SIEMENS

SINUMERIK

SINUMERIK 840D sl
Easy XML

Manuel de programmation

Valable pour :

Logiciel CNC version 4.95

07/2021
6FC5398-5EP40-0DA0

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: 6FC5398-5EP40-0DA0
Ⓟ 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 manuel s'adresse aux :

- programmeurs
- ingénieurs de projet

Utilité

Le manuel de programmation permet au groupe cible de concevoir des interfaces utilisateur spécifiques au client et à l'application avec des éléments de script basés sur XML.

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 840D sl

Une documentation détaillée relative aux fonctions de SINUMERIK 840D sl à partir de la version 4.8 SP4 est disponible sous Vue d'ensemble de la documentation 840D sl (https://support.industry.siemens.com/cs/ww/en/view/109766213).

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. : Fonctions
- Constructeur/S.A.V. : Matériel
- Constructeur/S.A.V. : Configuration/mise en service
- Constructeur/S.A.V. : Safety Integrated
- Constructeur/S.A.V. : SINUMERIK Integrate / MindApp
- Information et formation
- Constructeur/S.A.V. : SINAMICS

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).

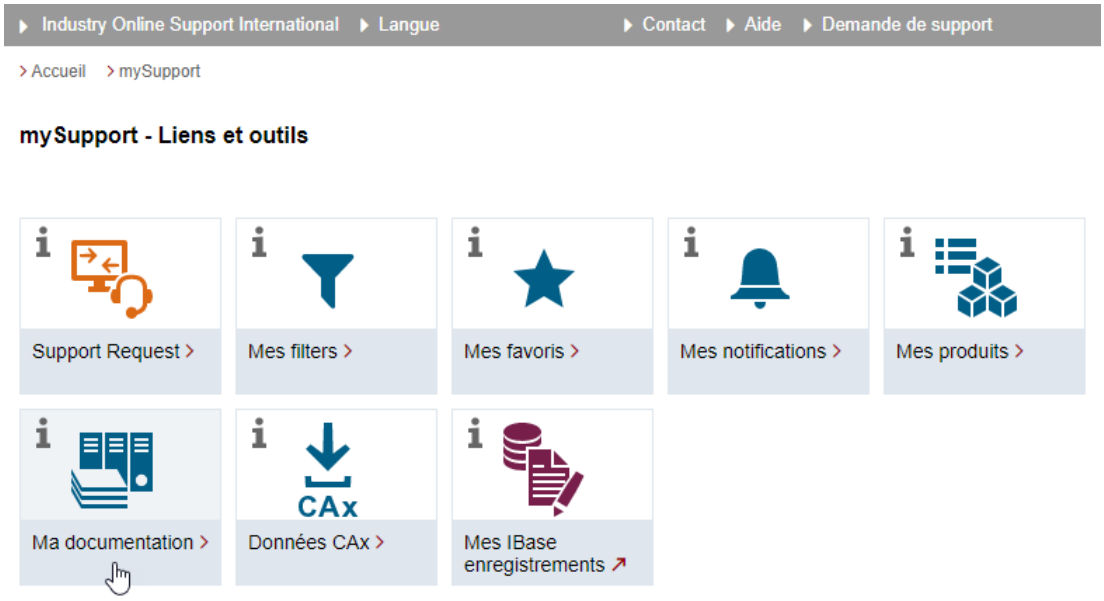
1.4 Remarques concernant la documentation technique

