propriété n'est disponible qu'avec le style de représentation actif "1 = spécifique à l'utilisateur".

TextColorDisabled – Couleur du texte à l'état désactivé

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorDisabled**") WriteCWProperty(TouchButtonVarName, "**TextColorDisabled**", Value)	
Description :	Lit/définit la couleur du texte lorsque le bouton tactile est à l'état désactivé.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	couleur lue de la propriété (String)
	Value	valeur à définir (String) comme valeur RVB de la forme "#RRGGBB", par ex. "#04B7FB"

Remarque

Cette propriété n'est disponible qu'avec le style de représentation actif "1 = spécifique à l'utilisateur".

Remarque

Voir aussi la propriété "Enabled".

BackgroundPicture – Image d'arrière-plan

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPicture**") WriteCWProperty(TouchButtonVarName, "**BackgroundPicture**", Value)	
Description :	Lit/définit l'image d'arrière-plan permanente à afficher, c.-à-d. indépendante de l'état.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	valeur lue de la propriété (String)
	Value	valeur à définir (String)

Le nom de fichier est donné comme valeur, par exemple "sk_ok.png". Le bouton tactile détermine automatiquement le fichier d'image correct dans le répertoire de résolution actuel (voir chapitre Utiliser des images ou des graphiques (Page 66)).

L'image d'arrière-plan est automatiquement représentée à l'état désactivé en niveaux de gris.

Remarque

Voir chapitre Utiliser des images ou des graphiques (Page 66).

Voir aussi les propriétés "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureAlignment – Orientation de l'image d'arrière-plan

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureAlignment**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureAlignment**", Value)
Description :	Lit/définit l'orientation de l'image d'arrière-plan.

Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouchButton
	Return Value	Valeur lue de la propriété (int)
	Value	Valeur à définir (int) : 1 = gauche 2 = droite 32 = haut 64 = bas 128 = centrée (default)

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

Voir aussi les propriétés "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureAlignmentString – Orientation de l'image d'arrière-plan

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureAlignmentString**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureAlignmentString**", Value)	
Description :	Lit/définit l'orientation de l'image d'arrière-plan. Cette propriété a la même signification que la propriété "BackgroundPictureAlignment". La valeur n'est cependant pas indiquée ici en chiffres (int), mais sous forme de chaîne.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouchButton
	Return Value	Valeur lue de la propriété (int)
	Value	Valeur à définir (int) : "Left" = gauche "Right" = droite "Top" = haut "Bottom" = bas "Center" = centrée (default)

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

Voir aussi les propriétés "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

ScaleBackgroundPicture – Mise à l'échelle de l'image d'arrière-plan

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ScalePicture**") WriteCWProperty(TouchButtonVarName, "**ScalePicture**", Value)	
Description :	Lit/définit si le bouton tactile doit mettre à l'échelle l'image à afficher en fonction de l'orientation.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE ou FALSE (default)

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

Voir aussi les propriétés "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureKeepAspectRatio – Mise à l'échelle de l'image d'arrière-plan en fonction du rapport largeur/hauteur

Syntaxe :	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPicture-KeepAspectRatio**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureKeepAspectRatio**", Value)	
Description :	Lit/définit si le rapport largeur/hauteur de l'image d'arrière-plan doit être conservé ou non lors de la mise à l'échelle (propriété "ScaleBackgroundPicture" = "TRUE").	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	Return Value	Valeur lue de la propriété (bool)
	Value	Valeur à définir (bool) : TRUE (default) ou FALSE

Remarque

Voir chapitre Positionnement et orientation de l'image et du texte (Page 272).

9.6.4 Fonctions

Vous pouvez appeler les fonctions indiquées ci-dessous avec la fonction CallCWMethod().

Exemple

Définition des marges de SlEsTouchButton lié par la variable d'affichage "MyTouchButton".

Le résultat est écrit dans le registre 0.

```
REG[0] = CallCWMethod("MyTouchButton", "setMargins", 20, 20, 20, 20,
20)
```

Vue d'ensemble

Fonction	Description
setMargins	Réglage des marges
serialize	Enregistrement/restauration de l'état actuel

setMargins – Réglage des marges

Syntaxe :	ReturnValue = CallCWMethod(TouchButtonVarName, "**setMargins**", left, top, right, bottom, center) ReturnValue = CallCWMethod(TouchButtonVarName, "**setMargins**", left, top, right, bottom, center, marginAlignment)	
Description :	Cette fonction permet de définir les marges du cadre et la distance entre l'image et le texte. Si une valeur de "-1" est indiquée, la marge par défaut du système s'applique pour cette valeur. Si la valeur est supérieure ou égale à "0", la valeur est définie comme marge de bordure.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouchButton
	Return Value	Errorcode (bool): TRUE = réussi
	left	bord gauche (int)
	top	bord supérieur (int)
	right	bord droit (int)
	bottom	bord inférieur (int)
	center	Distance entre l'image et le texte (int) lorsque Alignment n'est pas "center"
	marginAlignment	en option : détermine si la marge doit aussi agir pour le positionnement du texte affiché en plus de l'image d'affichage (bool), TRUE (default)

serialize – Enregistrement/restauration de l'état actuel

Syntaxe :	ReturnValue = CallCWMethod(TouchButtonVarName, "**serialize**", FilePath, IsStoring)	
Description :	Cette fonction permet d'écrire et de restaurer si nécessaire l'état actuel du SlEsTouchButton binairement dans un fichier ou DataStream. Cette fonction actualise automatiquement l'affichage.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouchButton
	Return Value	Errorcode (bool): TRUE = réussi
	FilePath	Chemin d'accès complet avec nom de fichier (QString)
	IsStoring	Enregistrement/restauration (bool) : TRUE = enregistrer

Remarque

La fonction n'est pas utilisée directement par le configurateur !

9.6.5 Signaux

Les signaux décrits ci-après peuvent être récupérés dans la configuration et il est possible d'y réagir en conséquence.

Vue d'ensemble

Fonction	Description
clickedDisabled	Actionnement d'un bouton tactile désactivé
clicked	Un clic ou un appui a été effectué
checked	Un basculement a été effectué

- Si un bouton tactile ne peut pas être utilisé ou basculé, le signal "clicked" est envoyé lors de l'actionnement.
- Si un bouton tactile ne peut pas être utilisé ou basculé, la séquence de signaux suivante est envoyée lors de l'actionnement :
  "checked" → "clicked"
- Si un bouton tactile ne peut pas être utilisé, le signal "clickedDisabled" est envoyé lors de l'actionnement.

Tous les signaux mentionnés ci-dessus ne sont envoyés qu'après une action opérateur, c.-à-d. uniquement lorsque la souris ou la touche espace est relâchée ou après exécution d'un appui avec la commande multitactile.

Le SlEsTouchButton a ainsi une fonctionnalité de commutateur, c.-à-d. qu'aucune fonctionnalité de touche ne peut ainsi être effectuée.

Remarque

Pour la commande multitactile, notez qu'un clic n'est envoyé que lorsque un geste d'appui (appuyer et relâcher en l'espace d'env. 0,7 seconde) a été totalement détecté. Si une action a déjà été effectuée suite à une simple pression, par ex. si un défilement/déplacement de la zone de défilement en arrière-plan était impossible, cela pourrait aboutir à une erreur de manipulation involontaire.

clicked – Un clic sur le bouton tactile a été effectué

Syntaxe :	SUB(on_<TouchButtonVarName>_clicked) ... END_SUB	
Description :	Une séquence complète d'appui et de relâchement d'un bouton tactile utilisable émet un signal "clicked". Il est recommandé de travailler principalement avec ce signal.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	SIGARG[0]	fournit l'état de basculement du bouton tactile (bool)

Exemple

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_clicked)
   DLGL("checked: " << SIGARG[0])
END_SUB
```

checked – Un basculement a été effectué sur un bouton tactile

Syntaxe :	SUB(on_<TouchButtonVarName>_checked) ... END_SUB	
Description :	Un bouton tactile pouvant être basculé a été enfoncé, son état s'étant ainsi modifié.	
Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	SIGARG[0]	fournit l'état de basculement du bouton tactile (bool)

Exemple

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_toggled)
   DLGL("toggled: " << SIGARG[0])
END_SUB
```

clickedDisabled – Un clic a été effectué sur un bouton tactile non utilisable

Syntaxe :	SUB(on_<TouchButtonVarName>_clickedDisabled) ... END_SUB
Description :	Une séquence complète d'appui et de relâchement d'un bouton tactile non utilisable émet un signal "clickedDisabled".

Paramètres :	TouchButtonVarName	Nom de la variable d'affichage contenant un SlEsTouch-Button
	SIGARG[0]	fournit l'état de basculement du bouton tactile (bool)

Exemple

```
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 Positionnement et orientation de l'image et du texte

Positionnement des images

Une image est positionnée comme suit avec l'orientation prédéfinie (Alignment) :

- La première étape consiste à déterminer l'image correspondant à la résolution actuelle dans le répertoire de résolution correspondant.
- Le rectangle de surface (ClientArea) est ensuite réduit afin de correspondre aux marges prédéfinies par rapport aux bords.
  La zone de marge (MarginArea) peut être modifiée avec la fonction "setMargins" :
  - setMargins(-1, -1, -1, -1, -1)

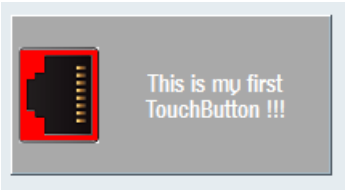


Figure 9-33 setMargins(-1, -1, -1, -1, -1)

  - setMargins(0, 0, 0, 0, 0)

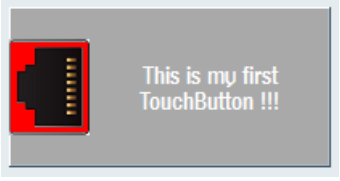


Figure 9-34 setMargins(0, 0, 0, 0, 0)

  - setMargins(20, 20, 20, 20, 20)

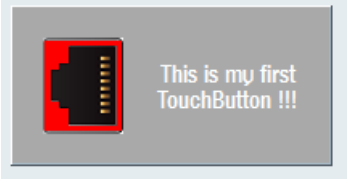


Figure 9-35 setMargins(20, 20, 20, 20, 20)

Exemple d'orientation "gauche"

La surface est partagée en deux horizontalement de manière à ce que l'image à afficher puisse prendre la moitié de la surface.

L'image se présente alors comme suit :

- Propriété "scalePicture" : FALSE
  L'image se présente sans aucune mise à l'échelle à cet endroit. Il faut veiller ici à ce que l'image de base ne soit pas trop grande et puisse vraiment être totalement représentée à l'intérieur du bouton tactile.

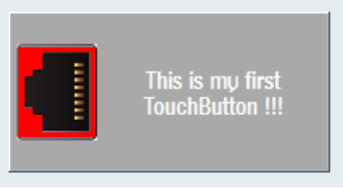


Figure 9-36 scalePicture FALSE

- Propriété "scalePicture" : TRUE
  L'image est mise à l'échelle (étirée ou réduite) jusqu'à ce qu'elle s'adapte dans la moitié gauche du bouton tactile. La propriété "pictureKeepAspectRatio" est alors prise en compte comme suit :
  - Propriété "pictureKeepAspectRatio" : FALSE
    L'image est mise à l'échelle horizontalement et verticalement de manière à s'adapter précisément dans la moitié gauche du bouton tactile. Le rapport largeur/hauteur de l'image d'origine n'est alors plus pris en compte.

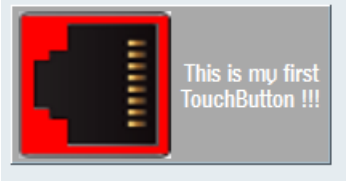


Figure 9-37 scalePicture - pictureKeepAspectRatio FALSE

  - Propriété "pictureKeepAspectRatio" : TRUE
    L'image est mise à l'échelle à la taille maximale dans la moitié gauche du bouton tactile en tenant compte du rapport largeur/hauteur de l'image d'origine.

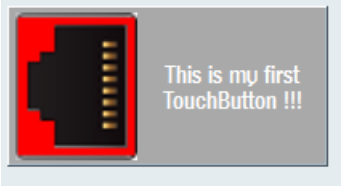


Figure 9-38 scalePicture - pictureKeepAspectRatio TRUE

Remarque

Le même principe s'applique pour les orientations "droite", "haut" et "bas".

Pour l'orientation "centrée", on essaie d'adapter l'image en fonction des propriétés "scalePicture" et "pictureKeepAspectRatio" dans toute la zone de MarginArea.

Positionnement du texte

En fonction de la propriété "textAlignedToPicture", le texte est positionné comme suit :

- Propriété "textAlignedToPicture" : FALSE
  Le texte est affiché de manière centrée verticalement et horizontalement dans MarginArea.

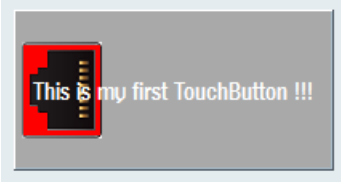


Figure 9-39 textAlignedToPicture FALSE

- Propriété "textAlignedToPicture" : TRUE
  Le texte est affiché de manière centrée horizontalement et verticalement dans la zone restante de MarginArea moins la zone de l'image représentée.

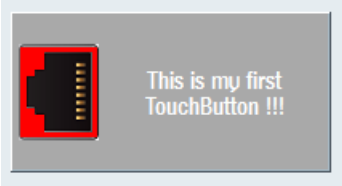


Figure 9-40 textAlignedToPicture TRUE

9.6.7 Textes localisés

Le SlEsTouchButton n'assure aucune prise en charge propre des langues étrangères. La manière de localiser est cependant illustrée dans l'exemple suivant :

Exemple

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 !!!"
```

Fichier de configuration :

```
//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
```

Les textes sont automatiquement coupés en limite de mot dans le rectangle mis à disposition. Un retour à la ligne forcé n'est possible que si le texte à afficher provient d'un fichier de langue. Pour cela, il faut insérer "%n" à l'endroit correspondant du texte (voir exemple suivant).

user_eng.txt:

```
85001 0 0 "This is%nmy%nfirst%nTouchButton !!!"
```

Fichier de configuration :

```
WRITECWPROPERTY("MyTB1", "text", $85001)
```

Résultat

Figure 9-41 Exemple de retour à la ligne dans un texte localisé

Groupe fonctionnel "Custom" 10

10.1 Activation du groupe fonctionnel "Custom"

Activation du groupe fonctionnel "Custom"

Le groupe fonctionnel "Custom" n'est pas activé à la livraison.

1. Copier d'abord le fichier "slamconfig.ini" du répertoire [Répertoire système siemens]/cfg dans le répertoire [Répertoire système oem]/cfg
   ou respectivement dans [Répertoire système addon]/cfg ou [Répertoire système user]/cfg.
2. Pour activer le groupe fonctionnel "Custom", l'entrée suivante est nécessaire :
   ```
   [Custom]
   Visible=True
   ```

Résultat

Après activation, la touche logicielle du groupe fonctionnel "Custom" figure dans le menu principal (F10), sur la barre de commutation de menu sur la TLH4 (= préréglage).

Le groupe fonctionnel "Custom" affiche une fenêtre vide, dont le titre est configurable, au-dessus de l'ensemble du groupe fonctionnel. Toutes les touches logicielles horizontales et verticales sont configurables.

10.2 Configuration de la touche logicielle pour "Custom"

Configuration de la touche logicielle pour le groupe fonctionnel "Custom"

La légende et la position de la touche logicielle pour le groupe fonctionnel "Custom" sont configurées dans le fichier "slamconfig.ini".

La configuration de la touche logicielle d'accès peut s'effectuer des façons suivantes :

1. Pour remplacer la légende par un **texte localisé** sur la touche logicielle, les entrées suivantes sont nécessaires dans la section [Custom] :
   ```
   TextId=MY_TEXT_ID
   TextFile=mytextfile
   TextContext=mycontext
   ```
   Dans cet exemple, la touche logicielle affiche le texte localisé consigné sous l'ID de texte "MY_TEXT_ID" dans le fichier de texte mytextfile_xxx.qm sous "MyContext" (xx étant l'identifiant de langue).
2. Pour remplacer la légende par un **texte indépendant de la langue** sur la touche logicielle, les entrées suivantes sont nécessaires dans la section [Custom] :
   ```
   TextId=HELLO
   TextFile=<empty>
   TextContext=<empty>
   ```
   Dans cet exemple, la touche logicielle pour le groupe fonctionnel "Custom" affiche le texte "HELLO" dans toutes les langues.
3. Outre le texte, la touche logicielle peut également afficher un **pictogramme**.
   Pour cela, l'entrée suivante est nécessaire dans la section [Custom] :
   ```
   Picture=mypicture.png
   ```
   La touche logicielle affiche alors le fichier mypicture.png sous forme de pictogramme. Les graphiques et les bitmaps sont enregistrés dans le répertoire suivant : [Répertoire système oem]/ico/ico<résolution>. Ils doivent être placés dans le répertoire correspondant à la résolution de l'écran.
4. En outre, il est également possible de configurer la **position** de la touche logicielle. Pour cela, adapter l'entrée suivante dans la section [Custom] :
   ```
   SoftkeyPosition=12
   ```
   La position par défaut est 12. Elle correspond à la TLH4 sur la barre de commutation de menu du menu groupe fonctionnel. Les positions 1-8 correspondent aux TLH1-TLH8 sur la barre de menus, les positions 9-16 correspondent aux TLH1-TLH8 sur la barre de commutation de menu.

10.3 Configuration du groupe fonctionnel "Custom"

Configuration de la touche logicielle pour le groupe fonctionnel "Custom"

Les fichiers "easyscreen.ini" et "custom.ini" sont nécessaires pour configurer le groupe fonctionnel. Des modèles de ces deux fichiers sont disponibles dans le répertoire [Répertoire système siemens]/templates/cfg.

1. Les fichiers doivent d'abord être copiés dans le répertoire [Répertoire système oem]/cfg avant d'être modifiés dans celui-ci.
2. Le fichier "easyscreen.ini" contient déjà une ligne de définition pour le groupe fonctionnel "Custom" :
   ```
   ;StartFile02 = area := Custom, dialog := SlEsCustomDialog,
   startfile := custom.com
   ```
   Le ";" au début de la ligne est un caractère de commentaire. La ligne est donc mise en commentaire et n'est de ce fait pas active. Pour l'activer, il faut supprimer le ";".
   L'attribut "startfile" dans cette ligne permet de définir que l'entrée doit renvoyer au fichier de projet custom.com lorsque le groupe fonctionnel "custom.com" est sélectionné.
3. Le **fichier de projet "custom.com"** doit être créé dans le répertoire [Répertoire système oem]/proj. Il contient la configuration, qui est effectuée de façon similaire à celle du fichier "aeditor.com" du groupe fonctionnel "Programme". Une fois configurées, les touches logicielles d'accès s'affichent dans le groupe fonctionnel "Custom".
4. Le **texte indépendant de la langue** pour la ligne de titre de la boîte de dialogue se configure dans le fichier "custom.ini".
   Le modèle contient à cet effet l'entrée suivante :
   ```
   [Header]
   Text=Custom
   ```
   Ce texte peut être remplacé par un texte personnalisé.
5. Pour configurer l'**image d'accueil** du groupe fonctionnel "Custom", utiliser l'entrée suivante, disponible dans le modèle :
   ```
   [Picture]
   Picture=logo.png
   ```
   Logo.png est le nom de l'image qui s'affiche dans la boîte de dialogue d'accueil du groupe fonctionnel "Custom". Celle-ci permet d'afficher, par exemple, un logo d'entreprise ou toute autre image. Le fichier doit être enregistré dans le répertoire correspondant à la résolution : [Répertoire système oem]/ico/ ...
6. Pour afficher directement un certain masque "Run MyScreens" lors du premier affichage du groupe fonctionnel "Custom", le modèle contient l'entrée suivante :
   ```
   ; Mask shown with startup of area "custom"
   [Startup]
   ;Startup = Mask:=MyCustomStartupMask, File:=mycustommasks.com
   ```
   Le cas d'utilisation typique est, par exemple, le démarrage direct avec un certain masque "Run MyScreens" lors du démarrage de SINUMERIK Operate.
   Pour cela, la clé "startuparea" doit être réglée comme suit dans le fichier de configuration "systemconfiguration.ini" dans la section "[miscellaneous]" :
   ```
   [miscellaneous]
   ;name of the area to be shown at system startup
   startuparea = Custom
   ```

10.4 Exemple de programmation pour le groupe "Custom"

Exemple

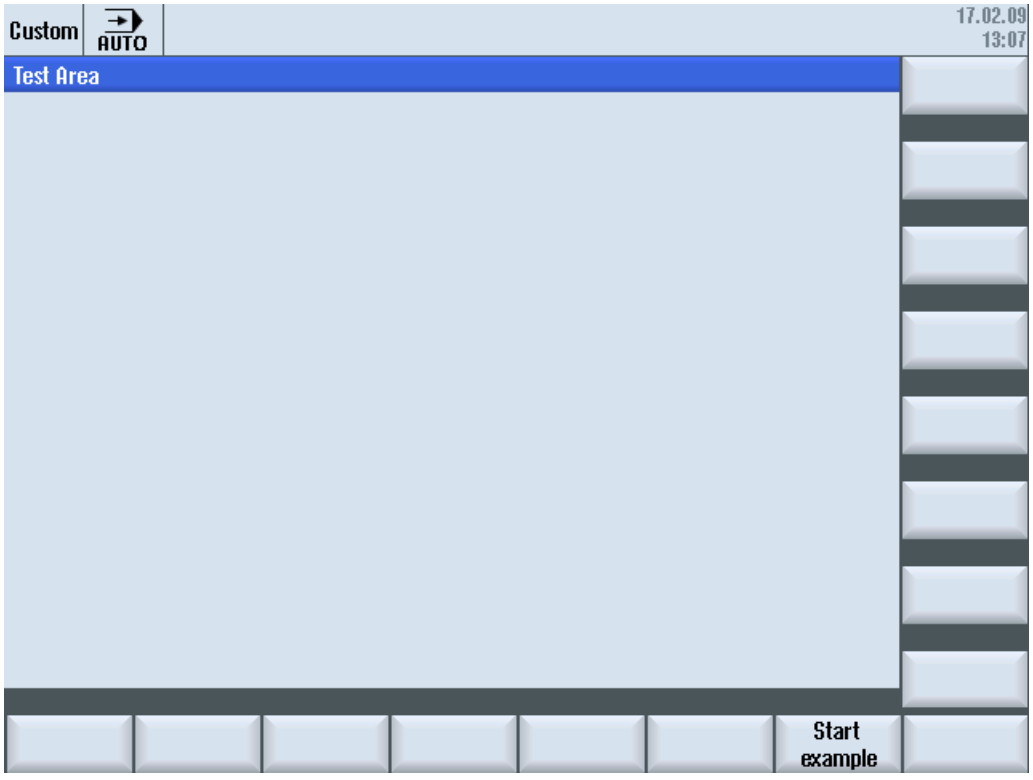


Figure 10-1 Exemple avec la touche logicielle "Start example"

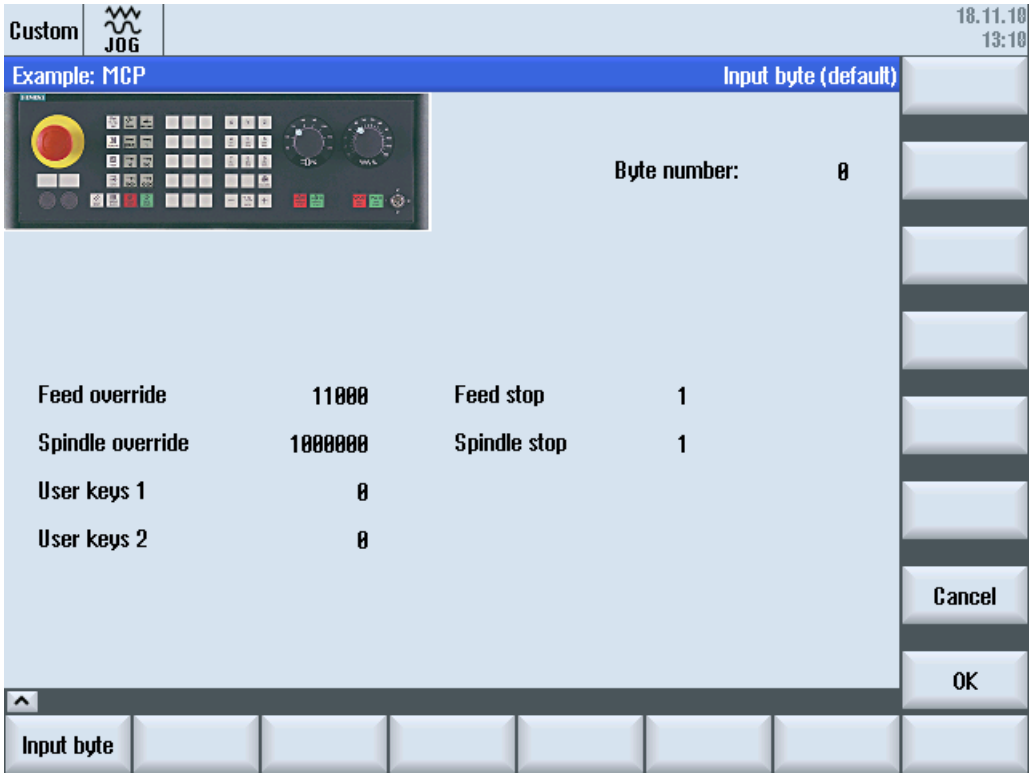


Figure 10-2 Exemple avec bitmap et champs de texte

Aperçu du fichier

Les fichiers suivants sont nécessaires :

- "custom.com"
- "easyscreen.ini"

Programmation

Contenu du fichier "custom.com" :

Remarque

Le fichier graphique intégré "mcp.png" est fourni à titre d'exemple. Si vous voulez reprogrammer cet exemple, vous devez remplacer le graphique par celui dont vous disposez.

```
//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
```

Sélection d'une boîte de dialogue 11

11.1 Sélection d'une boîte de dialogue avec les touches logicielles de l'AP

Configuration

Description de la procédure :

- Le fichier "systemconfiguration.ini" contient une section [keyconfiguration]. L'entrée spécifie une action pour une touche logicielle d'AP spécifique.
- L'action est désignée par un numéro. Si celui-ci est supérieur ou égal à 100, il s'agit d'un appel "Run MyScreens".
- Pour définir l'action à exécuter, il convient de créer dans le fichier "easyscreen.ini" une section dont le nom est formé du nom de groupe fonctionnel et du nom de boîte de dialogue (voir entrée sous [keyconfiguration] → Area:=..., Dialog:=...) → [<Area>_<Dialog>] → par ex. [AreaParameter_SlPaDialog]
- Les numéros d'action (qui ont été spécifiés dans le fichier "systemconfiguration.ini" → voir Action:=...) sont définis dans cette section. Il s'agit de deux commandes :
  1. LS("Barre de touches logicielles1","param.com") ... Chargement d'une barre de touches logicielles
  2. LM("Masque1","param.com") ... Chargement d'un masque

Sélection de barres de touches logicielles via les touches logicielles de l'AP

Pour "Run MyScreens", les barres de touches logicielles et les boîtes de dialogue "Run MyScreens" peuvent être sélectionnées via les touches logicielles de l'AP. A cet effet, l'attribut "action" à spécifier pour la configuration des touches logicielles d'AP concernées doit avoir une valeur supérieure ou égale à 100.

La configuration des touches logicielles d'AP s'effectue dans le fichier "systemconfiguration.ini", à la section [keyconfiguration] :

```
[keyconfiguration]
KEY75.1 = Area:=area, Dialog:=dialog, Screen:=screen, Action:= 100,
```

La configuration des ordres LM et LS, qui doivent s'exécuter lors de l'activation des touches logicielles d'AP correspondantes, s'effectue dans le fichier "easyscreen.ini" et dans les sections dont le nom est structuré comme suit :

`[areaname_dialogname]`	La première partie du nom "areaname" désigne le groupe fonctionnel, la deuxième partie "dialogname" désigne la boîte de dialogue à laquelle s'appliquent les ordres configurés dans cette section.