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

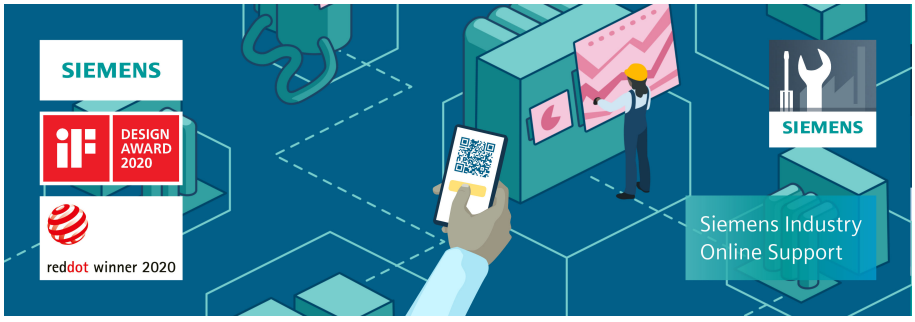
Les numéros de téléphones spécifiques à chaque pays pour l'assistance technique sont disponibles à cette adresse (https://support.industry.siemens.com/cs/ww/fr/sc/4868) dans la zone "Contact".

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

Choc électrique et danger de mort par d'autres sources d'énergie

Tout contact avec des pièces sous tension peut entraîner la mort ou des blessures graves.

- Ne travailler sur des appareils électriques que si l'on a les compétences requises.
- Respecter les règles de sécurité propre au pays lors de toute intervention.

Les étapes suivantes doivent généralement être observées pour garantir les conditions de sécurité :

1. Préparer la mise hors tension. Informer toutes les personnes concernées par la procédure.
2. Mettre le système d'entraînement hors tension et le condamner dans cet état.
3. Attendre la fin du temps de décharge qui est indiqué sur les panneaux d'avertissement.
4. Vérifier l'absence de tension entre les connexions de puissance de même qu'entre ces dernières et le conducteur de protection.
5. Vérifier que les circuits de tension auxiliaire existants sont hors tension.
6. S'assurer que les moteurs ne peuvent pas tourner.
7. Identifier toutes les autres sources d'énergie dangereuses, par exemple de l'air comprimé, de l'énergie hydraulique ou de l'eau. Mettre les sources d'énergie en configuration de sécurité.
8. S'assurer que le bon système d'entraînement est complètement verrouillé.

Au terme des travaux, rétablir l'état de marche en suivant les étapes dans l'ordre inverse.

⚠ ATTENTION

Choc électrique dû à la connexion d'une alimentation électrique inappropriée

Lors du raccordement d'une alimentation électrique inappropriée, il se peut que des pièces accessibles soient sous une tension dangereuse. Tout contact direct avec des tensions dangereuses peut provoquer des blessures graves ou la mort.

- Pour tous les connecteurs et toutes les bornes des modules électroniques, utiliser uniquement des alimentations qui fournissent des tensions de sortie TBTS (très basse tension de sécurité) ou TBTP (très basse tension de protection).

⚠ ATTENTION

Choc électrique dû à des appareils endommagés

Une manipulation inappropriée risque d'endommager les appareils. En cas d'endommagement des appareils, des tensions dangereuses peuvent être présentes sur l'enveloppe ou sur des composants accessibles et entraîner, en cas de contact, des blessures graves ou la mort.

- Lors du transport, du stockage et du fonctionnement, respecter les valeurs limites indiquées dans les caractéristiques techniques.
- Ne jamais utiliser d'appareils endommagés.

⚠ ATTENTION

Choc électrique dû à des blindages de câble non connectés

Le surcouplage capacitif peut engendrer des tensions de contact mortelles lorsque les blindages de câbles ne sont pas connectés.

- Connecter les blindages de câbles et les conducteurs inutilisés des câbles au potentiel de terre de l'enveloppe, au moins d'un côté.

⚠ ATTENTION

Choc électrique dû à l'absence de mise à la terre

Lorsque des appareils de la classe de protection I ne sont pas connectés au conducteur de protection ou si cette connexion est incorrecte, des tensions élevées risquent d'être présentes au niveau de pièces accessibles et d'entraîner, en cas de contact, des blessures graves ou la mort.

- Mettre l'appareil à la terre conformément aux directives.

IMPORTANT

Endommagement de l'appareil dû à des outils de vissage inappropriés

Tout outil ou procédé de vissage inapproprié risque d'endommager les vis de l'appareil.