`[AreaParameter_SlPaDialog]`
`100.screen1 = LS("Touche logicielle1","param.com")`
`101.screen3 = LM("Masque1","param.com")`
	Il convient d'utiliser les noms attribués pour le groupe fonctionnel et la boîte de dialogue dans le fichier "systemconfiguration.ini" L'indication de la boîte de dialogue est facultative.
	Elle peut être omise, notamment pour les groupes fonctionnels implémentés par une seule et unique boîte de dialogue. Voir l'exemple ci-contre.
	Si "screen1" est affiché dans le groupe fonctionnel AreaParameter implémenté par la boîte de dialogue SlPaDialog, l'ordre "LS("Softkey1","param.com")" est exécuté lorsque l'attribut "action" se voit appliquer la valeur 100.

`action.screen=ordre`	Les deux attributs "action" et "screen" indiquent de manière univoque quand l'ordre spécifié sera exécuté.
	L'indication de "screen" est facultative.
	Les ordres admissibles sont les suivantes :
	`LM (LoadMask)`
	`LS (LoadSoftkeys)`

11.1.1 Appel de masques de cycle Run MyScreens- dans l'éditeur de programme

Domaine d'application

Les touches de l'AP offrent la possibilité d'ouvrir directement les masques de cycle Run MyScreens dans l'éditeur de programme.

Condition supplémentaire

Un cycle technologique peut être sélectionné uniquement lorsque l'éditeur de programme est ouvert. Le cycle technologique paramétré via le masque est inséré à la position actuelle du curseur dans l'éditeur de programme.

Marche à suivre

1. Créer l'appel de cycle Run MyScreens via les touches matérielles (Page 288) de l'AP.
2. Pour que le masque Run MyScreens puisse être correctement intégré dans l'éditeur de programme, la configuration des touches de l'AP doit être complétée comme décrit dans l'exemple suivant :
   Exemple de configuration complète des touches :
   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. Le retour d'information du masque de cycle Run MyScreens ouvert peut être vérifié via l'interface AP "DB19.DBW24".
   Exemple :
   Dans cet exemple, la valeur "9876" est écrite sur l'interface AP lorsque le masque Run MyScreens est ouvert :
   Configuration avec ID AP=9876 :
   //M(MASK_E_DR_O1_MAIN/$85305/////XG1,FA2,TA7//"drilling.txt"/**9876**)
   ou
   //M{ MASK_E_DR_O1_MAIN, HD=$85305, LANGFILELIST=" drilling.txt", **PLC_ID=9876**}
   Voir aussi : Description fonctionnelle SINUMERIK 840D sl Fonctions de base

État de l'éditeur de programme sur l'interface OA

L'état de l'éditeur de programme peut être interrogé via la variable locale CAP "/Hmi/StepEditorInfo".

Nom de la variable :	/Hmi/StepEditorInfo
Description de la variable :	Informations de l'éditeur concernant le programme ayant le focus.
La chaîne de caractères comporte les informations suivantes	
[fileName]	Chemin d'accès au programme
[channel]	Numéro de canal
[technlogy]	Technologie
[appearance]	Identificateur du programme (ShopMill/ ShopTurn/G-Code)
[state]	État de StepEditor : • EditorClosed: l'éditeur est fermé (p. ex. IHM absent du groupe fonctionnel Programme) • EditEnabled: édition autorisée • EditDisabled: édition non autorisée (p. ex. lecture seule ou position du curseur non valide)

11.2 Sélection d'une boîte de dialogue via les touches matérielles de l'AP

Domaine d'application

Les fonctions suivantes peuvent être initiées dans le logiciel de commande à partir de l'AP :

- Sélection de groupe fonctionnel
- Sélection de certains scénarios au sein des groupes fonctionnels
- Exécution de fonctions configurées sur des touches logicielles

Touches matérielles

Toutes les touches (y compris les touches de l'AP) sont désignées par touches matérielles ci-après. Jusqu'à 254 touches matérielles au maximum peuvent être définies. La répartition suivante s'applique dans ce contexte :

Numéro de touche	Utilisation
Touche 1 - touche 9	Touches du pupitre opérateur
Touche 10 - touche 49	Réservées
Touche 50 - touche 254 Touche 50 – touche 81	Touches AP : réservées pour l'OEM
Touche 255	Affectée par défaut par une information de commande.

Les touches matérielles 1 - 9 sont affectées par défaut comme suit :

Désignation de touche		Action/effet
HK1	MACHINE	Sélection du groupe fonctionnel "Machine", dernière boîte de dialogue
HK2	PROGRAM	Sélection du groupe fonctionnel "Programme", dernière boîte de dialogue ou dernier programme
HK3	OFFSET	Sélection du groupe fonctionnel "Paramètres", dernière boîte de dialogue
HK4	PROGRAM MANAGER	Sélection du groupe fonctionnel "Programme", vue racine "Gestionnaire de programmes"
HK5	ALARM	Sélection du groupe fonctionnel "Diagnostic", boîte de dialogue "Liste des alarmes"
HK6	CUSTOM	Sélection du groupe fonctionnel "Custom"
HK7 [1)]	MENU SELECT	Sélection "Menu principal"
HK8 [1)]	MENU FUNCTION	Sélection du groupe fonctionnel "Function"
HK9 [1)]	MENU USER	Sélection du groupe fonctionnel "User"

1) uniquement pour 828D

Configuration

La configuration s'effectue dans le fichier de configuration "systemconfiguration.ini" à la section [keyconfiguration]. Chaque ligne définit un événement appelé événement de touche matérielle. Un événement de touche matérielle désigne le nième actionnement d'une certaine touche matérielle. Par exemple, le deuxième et le troisième actionnement d'une certaine touche matérielle peuvent conduire à des réactions différentes.

Les entrées du fichier de configuration "systemconfiguration.ini" peuvent être écrasées par des réglages personnalisés. Les répertoires [Répertoire système user]/cfg et [Répertoire système oem]/cfg sont disponibles à cet effet.

Les lignes de configuration des événements de touche matérielle ont la structure suivante :

```
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 : numéro de la touche matérielle, plage de valeurs 1 – 254

n : numéro d'événement, correspond au nième actionnement de la touche matérielle, plage de valeurs 0 – 9

Condition

Le programme AP utilisateur doit remplir la condition suivante :

Une seule touche matérielle est traitée à la fois. C'est pourquoi une nouvelle demande ne peut être émise que lorsque le logiciel de commande a acquitté la précédente demande. Lorsque le programme AP utilisateur dérive la touche matérielle à partir d'une touche du TCM, il doit prévoir une mise en mémoire tampon suffisamment longue de la ou des touches, afin qu'aucune pression de touche ne soit perdue même pour un actionnement rapide.

Interface AP

Dans l'interface AP, une zone est prévue pour la sélection d'une touche matérielle. Cette zone se trouve dans le DB19.DBB10. L'AP peut y spécifier directement une valeur de touche entre 50 et 254.

L'acquittement par le logiciel de commande s'effectue en deux étapes. Cette procédure est nécessaire afin que le logiciel de commande puisse détecter correctement deux occurrences successives du même code de touche comme deux événements distincts. Lors de la première étape, l'information de commande 255 est écrite dans l'octet DB19.DBB10. Chaque séquence de touche de l'AP peut être clairement reconnue par cette pression de touche virtuellement définie. L'information de commande est sans signification pour le programme AP utilisateur et ne doit pas être modifiée. Lors de la deuxième étape, l'acquittement effectif vis-à-vis de l'AP est effectué en supprimant DB19.DBB10. A partir de cet instant, le programme AP utilisateur peut spécifier une nouvelle touche matérielle. En même temps, la demande de la touche matérielle actuelle est traitée dans le logiciel de commande.

Exemple

Fichier de configuration :

```
; Touches matérielles AP (KEY50-KEY254)
[keyconfiguration]
KEY50.0 = nom := AreaMachine, boîte de dialogue := SlMachine
KEY51.0 = nom := AreaParameter, boîte de dialogue := SlParameter
KEY52.0 = nom := AreaProgramEdit, boîte de dialogue := SlProgramEdit
KEY53.0 = nom := AreaProgramManager, boîte de dialogue := SlPmDialog
KEY54.0 = nom := AreaDiagnosis, boîte de dialogue := SlDgDialog
KEY55.0 = nom := AreaStartup, boîte de dialogue := SlSuDialog
KEY56.0 = nom := Custom, boîte de dialogue := SlEsCustomDialog
```

Les descripteurs de zone et de boîte de dialogue figurent dans le fichier "systemconfiguration.ini" sous [Répertoire système siemens]/cfg.

11.3 Sélection d'une boîte de dialogue par la CN

Instruction MMC dans HMI Operate

Vous pouvez utiliser les instructions MMC comme décrit ci-après :

1. **Définition d'instructions MMC**
   Le fichier par défaut "systemconfiguration.ini" contient les combinaisons suivantes :
   address:=MCYCLES --> command:=LM
   address:=CYCLES --> command:=PICTURE_ON
   Cette distinction est nécessaire afin de pouvoir distinguer les cycles de mesure des autres cycles. En d'autres termes :
   - LM s'applique toujours pour les cycles de mesure, PICTURE_ON toujours pour les autres cycles
   - Les nouvelles instructions MMC définies ne doivent pas être désignées par PICTURE_ON ou LM
2. **Licence "Run MyScreens"**
   L'ensemble des boîtes de dialogue ouvertes par "Run MyScreens", à l'exception des cycles de mesure, est soumis à la licence "Run MyScreens" et, par conséquent, sans licence seul un nombre limité de boîtes de dialogue peut être utilisé.
   Exemple d'appel avec test.com (Run MyScreens) :

```
g0 f50
MMC("CYCLES,PICTURE_ON,test.com,masque1","A")
m0
MMC("CYCLES,PICTURE_OFF","N")
M30
```

Exemples de masques de cycle 12

12.1 Exemples de masques de cycle

Avec l'installation de SINUMERIK Operate, des exemples sont enregistrés pour les masques de cycle créés avec Run MyScreens.

Les exemples Run MyScreens sont disponibles au chemin d'archivage suivant :

[Répertoire système siemens]		
...\siemens\sinumerik\hmi\template\easyscreen\		
...		
	standard\	Vues d'aide, PNG
	x3d\	Vues d'aide, animations
...		
	Milling\	Opérations de fraisage
	Turning\	Opérations de tournage
...		
	deu\	description en allemand
	eng\	description en anglais

Des exemples dans le groupe fonctionnel Mise en service sont disponibles sur l'interface HMI :

Touche logicielle horizontale Données système → Données HMI → Modèles → Exemples → Easyscreen → ... (voir ci-dessus).

Brève description des exemples :

- Opérations de fraisage
  - Exemple de perçage : il est possible d'ajouter ici une position
  - Exemple de mesure de pièce : lorsque la boîte de dialogue est ouverte, il est possible de générer et d'exécuter ici un appel de cycle avec un démarrage de NC.
  - Exemple de compteurs de pièces cet exemple présente le travail avec les GUD
- Opérations de tournage
  - Dispositifs : ils incluent une poupée mobile, un embarreur, un récepteur de pièces
  - Exemple de mesure de pièce : lorsque la boîte de dialogue est ouverte, il est possible de générer et d'exécuter ici un appel de cycle avec un démarrage de NC.
  - Exemple de perçage : il est possible d'ajouter ici une position

Configuration dans le Sidescreen 13

Pour l'affichage des configurations Run MyScreens dans le Sidescreen, il existe les possibilités suivantes :

1. Affichage d'une configuration Run MyScreens dans une page Sidescreen (séparée) :
   La configuration Run MyScreens occupe alors toute la place sur la page Sidescreen.
2. Affichage de plusieurs configurations Run MyScreens dans une page Sidescreen (séparée) :
   Les différentes configurations Run MyScreens sont affichées les unes au-dessus des autres dans la page Sidescreen, dans les éléments appelés Sidescreen. Ces derniers, et par conséquent aussi les configurations Run MyScreens, peuvent défiler verticalement.
3. Ajout d'une ou de plusieurs configurations Run MyScreens à une page Sidescreen existante :
   En d'autres termes, la configuration Run MyScreens est ajoutée à une page Sidescreen contenue dans la fourniture SIEMENS. Ce cas correspond au point 2, sauf la page Sidescreen concernée.
4. Ajout d'une ou de plusieurs configurations Run MyScreens à un élément Sidescreen :
   Les différentes configurations Run MyScreens sont affichées les unes à côté des autres dans des widgets Sidescreen et peuvent défiler horizontalement.

13.1 Affichage d'une configuration Run MyScreens dans une page Sidescreen

Pour l'affichage d'une configuration Run MyScreens dans une page Sidescreen (séparée), il faut ajouter une nouvelle page Sidescreen à la configuration Sidescreen existante ("slsidescreen.ini"). La configuration Run MyScreens est ensuite affichée ou exécutée dans cette page Sidescreen.

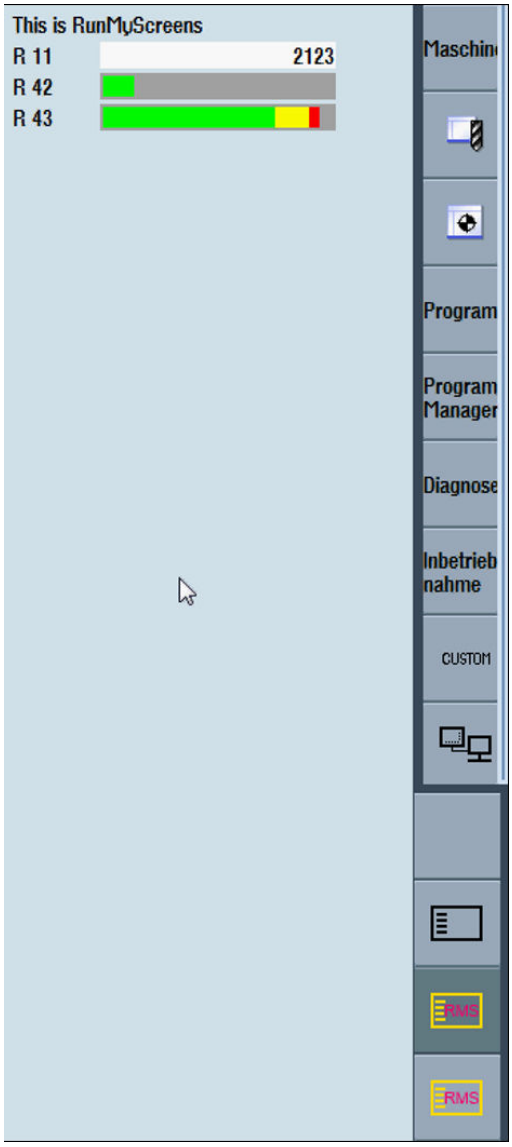


Figure 13-1 **Affichage d'une configuration Run MyScreens dans une page Sidescreen**

Marche à suivre

Ajouter l'entrée suivante dans "slsidescreen.ini" :

```
[Sidescreen]
PAGE100= name:=RMSPage,
implementation:=sleseasyscreen.SlEsSideScreenPage
```

- Le numéro de la page, ici "100", doit être incrémenté en fonction des pages Sidescreen existantes. Le numéro de la page doit être ≥ 100. Cela permet d'éviter des conflits avec les pages SIEMENS.
- Le nom de la page, ici "RMSPage", peut être choisi librement.
- L'indication pour l'implémentation de la page, ici "sleseasyscreen.SlEsSideScreenPage", ne doit en aucun cas être modifiée.

À l'étape suivante, paramétrer la configuration Run MyScreens à exécuter. Pour cela, créer une nouvelle section dont le nom contient le nom de la page tout juste créée :

```
[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"
```

- Le nom de la section est créé en ajoutant le nom de la page au préfixe "Page_", ici "RMSPage".
- Sous Icône, indiquer le nom du fichier, ici "rmspage.png". Le fichier contient le symbole pour le bouton de sélection de la page. Enregistrer le fichier dans le répertoire /user/sinumerik/hmi/ico/ico640 ou /oem/sinumerik/hmi/ico/ico640.
- Dans le Property maskPath, indiquer le nom du fichier qui contient la configuration RunMyScreens, ici "sidescreenmask.com".
- Dans le Property maskName, indiquer le nom du masque RunMyScreens qui doit être affiché, ici "Mask".
- Le focus de saisie est défini dans Property focusable. Ce n'est que lorsque cet attribut prend la valeur "true" que la page peut obtenir le focus de saisie. Cet attribut est requis pour les pages contenant des éléments de saisie.

Enregistrer le fichier "slsidescreen.ini" dans le répertoire /oem/sinumerik/hmi/cfg ou /user/sinumerik/hmi/cfg.

La configuration Run MyScreens est disponible au prochain démarrage IHM.

Exemple

Exemple de configuration complète dans le fichier "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 Affichage de plusieurs configurations Run MyScreens dans une page Sidescreen

Pour l'affichage de plusieurs configurations Run MyScreens dans une page Sidescreen (séparée), il faut configurer des éléments appelés Sidescreen, en plus de la page Sidescreen. Les éléments Sidescreen servent à l'affichage des configurations Run MyScreens.

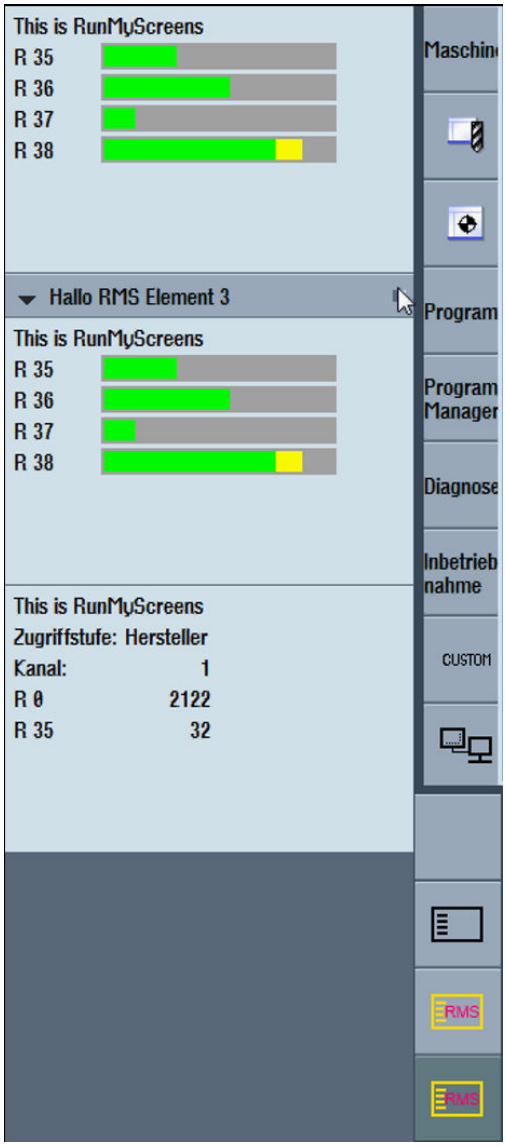


Figure 13-2 **Affichage de plusieurs configurations Run MyScreens dans une page Sidescreen**

Marche à suivre

Ajouter les entrées suivantes dans le fichier "slsidescreen.ini" :

```
[Sidescreen]
PAGE100= name:=RMSPage, implementation:=SlSideScreenPage
```

- Le numéro de la page, ici "100", doit être incrémenté en fonction des pages Sidescreen existantes. Le numéro de la page doit être ≥ 100. Cela permet d'éviter des conflits avec les pages SIEMENS.
- Le nom de la page, ici "RMSPage", peut être choisi librement.
- L'indication pour l'implémentation de la page, ici "SlSideScreenPage", ne doit en aucun cas être modifiée.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement001,
implementation:=sleseasyscreen.SlEsSideScreenElement
ELEMENT002= name:=RMSElement002,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Deux éléments Sidescreen sont attribués à la page – RMSElement001 et RMSElement002. Les noms des éléments peuvent être choisis librement.
- L'indication pour l'implémentation des éléments, ici "sleseasyscreen.SlEsSideScreenElement", ne doit en aucun cas être modifiée.

Pour finir, il faut attribuer encore aux éléments Sidescreen les configurations Run MyScreens s'affichant dans ces éléments :

```
[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"
```

Enregistrer le fichier "slsidescreen.ini" dans le répertoire /oem/sinumerik/hmi/cfg ou /user/sinumerik/hmi/cfg.

La configuration Run MyScreens est disponible au prochain démarrage IHM.

13.3 Ajout de configurations Run MyScreens à une page Sidescreen

La marche à suivre pour l'intégration d'une ou de plusieurs configurations Run MyScreens dans une page Sidescreen existante est identique à la procédure décrite au chapitre Affichage de plusieurs configurations Run MyScreens dans une page Sidescreen (Page 296). La seule différence est que les éléments Sidescreen pour l'affichage des configurations Run MyScreens sont ajoutés à une page Sidescreen déjà existante.

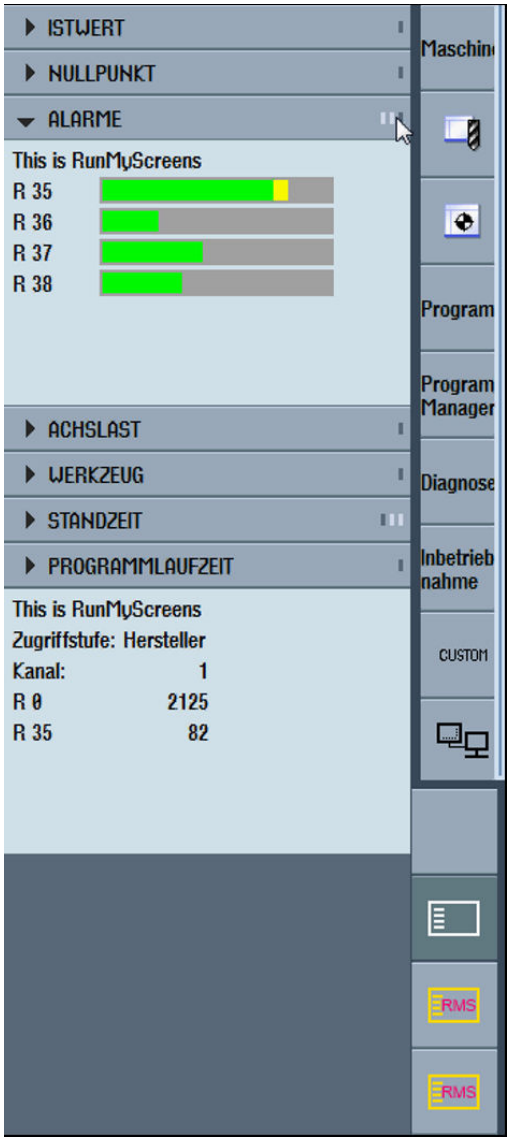


Figure 13-3 **Ajout de configurations Run MyScreens à une page Sidescreen**

Remarque

À partir des pages Sidescreen contenues dans la fourniture SIEMENS, les configurations Run MyScreens peuvent uniquement être ajoutées à la WidgetsPage (voir le fichier "slsidescreen.ini").

Marche à suivre

Ajouter les entrées suivantes dans le fichier "slsidescreen.ini" :

Dans la section [Page_WidgetsPage], ajouter un élément correspondant pour l'enregistrement de la configuration Run MyScreen.

```
[Page_WidgetsPage]
ELEMENT100= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Pour les nouveaux éléments, utiliser une plage de numéros ≥ 100. Cela permet d'éviter des conflits avec les éléments d'affichage SIEMENS. L'élément RMSElement est inséré en dessous les éléments SIEMENS dans la WidgetsPage, en fonction de cette numérotation.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask001"
```

13.4 Ajout de configurations Run MyScreens à un élément Sidescreen

La base de la marche à suivre décrite ci-après à titre d'exemple est une page Sidescreen avec un élément Sidescreen. Deux configurations Run MyScreens sont affichées l'une à côté de l'autre dans l'élément Sidescreen.

Configurer une page Sidescreen avec un élément Sidescreen comme décrit ci-après. Attribuer deux widgets Sidescreen à l'élément Sidescreen avec la configuration Run MyScreens correspondante.

Marche à suivre

Ajouter les entrées suivantes dans le fichier "slsidescreen.ini" :

```
[Sidescreen]
PAGE100= name:=RMSPage, implementation:=SlSideScreenPage
```

- L'indication de l'implémentation, ici "SlSideScreenPage", ne doit en aucun cas être modifiée.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement, implementation:= SlSideScreenElement
```

- L'indication de l'implémentation, ici "SlSideScreenElement", ne doit en aucun cas être modifiée.

```
[Element_RMSElement]
TextId=RMS_ELEMENT_TITLE
TextFile=rmstexts
TextContext=rmstexts
TextContext=rmstexts
WIDGET001= name:=RMSWidget001,
implementation:=sleseasyscreen.SlEsSideScreenWidget
WIDGET002= name:=RMSWidget002,
implementation:=sleseasyscreen.SlEsSideScreenWidget
```

- En cas d'utilisation de l'implémentation SlSideScreenElement pour l'élément Sidescreen (voir la section [Page_RMSPage]), l'élément Sidescreen possède une ligne d'en-tête dans laquelle un texte peut être affiché.
  Ce texte est configuré avec les entrées TextId, TextFile et TextContext :
  - TextId : ID texte du texte à afficher
  - TextFile : fichier dans lequel le texte est enregistré
  - TextContext : contexte dans lequel le texte se trouve dans ce fichier

  Un fichier séparé pour chaque langue doit être mis à disposition.
  Le nom de ces fichiers est créé selon le schéma suivant :
  <TextFile>_<suffixe langue>.ts, p. ex. rmstexts_deu.ts pour l'exemple ci-dessus.
  Si aucun fichier ni contexte n'est indiqué (TextFile=<empty> et TextContext=<empty>), l'ID texte s'affiche en tant que texte. Comme ID texte, une chaîne quelconque peut être indiquée.
  Le fichier texte (p. ex. rmstexts_deu.ts) est structuré comme suit :

```
<?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>
```

  Enregistrer le fichier dans le répertoire /oem/sinumerik/hmi/lng ou /user/sinumerik/hmi/lng . Au démarrage IHM suivant, le fichier est converti au format de fichier requis pour l'exécution.
- Les noms des widgets attribués à l'élément, ici "RMSWidget001" et "RMSWidget002", peuvent être choisis librement.
- L'implémentation de widget "sleseasysscreen.SlEsSideScreenWidget" ne doit en aucun cas être modifiée.

```
[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"
```

- Les noms des sections pour la configuration des widgets Sidescreen sont créés en ajoutant au préfixe "Widget_" le nom du widget devant être configuré dans cette section.
- Dans le Property maskPath, indiquer le nom du fichier qui contient la configuration RunMyScreens, ici "mymask001.com" ou "mymask002.com".
- Dans le Property maskName, indiquer le nom du masque RunMyScreens qui doit être affiché, ici "MyMask001" ou "MyMask002".

13.5 Remarques concernant l'exécution des configurations Run MyScreens dans le Sidescreen

Tenir compte des remarques suivantes :

- Les fonctions LS, LM, DLGL, EXIT, EXITLS ne sont pas disponibles lors de l'exécution des configurations Run MyScreens dans le Sidescreen.
- Toute requête de la taille du masque dans le bloc LOAD est impossible.
- Les textes sont toujours affichés avec la police qui est utilisée dans SINUMERIK Operate pour une résolution d'écran de 640 x 480 pixels.
- Les positions configurées des organes de commande sont converties sans modification (par ex. étirement).
- La taille des boîtes de dialogue est adaptée automatiquement.
- Si des configurations Run MyScreens sont exécutées dans une page Sidescreen, toute la surface de la page est disponible pour la configuration de l'affichage. Lors de l'exécution dans les éléments ou widgets Sidescreen, la surface disponible pour l'affichage est prédéfinie par le système en fonction du contexte.
- L'en-tête et les touches logicielles ne sont pas configurables
- Toutes les indications d'erreur ou de débogage sont écrites de manière séquentielle dans le fichier texte "easyscreen_log.txt". Aucun identificateur spécifique dont la configuration permet d'émettre un message n'est disponible.
- L'état d'une configuration Run MyScreens affichée dans le Sidescreen est rejeté lorsque la configuration est masquée.