- Utiliser des empreintes de vis parfaitement adaptées à la tête de vis.
- Serrer les vis avec le couple de serrage indiqué dans la documentation technique.
- Utiliser une clé dynamométrique ou un tournevis de précision mécanique avec un capteur de couple dynamique et une limitation de vitesse

⚠ ATTENTION

Propagation d'incendie due à des appareils encastrables

En cas d'incendie, l'enveloppe des appareils encastrables ne peut pas empêcher le feu et la fumée de s'échapper. Il peut en résulter des dommages corporels et matériels graves.

- Incorporer les appareils encastrables dans une armoire électrique en métal de manière à protéger les personnes et le matériel du feu et de la fumée, ou bien protéger les personnes par d'autres mesures adéquates.
- S'assurer que la fumée s'échappe uniquement par des voies prévues à cet effet.

⚠ ATTENTION

Mouvement de machine intempestif déclenché par des équipements radio ou téléphones mobiles

L'utilisation d'équipements radio ou de téléphones mobiles à proximité immédiate des composants peut perturber le fonctionnement des appareils. Les dysfonctionnements risquent de porter préjudice à la sécurité fonctionnelle des machines et de mettre ainsi en danger les personnes ou de causer des dommages matériels.

- En cas d'approche à moins de 20 cm des composants, éteindre les équipements radio et les téléphones mobiles.
- Utiliser l'appli "SIEMENS Industry Online Support App" uniquement lorsque l'appareil est éteint.

⚠ ATTENTION

Incendie pour cause d'espaces de dégagements de circulation d'air insuffisants

Des dégagements de circulation d'air insuffisants peuvent entraîner une surchauffe des constituants et provoquer un dégagement de fumée et un incendie. Cela peut entraîner des blessures graves ou la mort, De plus, ils peuvent provoquer des défaillances plus fréquentes et réduire la durée de vie des appareils/systèmes.

- Respectez les distances minimales pour les dégagements de circulation d'air indiquées pour chaque composant.

IMPORTANT

Surchauffe en cas de montage en position inadéquate

Une position de montage de l'appareil non autorisée peut entraîner sa surchauffe et risque de l'endommager.

- Exploiter l'appareil uniquement en position de montage autorisée.

⚠ ATTENTION

Mouvement de machine intempestif dû à des fonctions de sécurité inactives

Des fonctions de sécurité inactives ou non adaptées peuvent déclencher des mouvements intempestifs des machines qui risquent de causer des blessures graves ou la mort.

- Tenir compte, avant la mise en service, des informations contenues dans la documentation produit correspondante.
- Effectuer, pour les fonctions conditionnant la sécurité, une évaluation de la sécurité de l'ensemble du système, y compris de tous les constituants de sécurité.
- S'assurer par un paramétrage adéquat que les fonctions de sécurité sont adaptées aux tâches d'entraînement et d'automatisation et qu'elles sont activées.
- Effectuer un test des fonctions.
- N'exploiter l'installation en production qu'après s'être assuré de l'exécution correcte des fonctions conditionnant la sécurité.

Remarque

Importantes consignes de sécurité relatives aux fonctions Safety Integrated

Si vous voulez utiliser les fonctions Safety Integrated, tenez compte des consignes de sécurité indiquées dans les manuels Safety Integrated.

⚠ 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 Endommagement d'appareils par des champs électriques ou des décharges électrostatiques.

Les composants sensibles aux décharges électrostatiques (ESD) sont des composants individuels, des connexions, modules ou appareils intégrés pouvant subir des endommagements sous l'effet de champs électrostatiques ou de décharges électrostatiques.

IMPORTANT

Endommagement d'appareils par des champs électriques ou des décharges électrostatiques.

Les champs électriques ou les décharges électrostatiques peuvent induire des perturbations de fonctionnement en raison de composants individuels, de connexions, modules ou appareils intégrés endommagés.