Remarque

L'intégration des configurations Run MyScreens dans le Sidescreen entraîne l'exécution simultanée de plus d'une configuration Run MyScreens. Pour maintenir le bon fonctionnement du système global, respecter la charge supplémentaire générée du système, selon le matériel utilisé.

Configuration dans le Display Manager 14

14.1 Affichage des configurations Run MyScreens dans le Display Manager

Pour l'affichage des configurations Run MyScreens dans le Display Manager, la boîte de dialogue SlSideScreenDialog est disponible. Cette dernière peut afficher une ou plusieurs pages Sidescreen (voir IM9 chapitre 3.13), en fonction de la place disponible. Plusieurs pages sont affichées les unes à côtés des autres sous forme de colonnes. La place disponible (largeur) est répartie uniformément sur les différentes colonnes. Le nombre de pages à afficher et leur contenu sont configurés. Chaque page a son propre fichier de configuration. Les noms de ces fichiers de configuration sont indiqués lors de la configuration de la boîte de dialogue dans le fichier "systemconfiguration.ini", dans le paramètre de ligne de commande "cmdline". Derrière l'identificateur de paramètres "–sidescreen1" se trouve le nom du fichier de configuration pour la première colonne ; derrière l'identificateur de paramètres "–sidescreen2" se trouve le nom du fichier de configuration pour la deuxième colonne, etc.

Exemple

```
[dialogs]
DLG500= name:=RMSApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false, cmdline:="-sidescreen1
rmspage1.ini –sidescreen2 rmspage2.ini –spacing 3”
```

L'instance de la boîte de dialogue SlSideScreenDialog créée dans cet exemple a le nom RMSApp. Elle contient 2 pages/colonnes. Les deux pages/colonnes sont configurées dans les fichiers "rmspage1.ini" et "rmspage2.ini". Un écart de 3 pixels est présent entre les deux colonnes.

Remarque

Les noms des fichiers avec les configurations de la page, ici "rmspage1.ini" et "rmspage2.ini", peuvent être choisis librement.

L'utilisation des configurations Run MyScreens dans le Display Manager a lieu comme décrit au chapitre Configuration dans le Sidescreen (Page 293). La seule différence est que l'intégration des configurations n'a pas lieu dans le fichier "slsidescreen.ini", mais dans les fichiers prévus respectivement pour la configuration des pages existantes.

Pour l'intégration des configurations Run MyScreens dans le Display Manager de SINUMERIK Operate, deux variantes sont disponibles. Celles-ci sont décrites dans les chapitres suivants.

14.2 Intégration dans la SISideScreenDialog

Pour l'intégration de la configuration Run MyScreens, il faut créer l'instance de la boîte de dialogue SlSideScreenDialog dans le fichier "systemconfiguration.ini" et le fichier pour l'intégration de la configuration Run MyScreens.

L'exemple suivant montre l'intégration d'une configuration Run MyScreens. La configuration Run MyScreens occupe la fenêtre complète de la boîte de dialogue SlSideScreenDialog, c.-à-d. que seule une page/colonne est disponible.

Marche à suivre

Créer d'abord une instance de la boîte de dialogue SlSideScreenDialog dans le fichier "systemconfiguration.ini" de la section `[dialogs]`. L'instance reçoit le nom "RMSApp".

```
[dialogs]
DLG500= name:=RMSApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false,
cmdline:="-sidescreen1 rmsapp.ini"
```

L'étape suivante consiste à paramétrer ou intégrer la configuration RunMyScreens dans le fichier "rmsapp.ini" :

```
[Sidescreen]
PAGE001= name:=RMSPage, implementation:=SlSideScreenPage
```

- Créer une page avec le nom "RMSPage".

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Sur la page RMSPage, créer un élément avec le nom RMSElement. Cet élément sert à l'affichage de la configuration Run MyScreens.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

- Dans le Property maskPath, indiquer le nom du fichier qui contient la configuration RunMyScreens, ici "sidescreenmask.com".
- Dans le Property maskName, indiquer le nom du masque RunMyScreens qui doit être affiché, ici "Mask".

Enregistrer les fichiers "rmsapp.ini" et "systemconfiguration.ini" (voir ci-dessus) dans le répertoire /oem/sinumerik/hmi/cfg ou /user/sinumerik/hmi/cfg.

La configuration Run MyScreens est disponible au prochain démarrage IHM.

14.3 Intégration dans SIWidgetsApp

L'intégration d'une configuration Run MyScreens dans l'application SlWidgetsApp incluse dans la fourniture SIEMENS est décrite ci-après.

En fonction de la page/colonne dans laquelle l'application SlWidgetsApp doit intégrer la configuration Run MyScreens, il faut soit étendre le fichier "sldmwidgets1.ini", soit le fichier "sldmwidgets2.ini".

L'exemple suivant montre l'intégration d'une configuration Run MyScreens dans la page/colonne gauche de l'application SlWidgetsApp.

Marche à suivre

Étendre le fichier "sldmwidgets1.ini" comme suit :

```
[Sidescreen]
PAGE001= name:=WidgetsPage, implementation:=SlSideScreenPage
[Page_WidgetsPage]
ELEMENT100= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Créer un élément avec le nom "RMSElement" pour l'affichage de la configuration Run MyScreens.

Remarque

Pour les nouveaux éléments, utiliser une plage de numéros ≥ 100. Cela permet d'éviter des conflits avec les éléments d'affichage SIEMENS. L'élément RMSElement est inséré en dessous les éléments SIEMENS dans la WidgetsPage, en fonction de cette numérotation.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

- Dans le Property maskPath, indiquer le nom du fichier qui contient la configuration RunMyScreens, ici "sidescreenmask.com".
- Dans le Property maskName, indiquer le nom du masque Run MyScreens qui doit être affiché, ici "Mask".

Enregistrer le fichier "sldmwidgets1.ini" dans le répertoire /oem/sinumerik/hmi/cfg ou /user/sinumerik/hmi/cfg.

La configuration Run MyScreens est disponible au prochain démarrage IHM.

14.4 Remarques concernant l'exécution des configurations Run MyScreens dans le Display Manager

Tenir compte des remarques suivantes :

- Les fonctions LS, LM, DLGL, EXIT, EXITLS ne sont pas disponibles lors de l'exécution des configurations Run MyScreens dans le Display Manager.
- Toute requête de la taille du masque dans le bloc LOAD est impossible.
- Les textes sont toujours affichés avec la police qui est utilisée dans SINUMERIK Operate pour une résolution d'écran de 640 x 480 pixels.
- Les positions configurées des organes de commande sont converties sans modification (p. ex. étirement).

- Si des configurations Run MyScreens sont exécutées dans une page Sidescreen, toute la surface de la page est disponible pour la configuration de l'affichage. Lors de l'exécution dans les éléments ou widgets Sidescreen, la surface disponible pour l'affichage est prédéfinie par le système en fonction du contexte.
- Toutes les indications d'erreur ou de débogage sont écrites de manière séquentielle dans le fichier texte easyscreen_log.txt. Aucun identificateur spécifique dont la configuration permet d'émettre un message n'est disponible.
- L'état d'une configuration Run MyScreens affichée dans le Display Manager est rejeté lorsque la configuration est masquée.

Remarque

L'intégration des configurations Run MyScreens dans le Display Manager entraîne l'exécution simultanée de plus d'une configuration Run MyScreens. Pour maintenir le bon fonctionnement du système global, respecter la charge supplémentaire générée du système, selon le matériel utilisé.

Listes de référence

A

A.1 Listes des touches logicielles d'accès

A.1.1 Liste des touches logicielles d'accès pour le tournage

Groupe fonctionnel Programme - Tournage

TLH1	TLH2	TLH3	TLH4	TLH5	TLH6	TLH7	TLH8
Edit	**Perçage**	**Tournage**	**Tournage contour**	**Fraisage**	**Divers**	**Simulation**	**Sélection CN**
TLH9	TLH10	TLH11	TLH12	TLH13	TLH14	TLH15	TLH16
--	Droite Cercle (uniquement pour Shop-Turn)	--	--	Mesurer pièce	Mesurer outil	**OEM**	--

Tournage

Dans les tableaux suivants, les touches logicielles d'accès possibles de la technologique de tournage sont représentées. Des attributions de touches logicielles d'accès individuels peuvent diverger en fonction du système. Les touches logicielles OEM spécifiées sont autorisées pour "Run MyScreens".

programGUIDE (code G) touches logicielles d'accès :

	Perçage	**Tournage**	**Tournage contour**		**Fraisage**		**Divers**	
	TLH2	**TLH3**	**TLH4**		**TLH5**		**TLH6**	
TLV1	Centrage	Chariotage	Contour	--	Surfaçage	Contour	Réglages	High Speed Settings
TLV2	Perçage et alésage à l'alésoir	Gorge	Chariotage	--	Poche	Contournage	Pivoter plan	Axes parallèles
TLV3	Perçage de trous profonds	Dégagement	Chariotage résiduel	--	Tourillon polygonal	Perçage d'avant-trous	Orientation outil	--
TLV4	Alésage	Filetage	Plongée	--	Rainure	Poche	--	--
TLV5	Filetage	Tronçonnage	Plongée résiduelle	--	Fraisage de filetages	Poche mat. résiduel	--	--
TLV6	**OEM**	--	Plongée G+D	--	Gravure	Tourillon	Sous-programme	--

TLV7	Positions	**OEM**	Plongée résiduelle	**OEM**	**OEM**	Tourillon mat. résiduel	--	**OEM**
TLV8	Répéter position	--	>>	<<	Fraisage de contours	<<	>>	<<

Touches logicielles d'accès ShopTurn :

	Perçage	**Tournage**	**Tournage contour**		**Fraisage**		**Divers**		
	TLH2	**TLH3**	**TLH4**		**TLH5**		**TLH6**		**TLH10**
TLV1	Perçage au centre	Chariotage	Nouveau contour	--	Surfaçage	Nouveau contour	Réglages	High Speed Settings	Outillage
TLV2	Centrage	Gorge	Chariotage	--	Poche	Contournage	Pivoter plan	Axes parallèles	Droite
TLV3	Perçage et alésage à l'alésoir	Dégagement	Chariotage résiduel	--	Tourillon polygonal	Perçage d'avant-trous	Orientation outil	Répéter programme	P. centrale cercle
TLV4	Perçage de trous profonds	Filetage	Plongée	--	Rainure	Poche	Contre-broche	--	Rayon du cercle
TLV5	Filetage	Tronçonnage	Plongée résiduelle	--	Fraisage de filetages	Poche mat. résiduel	Transformations	--	Système polaire
TLV6	**OEM**	--	Plongée G+D	--	Gravure	Tourillon	Sous-programme	--	Arrêt/marche
TLV7	Positions	**OEM**	Plongée résiduelle	**OEM**	**OEM**	Tourillon mat. résiduel	--	**OEM**	--
TLV8	Répéter position	--	>>	<<	Fraisage de contours	<<	>>	<<	--

A.1.2 Liste des touches logicielles d'accès pour le fraisage

Groupe fonctionnel Programme - Fraisage

TLH1	TLH2	TLH3	TLH4	TLH5	TLH6	TLH7	TLH8
Edit	**Perçage**	**Fraisage**	**Fraisage de contours**	**Tournage** (uniquement pour code G)	**Divers**	**Simulation**	**Sélection CN**
TLH9	TLH10	TLH11	TLH12	TLH13	TLH14	TLH15	TLH16
--	Droite Cercle (uniquement pour ShopMill)	--	--	Mesurer pièce	Mesurer outil	**OEM**	--

Fraisage

Dans les tableaux suivants, les touches logicielles d'accès possibles de la technologique de fraisage sont représentées. Des attributions de touches logicielles d'accès individuels peuvent diverger en fonction du système. Les touches logicielles OEM spécifiées sont autorisées pour "Run MyScreens".

programGUIDE (code G) touches logicielles d'accès :

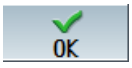
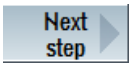
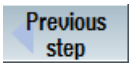
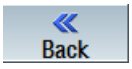
	Perçage	**Fraisage**	**Fraisage de contours**		**Tournage**		**Divers**	
	TLH2	**TLH3**	**TLH4**		**TLH5**		**TLH6**	
TLV1	Centrage	Surfaçage	Contour	--	Chariotage	Contour	Réglages	--
TLV2	Perçage et alésage à l'alésoir	Poche	Contournage	--	Gorge	Chariotage	Pivoter plan	Axes parallèles
TLV3	Perçage de trous profonds	Tourillon polygonal	Perçage d'avant-trous	--	Dégagement	Chariotage résiduel	Orientation outil	--
TLV4	Alésage	Rainure	Poche	--	Filetage	Plongée	High Speed Settings	--
TLV5	Filetage	Fraisage de filetages	Poche mat. résiduel	--	Tronçonnage	Plongée résiduelle	--	--
TLV6	**OEM**	Gravure	Tourillon	--	--	Plongée G +D	Sous-programme	--
TLV7	Positions	**OEM**	Tourillon mat. résiduel	**OEM**	**OEM**	Plongée résiduelle	--	**OEM**
TLV8	Répéter position	--	>>	<<	Tournage contour	<<	>>	<<

Touches logicielles d'accès ShopMill :

	Perçage	**Fraisage**	**Fraisage de contours**		**Divers**		**Droite, cercle**
	TLH2	**TLH3**	**TLH4**		**TLH6**		**TLH10**
TLV1	Centrage	Surfaçage	Nouveau contour	--	Réglages	--	Outillage
TLV2	Perçage et alésage à l'alésoir	Poche	Contournage	--	Pivoter plan	Axes parallèles	Droite
TLV3	Perçage de trous profonds	Tourillon polygonal	Perçage d'avant-trous	--	Orientation outil	Répéter programme	P. centrale cercle
TLV4	Alésage	Rainure	Poche	--	High Speed Settings	--	Rayon du cercle
TLV5	Filetage	Fraisage de filetages	Poche mat. résiduel	--	Transformations	--	Hélice
TLV6	**OEM**	Gravure	Tourillon	--	Sous-programme	--	Système polaire
TLV7	Positions	**OEM**	Tourillon mat. résiduel	**OEM**	--	**OEM**	--
TLV8	Répéter position	--	>>	<<	>>	<<	--

A.2 Liste des touches logicielles prédéfinies

Tableau A-1 Touches logicielles prédéfinies

Nom	Touche logicielle
SOFTKEY_OK	
SOFTKEY_CANCEL	
SOFTKEY_APPLY	
SOFTKEY_MORE	
SOFTKEY_BACK	
SOFTKEY_ASSISTANT_NEXT	
SOFTKEY_ASSISTANT_PREVIOUS	
SOFTKEY_NAV_BACK	

A.3 Liste des niveaux d'accès

Les différents niveaux d'accès ont la signification suivante :

Niveau de protection	Verrouillage par	Groupe
ac0	Réservé à Siemens	
ac1	Mot de passe	Constructeur de machines
ac2	Mot de passe	Maintenance
ac3	Mot de passe	Utilisateur
ac4	Commutateur à clé, position 3	Programmeur, régleur
ac5	Commutateur à clé, position 2	Opérateur qualifié
ac6	Commutateur à clé, position 1	Opérateur formé
ac7	Commutateur à clé, position 0	Opérateur spécialisé

A.4 Liste des couleurs

Couleurs système

Un tableau de couleurs standard est disponible pour la configuration de la boîte de dialogue (sous-ensemble des couleurs standard correspondantes). Pour chaque élément (texte, champ de saisie, arrière-plan, etc.), on peut sélectionner l'une des couleurs suivantes entre 0 et 133.

Vous pouvez également indiquer des couleurs avec les valeurs RVB ("#RRGGBB") à la place des couleurs prédéfinies.

Index	Pictogramme	Couleur	Description de couleur
1		noir	
2		orange	
3		vert foncé	
4		gris clair	
5		gris foncé	
6		bleu	
7		rouge	
8		brun	
9		jaune	
10		blanc	
126		noir	Couleur du texte d'un champ de saisie et de visualisation sur lequel se trouve actuellement le focus
127		orange clair	Couleur d'arrière-plan d'un champ de saisie et de visualisation sur lequel se trouve actuellement le focus
128		orange	Couleur système de mise en relief
129		gris clair	Couleur d'arrière-plan
130		bleu	Couleur en-tête (actif)
131		noir	Couleur texte d'en-tête (actif)

Index	Pictogramme	Couleur	Description de couleur
132		turquoise	Couleur d'arrière-plan d'un champ bascule
133		bleu clair	Couleur d'arrière-plan d'un champ de liste

A.5 Liste des identifiants de langue dans les noms de fichier

Langues prises en charge

Langues standard :

Langue	Abréviation dans le nom de fichier
Chinois simplifié	chs
Allemand	deu
Anglais	eng
Français	fra
Italien	ita
Espagnol	esp

Autres langues :

Langue	Abréviation dans le nom de fichier
Chinois traditionnel	cht
Danois	dan
Finnois	fin
Indonésien	ind
Japonais	jpn
Coréen	kor
Malais	msl
Néerlandais	nld
Polonais	plk
Portugais (Brésil)	ptb
Roumain	rom
Russe	rus
Suédois	sve
Slovaque	sky
Slovène	slv
Thaï	tha
Tchèque	csy
Turc	trk

Langue	Abréviation dans le nom de fichier
Hongrois	hun
Vietnamien	vit

A.6 Liste des variables système accessibles

Bibliographie

Tables de paramètres Variables système, /PGAsl/

A.7 Comportement à l'ouverture de la boîte de dialogue (attribut CB)

Le tableau suivant présente une vue d'ensemble des conditions sous lesquelles la méthode CHANGE est appelée.

En ce qui concerne l'attribut CB :

CB0

La méthode CHANGE est déclenchée lors de l'affichage du masque si la variable a une valeur valide à ce moment-là (p. ex. par un préréglage ou une variable CN/AP).

CB1 (par défaut)

La méthode CHANGE n'est pas déclenchée de manière explicite lors de l'affichage du masque. Si la variable contient une variable CN/AP configurée, la méthode CHANGE est évidemment appelée malgré tout.

Condition				Réaction
Type	Variable système ou utilisateur	Valeur par défaut	Exécution de la méthode CHANGE	
Champ d'E/S	Oui	Oui	Oui	Avec la variable CN/AP configurée on obtient toujours au moins un appel automatique de la méthode CHANGE avec la valeur actuelle de la variable CN/AP. Le préréglage de CB n'a aucun effet.
			Non	
		Non	Oui	
			Non	
	Non	Oui	Oui	La méthode CHANGE est appelée avec la valeur par défaut.
			Non	La méthode CHANGE n'est pas appelée.
		Non	Oui	Pas d'appel car il n'y a pas de valeur valide pour appeler une méthode CHANGE.
			Non	

Condition				**Réaction**
Type	**Variable système ou utilisateur**	**Valeur par défaut**	**Exécution de la méthode CHANGE**	
Bascule	Oui	Oui	Oui	Avec la variable CN/AP configurée on obtient toujours au moins un appel automatique de la méthode CHANGE avec la valeur actuelle de la variable CN/AP. Le préréglage de CB n'a aucun effet.
			Non	
		Non	Oui	
			Non	
	Non	Oui	Oui	La méthode CHANGE est appelée avec la valeur par défaut.
			Non	La méthode CHANGE n'est pas appelée.
		Non (la première valeur de la liste de basculement est affectée par défaut)	Oui	La méthode CHANGE est appelée avec la première valeur de la liste de basculement en tant que valeur par défaut.
			Non	La méthode CHANGE n'est pas appelée.

Conseils et astuces

B

B.1 Astuces d'ordre général

- Selon les possibilités, utiliser la variante OPI pour les variables système (variables $).
  Motif :
  Cela permet d'éviter une résolution de nom lourde en ressources selon les circonstances.
  Exemple :
  Au lieu de "$R[10]" utiliser plutôt "/Channel/Parameter/R[u1,10]".
- Eviter une définition cyclique (de même type), p. ex. de la propriété de masque et de variable HLP. Comparer la valeur précédente à définir avec la valeur actuelle et ne la définir effectivement qu'en cas de divergence.
  Motif :
  Eviter une recherche des images d'aide dépendantes de la résolution qui demanderait beaucoup de ressources.
  Exemple :

```
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
```

Recommandation :

```
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
```

- Remplacer plusieurs fonctions RNP() successives par une fonction MRNP().
  Remplacer plusieurs fonctions RDOP() successives par une fonction MRDOP().
  Motif :
  Réduction de la charge de communication et hausse des performances.
- Ne pas lire les paramètres d'entraînement à un cycle plus rapide que 1 seconde, plutôt plus lentement.
  Motif : Autrement la communication avec les entraînements peut être extrêmement perturbée ou des défaillances peuvent survenir.
- Eviter les opérations de calcul successives avec des variables de boîte de dialogue reliées à des variables système ou utilisateur. Pour cela, utiliser p. ex. le registre (REG[x]) ou des variables auxiliaires (invisibles).
  Motif : Chaque affectation d'une valeur entraîne également une écriture dans la variable système ou utilisateur liée.
- Des codes de même type utilisés dans des blocs différents doivent être regroupés dans un bloc SUB pour des raisons de clarté, de maintenabilité et de performance (lors de l'affichage du masque). Celui-ci peut être appelé à l'endroit correspondant avec la fonction CALL().
- En observant la charge de la CPU dans la ligne de la boîte de dialogue (réglage dans slguiconfig.xml), on peut analyser quelles modifications influent sur les performances et dans quelle mesure.

B.2 Astuces concernant le débogage

- Utiliser la fonction DEBUG() à des fins de diagnostic. Lorsque la fonction DEBUG() est appelée, la chaîne de caractères transmise est écrite dans "easyscreen_log.txt". L'affichage d'informations dans la ligne de la boîte de dialogue par la fonction DLGL() peut également s'avérer utile.
  Une fois le développement des masques terminés, cette fonction doit toutefois être supprimée ou mise en commentaire pour des raisons de performances.
- Une fois la création d'un masque terminée, le fichier journal "easyscreen_log.txt" doit toujours être vide (aucune entrée).

B.3 Astuces concernant la méthode CHANGE

- Toujours maintenir les méthodes CHANGE très brèves et très petites, en particulier celles dont les variables sont liées à une variable système ou utilisateur et qui changent très fréquemment.
  Motif :
  Hausse des performances du masque.
- Si possible, ne configurer aucune fonction RNP() dans des méthodes CHANGE. Au lieu de cela, il vaut mieux créer en parallèle une variable invisible avec la variable système ou utilisateur à lire et utiliser celle-ci à la place.
  Motif :
  Une fonction RNP() serait nécessairement transmise à chaque appel. Dans l'autre cas, l'accès se fait uniquement à la valeur actuelle qui existe déjà de toute manière.

Exemple :

Une résolution de nom est réalisée à chaque modification du déplacement d'axe par RNP() pour lire un paramètre machine spécifique au canal :

```
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
```

A l'aide d'une variable invisible, maintenir le paramètre machine spécifique au canal à jour, copier tout changement de valeur dans une variable temporaire, p. ex. registre.

Cette variable temporaire peut alors être utilisée dans la méthode CHANGE du changement de valeur de la position de l'axe sans devoir chaque fois réaliser une résolution de nom du paramètre machine et l'accès en lecture consécutif :