- Emballer, stocker, transporter ou expédier les composants, modules ou appareils électroniques uniquement dans l'emballage d'origine du produit ou dans d'autres matériaux appropriés comme du papier aluminium ou du caoutchouc mousse possédant des propriétés conductrices.
- Ne toucher les composants, modules et appareils que si vous êtes relié à la terre par l'une des méthodes suivantes :
  - Port d'un bracelet antistatique
  - Port de chaussures antistatiques ou de chaussures munies de bandes de terre antistatiques dans les zones ESD pourvues de planchers conducteurs
- Ne poser les composants, modules ou appareils électroniques que sur des surfaces conductrices (table à revêtement antistatique, mousse conductrice antistatique, sachets antistatiques, conteneurs antistatiques).

2.3 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.

2.4 Notes relatives 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.

2.5 Risques résiduels des systèmes d'entraînement (Power Drive Systems)

Le constructeur de la machine ou de l'installation doit tenir compte lors de l'évaluation des risques de sa machine ou installation conformément aux prescriptions locales en vigueur (par ex. Directive machine CE) des risques résiduels émanant des composants de commande et d'entraînement :

1. Mouvement incontrôlé de machines ou parties d'installations entraînées à la mise en service, en service, pendant la maintenance ou en cours de réparation en raison :
   - des défauts matériels et/ou logiciels des capteurs, de la commande, des actionneurs et de la connectique
   - les temps de réponse de la commande et des entraînements
   - des conditions d'exploitation et/ou ambiantes ne correspondant pas à la spécification
   - de la condensation / un encrassement ayant des propriétés conductrices
   - des erreurs de paramétrage, de programmation, de câblage et de montage
   - l'utilisation d'émetteurs-récepteurs radio ou de téléphones portables à proximité directe des composants électroniques
   - des impacts / dommages extérieurs
   - des rayons X, rayons ionisants ou rayons cosmiques (altitude)
2. En cas de défaut, des températures inhabituellement élevées peuvent apparaître à l'intérieur et à l'extérieur des composants avec possibilité de flamme et d'émission de lumière, de particules, de gaz etc., par ex. en raison
   - des composants défaillants
   - d'erreurs de logiciel
   - des conditions d'exploitation et/ou ambiantes ne correspondant pas à la spécification
   - des impacts / dommages extérieurs
3. Tension de contact dangereuses, par exemple en raison de
   - des composants défaillants
   - de l'influence de charges électrostatiques
   - de tensions induites par des moteurs en mouvement
   - des conditions d'exploitation et/ou ambiantes ne correspondant pas à la spécification
   - de la condensation / un encrassement ayant des propriétés conductrices
   - des impacts / dommages extérieurs
4. des champs électriques, magnétiques et électromagnétiques au cours du fonctionnement pouvant p. ex. présenter un danger pour les porteurs d'un stimulateur cardiaque, d'un implant ou d'objets métalliques en cas de distance insuffisante
5. dégagement de substances et d'émissions nocives pour l'environnement en cas de fonctionnement inapproprié et/ou d'élimination incorrecte des constituants
6. influences négatives sur les communications filaires des réseaux, par exemple lissage de consommation ou communication sur le réseau d'énergie.

Des informations plus détaillées sur les risques résiduels des composants d'un système d'entraînement sont donnés aux chapitres correspondant de la documentation technique utilisateur.

Création de boîtes de dialogue utilisateur 3

3.1 Ensemble fonctionnel

Vue d'ensemble

La fonction "Création de boîtes de dialogue utilisateur" assure l'adaptabilité qui permet à l'utilisateur de concevoir des interfaces spécifiques au client et à l'application dans SINUMERIK Operate.

Pou la création des boîtes de dialogue utilisateur, la commande offre un langage de script basé sur XML.

Ce langage de script permet d'afficher des menus et des masques de dialogue spécifiques à la machine dans SINUMERIK Operate, dans le groupe fonctionnel <CUSTOM>.

Tous les masques de dialogue peuvent être réalisés indépendamment de la langue. Dans ce cas, le système extrait les textes à afficher de la base de données linguistique fournie.