```
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
```

- La vitesse de rafraîchissement et donc l'exécution de la méthode CHANGE correspondante de variables liées à des variables système ou utilisateur dont la valeur change très fréquemment peuvent être réduites en utilisant la propriété de variable UR, p. ex. la variable couplée aux valeurs de l'axe.
  Motif :
  Ainsi, la méthode CHANGE est exécutée dans une trame spécifiée de manière permanente lors d'un changement de valeur.

B.4 Astuces concernant les boucles DO-LOOP

Etant donné que les boucles peuvent entraver les performances de SINUMERIK Operate selon les circonstances, elles doivent être utilisées avec précaution et dans la mesure du possible il convient de renoncer aux actions chronophages.

Il est également recommandé d'utiliser p. ex. un registre (REG[]) comme variable d'exécution, car les variables d'affichage normales (en particulier celles avec une liaison OPI) peuvent

entraver les performances en raison de l'extrême fréquence des mises à jour et des processus d'écriture.

Noter que le nombre de lignes de script traitées "en une fois" est actuellement limité à 10 000. Si ce nombre est atteint, l'exécution de scripts sera interrompue avec une entrée correspondante dans "easyscreen_log.txt".

Motifs :

- Empêcher les boucles infinies
- "Run MyScreens" doit rester opérante

Eléments animés

C

C.1 Introduction

Introduction

Le manuel décrit l'utilisation du X3D Viewer à l'aide duquel il est possible d'intégrer des scènes graphiques fixes et animées (éléments animés) – désignées ci-après par "images d'aide" – dans l'interface graphique de SINUMERIK Operate à partir de la version V4.7 SP1.

Vous pouvez représenter de manière ciblée les séquences de mouvement et les paramètres dans les images d'aide contextuelles. Vous pouvez ainsi rendre les applications plus conviviales et concevoir une interface utilisateur plus attrayante.

Le passage de l'idée à l'image d'aide finalisée comprend les étapes suivantes :

- Création d'éléments graphiques et de modèles 3D pour les images d'aide qui seront visibles au final dans X3D Viewer (voir chapitre Modélisation (Page 320)).
- Création du fichier de description de scène dans lequel les données de modèle du fichier graphique sont affectées aux scènes et aux animations à afficher de manière ciblée (voir chapitre Structure du fichier de description de scène (Page 327)).
- Conversion des fichiers X3D et XML en fichiers IHM (voir chapitre Conversion en fichier hmi (Page 331)).
- Intégration C++ de X3D Viewer dans l'application personnalisée (voir chapitre Exemple de mise en œuvre (Page 333)).
- Application dans Run MyScreens (voir chapitre Affichage dans Run MyScreens (Page 334)).

Vue d'ensemble des formats de fichier

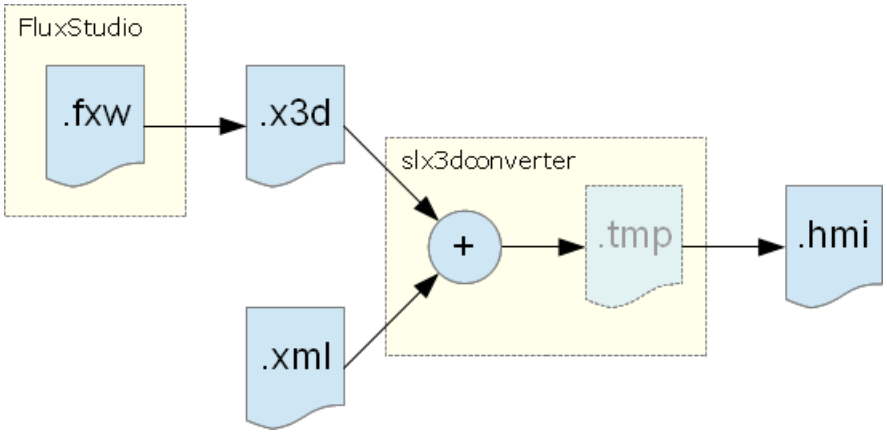


Figure C-1 Formats de fichier

Format de fichier	Description
.fxw	Fichier graphique de Vivaty Studio
.x3d	Fichier de modèle au format d'échange
.xml	Fichier de définition des animations et scènes
.hmi	Fichier de sortie pour X3D Viewer

C.2 Modélisation

C.2.1 Conditions

L'objectif de la modélisation est de créer un fichier avec les modèles 3D générés au format .x3d, norme officielle pour les contenus 3D.

La modélisation s'effectue dans **Vivaty Studio**. Vivaty Studio est un outil de modélisation 3D disponible gratuitement avec de multiples possibilités d'exportation et d'importation.

Conditions

- Vivaty Studio version 1.0 a été installé.
- L'utilisateur est familiarisé avec la modélisation et la manipulation de Vivaty Studio.
- Le format de fichier .x3d n'est pas abordé dans cette description, les connaissances requises sont supposées acquises.

Remarque

Vivaty Studio peut être téléchargé sur Internet sous le lien suivant (https://www.web3d.org/projects/vivaty-studio).

C.2.2 Règles relatives à la modélisation

Veuillez respecter les règles suivantes pour la modélisation. Le respect de ces règles est nécessaire pour la préparation ultérieure du fichier graphique d'aide.

Règles relatives à la modélisation

1. Le TimeSensor doit porter le nom "Animation".
2. Tous les éléments qui vont ensemble doivent être affectés à un groupe.
3. Tous les groupes doivent être placés à la position suivante :
   x = 0, y = 0, z = 0

4. Pour rendre les groupes invisibles, les valeurs de translation doivent être définies comme suit : x = 1000000, y = 1000000 et z = 1000000
5. Les éléments suivants doivent porter le même nom dans les modèles graphiques de Vivaty Studio et dans les définitions de scène XML (une comparaison à ce sujet figure au chapitre Présentation (Page 326)) :
   - Outil
   - Caméras
6. Pour créer un fichier .x3d, vous devez procéder aux réglages suivants dans la boîte de dialogue "Export Options" :

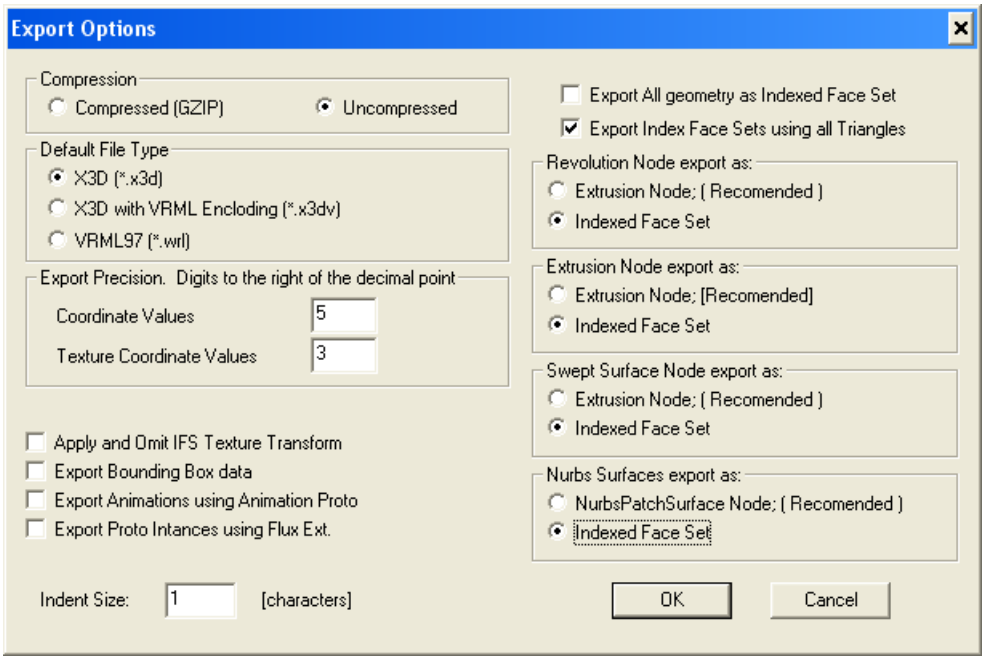


Figure C-2 Réglages dans la boîte de dialogue Export Options

7. Nous recommandons les réglages suivants pour les textes (voir aussi la boîte de dialogue suivante) :
   - Les textes doivent toujours être centrés.
   - La taille du texte doit être de 0,2. Cela permet de bien le positionner. L'affichage de texte par X3D Viewer s'effectue indépendamment de cette valeur dans la taille de police de l'interface utilisateur.

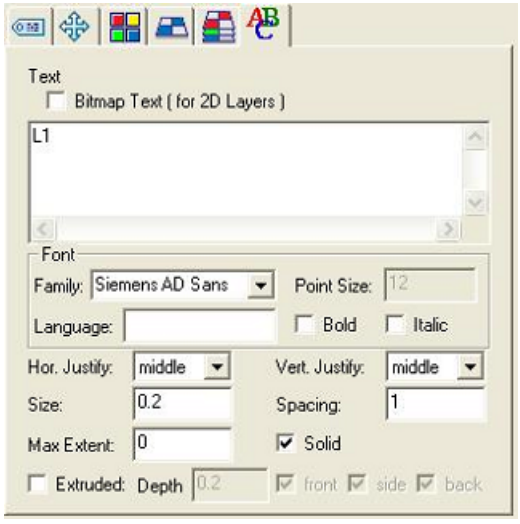


Figure C-3 Réglages relatifs au texte

8. Pour créer une hachure, utilisez le fichier schnitt.png en tant que texture. Les lignes vont toujours de la partie inférieure gauche à la partie supérieure droite en formant un angle de 45°. Le facteur d'échelle doit être réglé sur 3 dans les deux dimensions. La rotation dépend de la forme de construction et doit être adaptée manuellement.

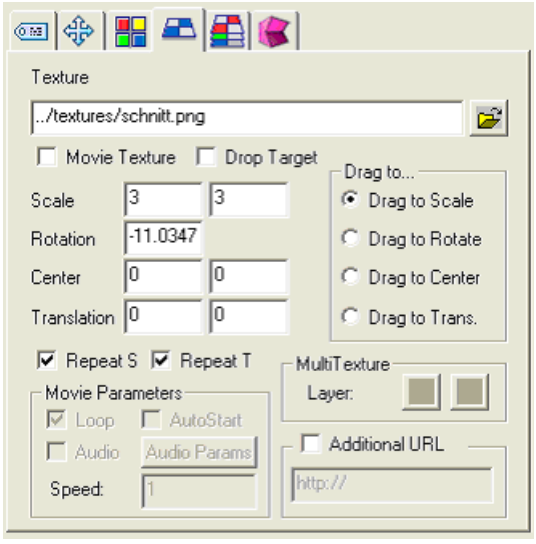


Figure C-4 Réglages relatifs à la hachure

9. Les grandeurs/dimensions suivantes sont recommandées pour les pièces brutes :
   - Cylindre pour tournage :
     longueur = 5,5 ; rayon = 1,4 (cf. modèle turning_blank.fxw)
   - Parallélépipède pour fraisage :
     x = 4,75 ; y = 3 ; z = 3 (cf. modèle milling_blank.fxw)

Voir aussi

Commandes XML (Page 326)

C.2.3 Importation de graphiques (modèles)

Vous pouvez importer des graphiques externes (modèles) dans Flux Studio. Les modèles fréquemment utilisés tels que les outils par exemple, peuvent être enregistrés de manière centralisée sous forme de fichiers .x3d ou .hmi et réutilisés en tant qu'éléments finis dans d'autres images d'aide. Une liste de modèles de modélisation fournis figure au chapitre Modèles pour la modélisation (Page 325).

Des objets complexes peuvent également être créés puis importés à l'aide d'autres logiciels de modélisation, tels que Cinema4D par exemple.

La section suivante décrit comment réutiliser ces modèles.

Importation en tant que Inline

Flux Studio offre l'objet "Inline" pour l'importation de fichiers .x3d. Des éléments externes peuvent par conséquent être intégrés sans avoir à multiplier les données 3D de ces modèles.

L'option de menu **Create** -> **Create Inline** permet d'insérer l'objet. Spécifiez le nom de fichier dans les propriétés de l'objet.

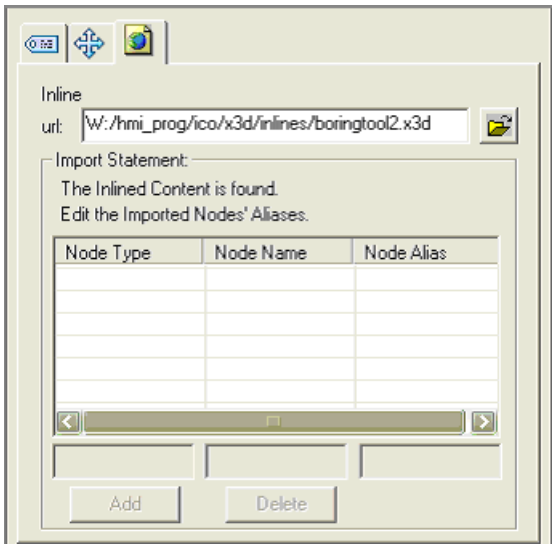


Figure C-5 Importation via l'objet "Inline"

Importation de données de modèle 3D

Pour importer des données de modèle à partir d'un fichier, procédez comme suit.

1. Via l'option de menu **File** -> **Import Other Format**, ouvrez la boîte de dialogue "Flux Studio Accutrans3D Translation Utility".
2. Dans le champ de sélection "File Types", sélectionnez le format correspondant. Par exemple, pour importer un fichier Cinema4D, vous devez sélectionner le type de fichier "3D Studio".
3. Sélectionnez ensuite le fichier souhaité à l'aide du bouton **Select File**, puis lancez l'importation.

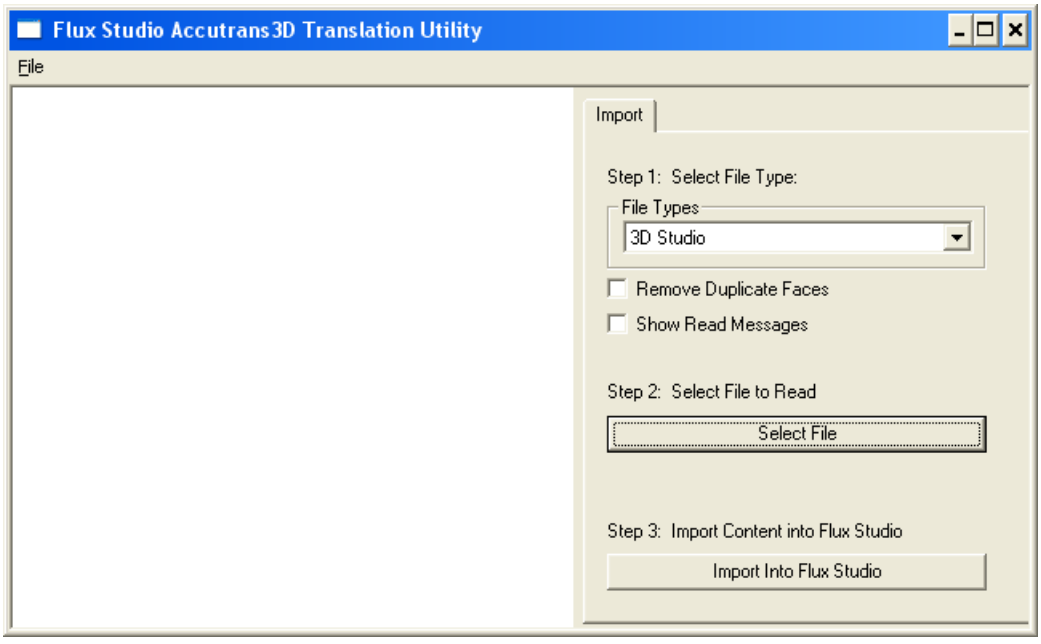


Figure C-6 Importation de données de modèle 3D

Les données de modèle du fichier Cinema4D sont insérées en tant que groupe complet. Toutes les caméras et autres informations importées en même temps doivent être supprimées. Seuls les éléments graphiques du fichier Cinema4D sont requis.

C.2.4 Modèles pour la modélisation

Le présent chapitre contient une liste d'éléments graphiques servant de base pour la conception, le choix des couleurs et le dimensionnement. Pour une apparence uniforme de l'interface utilisateur, nous vous recommandons d'utiliser le style de modèle voire les modèles eux-mêmes pour créer vos propres graphiques.

Général

Modèle	Description
intersection_texture.png	Texture pour une surface de coupe
rapid_traverse_line_hori.fxw	Ligne horizontale pour mettre en évidence le déplacement de l'outil en marche rapide

Modèle	Description
rapid_traverse_line_vert.fxw	Ligne verticale pour mettre en évidence le déplacement de l'outil en marche rapide
feed_traverse_line_hori.fxw	Ligne horizontale pour mettre en évidence la trajectoire de l'outil au cours de l'avance
feed_traverse_line_vert.fxw	Ligne verticale pour mettre en évidence la trajectoire de l'outil au cours de l'avance
dimensioning_lines_hori.fxw	Lignes horizontales de rappel des cotes
dimensioning_lines_vert.fxw	Lignes verticales de rappel des cotes
z1_inc.fxw	Ligne de cote horizontale avec le descripteur Z1
x1_inc.fxw	Ligne de cote verticale avec le descripteur X1
3d_coordinate_origin.fxw	Croix de coordonnées 3D
3d_zero_point.fxw	Point d'origine 3D

Tournage

Modèle	Description
turning_blank.fxw	La pièce brute pour la technologie de tournage : un cylindre simple
turning_centerline.fxw	Ligne médiane
turning_centerpoint.fxw	Croix de coordonnées pour illustrer le point d'origine dans le système de coordonnées pièce (SCP)
turning_refpoint.fxw	Point de référence, p. ex. pour un usinage
turning_machining_area.fxw	Lignes de délimitation de la surface d'usinage

Fraisage

Modèle	Description
milling_blank.fxw	La pièce brute pour la technologie de fraisage : un parallélépipède simple
milling_centerline.fxw	Ligne médiane
milling_refpoint.fxw	Point de référence, p. ex. pour un usinage

C.3 Commandes XML

C.3.1 Présentation

Le fichier de description de scène est requis pour que X3D Viewer puisse accéder aux graphiques (*.xml). Dans le fichier de description de scène, affectez les noms de scène, appelés à partir de la configuration, aux instants qui figurent dans le fichier *.x3d (TimeSensor) auxquels les graphiques doivent être affichés.

C.3.2 Structure du fichier de description de scène

La structure du fichier de description de scène est présenté ci-après avec une explication des commandes XML correspondantes :

Structure et description

<!-- Traitement des textes dépendants du plan pour le tournage -->

```
<TextPlane plane="G18_ZXY" />
```

Description

<TextPlane plane="G18_ZXY" />	Traitement des textes dépendants du plan (noms d'axe) spécifiquement pour les opérations de tournage. Par exemple, si un texte est utilisé avec l'identification "Z" (pour G18, premier axe géométrique), le descripteur d'axe géométrique correspondant de la commande est affiché dans l'image d'aide (p. ex. "Z").

<!--Orientation du texte -->

```
<TextPosition center='true' />
```

Description

<TextPosition center='true' />	Marquage indiquant que les textes sont centrés. Cela est important pour l'affichage des textes dans des images ayant subi une rotation. L'utilisation de ce réglage est recommandée.

<!--Adaptation de la vitesse de l'outil -->

```
<ToolSpeedFactors planeSpeedFactor="0.4" rapidSpeedFactor="0.7"
reducedSpeedFac¬tor="1.0"/>
```

Description

<ToolSpeedFactors	Cette entrée peut être ajoutée si les vitesses d'outil doivent être modifiées.
planeSpeedFactor="0.4"	Facteur pour la vitesse d'avance (0,4 = 40 %)
rapidSpeedFactor="0.7"	Facteur pour la vitesse rapide (0,7 = 70 %)
reducedSpeedFactor="0.1" />	Facteur pour une troisième vitesse (0,1 = 10 %)

<!--Définition d'une 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">
```

Description

<SceneKey name='BoringAnimation'	Nom de l'animation
masterRotationSpeed="-64.0"	Sens et vitesse de rotation de l'outil. Le réglage est facultatif.
maxRotAngle="45.0"	Prévention de l'effet alias (l'outil semble tourner dans le mauvais sens). La valeur est généralement un quart de la symétrie de l'outil. Le réglage est facultatif.
beginTime='0.110	Heure de début de l'animation sur le TimeSensor
endTime='0.118'	Heure de fin de l'animation sur le TimeSensor
view='camiso'	Caméra utilisée pour l'animation
speedMaster='boringtool'	Nom de l'outil pour l'animation
type="VIEW_3D_TURN_CYL">	Le type de vue détermine la vue (voir chapitre Type de vue (Page 330))

<!-- Eléments -->

```
<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'/>
```

Description

<Element name='1'	Indique qu'il s'agit du premier élément de l'animation.
time='0.110'	Instant du premier élément sur le TimeSensor
feed='rapid'	Vitesse de déplacement de l'élément plane = avance d'usinage rapid = marche rapide reduced = vitesse réduite, plus lente que "plane"
dwell='2'/>	Pause dans le mouvement de déplacement de l'animation Un outil rotatif reste en rotation.

<!-- Fin de la définition de l'animation -->

```
</SceneKey>
```

Description

</SceneKey>	Fin de la définition de l'animation

<!-- Animation existante en tant que source d'une nouvelle animation -->

```
<SceneKey source=''BoringAnimation' name=''BoringAnimationRear' mirror='
MirrorScreenX ' />
```

Description

<SceneKey source="BoringAnimation'	Nom d'une animation qui doit être utilisée comme source d'une autre animation.
name="BoringAnimationRear'	Nom de la nouvelle animation basée sur la source
mirror=' MirrorScreenX ' />	Fonction miroir dans le sens de l'axe X, l'affichage s'effectue dans la nouvelle animation.

<!-- Définition d'une scène statique (image), (4 exemples) -->

```
<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'/>
```

Description

<SceneKeyname='Z1abs'	Nom de l'image
time='0.066'	Instant de l'image sur le TimeSensor
view='camside'	Caméra utilisée pour l'image
type="VIEW_SIDE"	Le type de vue détermine la vue (voir chapitre Type de vue (Page 330))
highLightedGroup='dad_Z1abs'/>	Nom du groupe qui doit être mis en évidence. Le groupe a été nommé "Z1abs" dans FluxStudio. Le préfixe "dad_" est ajouté automatiquement par Flux.

<!-- Fin du fichier xml –!>

C.3.3 Symétries miroir et rotations

La commande **mirror** permet de définir la symétrie miroir ou bien la rotation de l'image / du modèle.

Description

mirror="RotateScreenX"	L'image subit une rotation autour de l'axe X
mirror="RotateScreenY"	L'image subit une rotation autour de l'axe Y
mirror="RotateScreenZ"	L'image subit une rotation autour de l'axe Z
mirror="MirrorScreenX"	L'image est inversée (fonction miroir) par rapport à l'axe X
mirror="MirrorScreenY"	L'image est inversée (fonction miroir) par rapport à l'axe Y
mirror="MirrorScreenZ"	L'image est inversée (fonction miroir) par rapport à l'axe Z
mirror="RotatePieceX"	Le modèle subit une rotation autour de l'axe X
mirror="RotatePieceY"	Le modèle subit une rotation autour de l'axe Y
mirror="RotatePieceZ"	Le modèle subit une rotation autour de l'axe Z
mirror="MirrorPieceX"	Le modèle est inversé (fonction miroir) par rapport à l'axe X
mirror="MirrorPieceY"	Le modèle est inversé (fonction miroir) par rapport à l'axe Y
mirror="MirrorPieceZ"	Le modèle est inversé (fonction miroir) par rapport à l'axe Z

Toutes les possibilités de rotation et de symétrie peuvent être combinées. La rotation exige la définition d'un angle de rotation.

Exemples

mirror="RotatePieceZ=180"

mirror="RotatePieceZ=-90 MirrorScreenX"

mirror="RotateScreenY=90"

mirror="MirrorPieceX MirrorPieceY"

mirror="MirrorPieceZ RotatePieceX=-90"

C.3.4 Type de vue

Le type de vue permet de définir les vues. Elles sont basées sur des valeurs empiriques et des règles qui entraînent un affichage judicieux des objets.
La représentation des objets est ainsi garantie en fonction du réglage du système de coordonnées de l'interface utilisateur (PM 52000 $MCS_DISP_COORDINATE_SYSTEM ou bien PM 52001 $MCS_DISP_COORDINATE_SYSTEM_2).

Description

"VIEW_STATIC"	Vue directe sans conversion
"VIEW_3D_TURN_CYL"	Vue 3D pour opérations de tournage (cylindre)
"VIEW_3D_MILL_CUBE"	Vue 3D pour opérations de fraisage (parallélépipède)
"VIEW_3D_DRILL_CUT"	Vue 3D pour opérations de perçage (vue en coupe)
"VIEW_SIDE"	Vue 2D pour opérations de perçage (vue en coupe)
"VIEW_TOP_GEO_AX_1"	Vue 2D depuis la direction du premier axe géométrique
"VIEW_TOP_GEO_AX_2"	Vue 2D depuis la direction du deuxième axe géométrique
"VIEW_TOP_GEO_AX_3"	Vue 2D depuis la direction du troisième axe géométrique

C.4 Conversion en fichier hmi

Remarque

Les fichiers X3D y compris les fichiers XML qui y sont associés sont convertis en fichiers HMI au démarrage de l'IHM.

Pour chaque fichier X3D, un fichier XML correspondant doit être créé avec le même nom. Les fichiers X3D et les fichiers XML doivent être enregistrés dans le répertoire `Chemin de recherche de l'IHM\ico\x3d\turning` ou `milling`. La procédure est analogue à celle des fichiers .ts.

C.5 Affichage dans Create MyHMI /3GL

C.5.1 X3D Viewer

Pour afficher des images d'aide de manière ciblée dans une application OA personnalisée, il convient d'intégrer le widget X3D Viewer dans cette application.

Le widget X3D Viewer propose des interfaces qui permettent la présentation de contenus X3D à l'intérieur de l'IHM.

La classe SlX3dViewerWidget permet de représenter des scènes graphiques. La définition de la classe se situe dans le fichier d'en-tête correspondant slx3dviewerwidget.h qui se trouve dans le répertoire GUI Include global `\hmi_prog\gui\include`.

C.5.2 Classe SlX3dViewerWidget

La classe offre un widget qui peut être utilisé de manière flexible, qui représente de manière autonome les contenus d'un fichier de modèle spécifié lors de l'exécution et qui permet, le cas échéant, d'exécuter l'animation.

L'interface de la classe est constituée d'un constructeur, d'un destructeur et de deux méthodes destinés à commander l'affichage graphique.

Une interface supplémentaire beaucoup plus riche et non décrite ici, directement dérivée de la classe Qt QWidget, est également disponible (p. ex. show(), hide() et resize(...), pour laquelle vous trouverez de plus amples informations dans la documentation Qt).

C.5.3 Méthodes publiques

SlX3dViewerWidget (QWidget* pParent = 0)

Constructeur du widget X3D Viewer.

Paramètres	Signification
pParent	Le paramètre est transmis au constructeur du Qwidget.

~SlX3dViewerWidget ( )

Destructeur du widget X3D Viewer

Paramètres	Signification
-	-

C.5.4 Slots publics

void viewSceneSlot (const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene, int nChannel, int nPlane, SlStepTechnology nTechnology)

La méthode viewSceneSlot de X3D Viewer donne l'instruction de charger la scène statique rsScene et la scène animée rsAnimationScene à partir du fichier rsFileName et de les afficher en alternance.

Concrètement, c'est d'abord la scène statique qui est affichée à plusieurs reprises pendant une durée définie, puis la scène animée.

Si aucune scène statique n'est indiquée, l'animation est aussitôt affichée ; en conséquence, l'indication d'une scène animée peut également être omise.

Le numéro de canal, le plan et la technologie sont utilisés pour faire pivoter les scènes dans la position appropriée (en fonction du système de coordonnées machine réglé).

Paramètres	Signification
rsFileName	Nom du fichier contenant les scènes à afficher.
rsScene	Nom de la scène statique.
rsAnimationScene	Nom de la scène animée.
nChannel	Numéro de canal
nPlane	Plan
nTechnology	Technologie Pour le type d'énumération SlStepTechnology, les constantes suivantes sont définies : • 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)

Forme simplifiée de la méthode viewSceneSlot, avec laquelle X3D Viewer peut recevoir la consigne de charger et d'afficher la scène statique rsScene et/ou la scène animée rsAnimationScene à partir du fichier rsFileName.

Paramètres	Signification
rsFileName	Nom du fichier contenant les scènes à afficher.
rsScene	Nom de la scène statique.
rsAnimationScene	Nom de la scène animée.

C.5.5 Bibliothèques

Pour pouvoir utiliser X3D Viewer dans des projets personnalisés, l'entrée 'slx3dviewer.lib' doit être ajoutée à la liste des dépendances de bibliothèque.

C.5.6 Exemple de mise en œuvre

Un exemple de mise en œuvre figure dans le paquet Create MyHMI/3GL sous **\examples \GUIFrameWork\SlExGuiX3D**.

C.5.7 Paramètres machine

PM d'affichage 9104 $MM_ANIMATION_TIME_DELAY

Temporisation en secondes jusqu'à la mise en œuvre de l'animation dans les images d'aide. Le réglage est sans effet sur les images d'aide exclusivement animées.

Le réglage prend effet de manière globale pour toutes les animations de SINUMERIK Operate.

C.5.8 Consignes de mise en application

- L'animation est interrompue si le widget X3D Viewer est grisé. Il n'est pas nécessaire de sélectionner une scène sans animation dans ce cas.
- En ce qui concerne les performances et l'espace mémoire requis, il convient d'éviter une instanciation fréquente ou répétée du widget X3D Viewer. L'utilisation (mise en œuvre) d'un singleton X3D Viewer est recommandée pour de tels scénarios d'application.
- X3D Viewer permet également l'utilisation dans les concepts de signaux et slots.
- Si une erreur se produit (p. ex. fichier introuvable ou nom de scène inconnu), le widget X3D Viewer affiche une zone de message. Cette zone est de nouveau grisée automatiquement dès qu'un autre fichier d'image d'aide est commandé.

C.6 Affichage dans Run MyScreens

Ce chapitre s'adresse aux développeurs chevronnés de "Run MyScreens". Les connaissances de base requises peuvent être consultées dans la documentation associée.

Outre l'utilisation d'images au format .bmp ou .png, X3D Viewer permet également d'afficher des images d'aide animées.

Pour l'intégration, utilisez comme d'habitude l'interface Run MyScreens pour images d'aide.

Pour afficher des images au format .bmp ou .png, entrer l'indication XG0 dans la définition d'une boîte de dialogue à la section Attribute ou laisser le paramètre vide. Pour intégrer des images d'aide X3D dans une boîte de dialogue, indiquer **XG1** dans les attributs.

```
//M(MASK_F_DR_O1_MAIN/$85407////52,80/XG1)
```

Les paramètres suivants, dont la signification est décrite au chapitre 5.3, Slots publics, sont nécessaires pour commander les différentes images d'aide :

1. Nom de fichier
2. StaticScene (facultatif)
3. AnimationScene (facultatif)
4. Technologie (facultatif)
5. Plan (facultatif)

Les paramètres sont regroupés dans cet ordre et séparés par une virgule dans une chaîne de caractères.

```
Hlp = "Nom de fichier,StaticScene,AnimationScene,Technologie,Plan"
```

Exemples

- L'image d'aide par défaut peut être définie avec la variable Hlp prédéfinie.
  Dans l'exemple suivant, l'animation "MyAnimation" est générée à partir du fichier "MyDlgHelp.hmi". Aucune StaticScene n'est spécifiée, c'est-à-dire qu'aucune scène statique n'est affichée. Aucune indication n'est fournie également en ce qui concerne la technologie et le plan, c'est-à-dire que les valeurs par défaut sont utilisées.
  ```
  Hlp = "MyDlgHelp.hmi,,MyAnimation"
  ```
- Pour les différents paramètres du masque de saisie, il est possible de définir au niveau des variables associées la propriété hlp sur une image d'aide spécifique.
  Dans l'exemple suivant, la scène statique "MyParam" et l'animation "MyAnimation" sont générées dans le plan G17 à partir du fichier "MyDlgHelp.hmi". Aucune indication n'est fournie quant à la technologie, c'est-à-dire qu'une valeur par défaut est utilisée.
  ```
  VarMyParam.hlp = "MyDlgHelp.hmi,MyParam,MyAnimation,,G17"
  ```

Index

G

H

I

L

M

N

O

P

R

S

T

V