Utilisation

Les instructions XML définies donnent accès aux propriétés suivantes :

1. Affichage de boîtes de dialogue et mise à disposition de :
   - touches logicielles
   - Variables
   - texte et texte d'aide
   - graphiques et images d'aide
2. Appel de boîtes de dialogue par :
   - actionnement de touches logicielles (d'accès)
3. 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
   - créer de façon dynamique des parties de script exécutables et les intégrer dans le traitement du script

4. Déclenchement d'actions en :
   – affichant les boîtes de dialogue
   – saisissant des valeurs (variables)
   – actionnant des touches logicielles
   – quittant les boîtes de dialogue
5. Échange de données entre boîtes de dialogue
6. Variables
   – lecture (variables CN, AP, utilisateur)
   – écriture (variables CN, AP, utilisateur)
   – combinaison par opérateurs mathématiques, de comparaison ou logiques
7. Exécution de fonctions :
   – sous-programmes
   – fonctions de fichier
   – services PI
8. Prise en compte des niveaux de protection en fonction des groupes d'utilisateurs

Le chapitre "Balises XML" (Page 46) décrit les éléments valides (balises) pour le langage de script.

Remarque

La section suivante "Notions de base pour la configuration" ne se prétend pas exhaustive en ce qui concerne la description XML (Extensible Markup Language). Pour des informations complémentaires, il convient de se reporter à la littérature spécialisée correspondante.

3.2 Bases pour la configuration

Fichiers de configuration

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 à la livraison ; ils doivent d'abord être créés ou rechargés.

Un éditeur XML ou un autre éditeur de texte peut être utilisé pour la création des fichiers de configuration.

Remarque

Les noms de fichier doivent uniquement contenir des minuscules.

Principe de l'arborescence de menus

Plusieurs boîtes de dialogue liées entre elles constituent une arborescence de menus. Un lien existe lorsqu'il est possible de passer d'une boîte de dialogue à l'autre. À 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 tout autre boîte de dialogue désirée.

Sous le menu principal, il est possible de générer une autre arborescence de menus via les touches logicielles d'accès configurées :

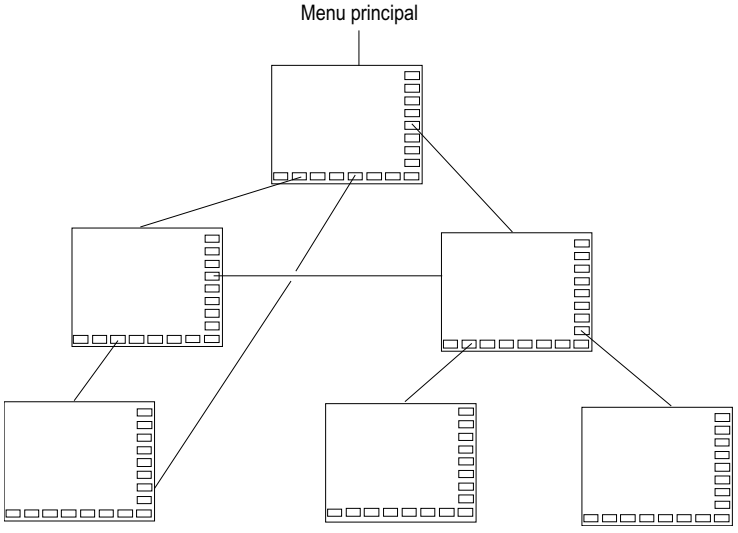


Figure 3-1 Arborescence de menus - Boîtes de dialogue utilisateur

Menu principal

Le menu principal est défini par le nom "main" dans le fichier "xmldial.xml". Il constitue le point de départ de séquences opératoires définies par l'utilisateur.

Le menu "main" permet de relier le chargement de boîtes de dialogue utilisateur ou de barres de touches logicielles supplémentaires qui permettent d'exécuter d'autres actions.

La figure suivante montre le répertoire constructeur "Carte CF système/oem/sinumerik/hmi" sur la commande.

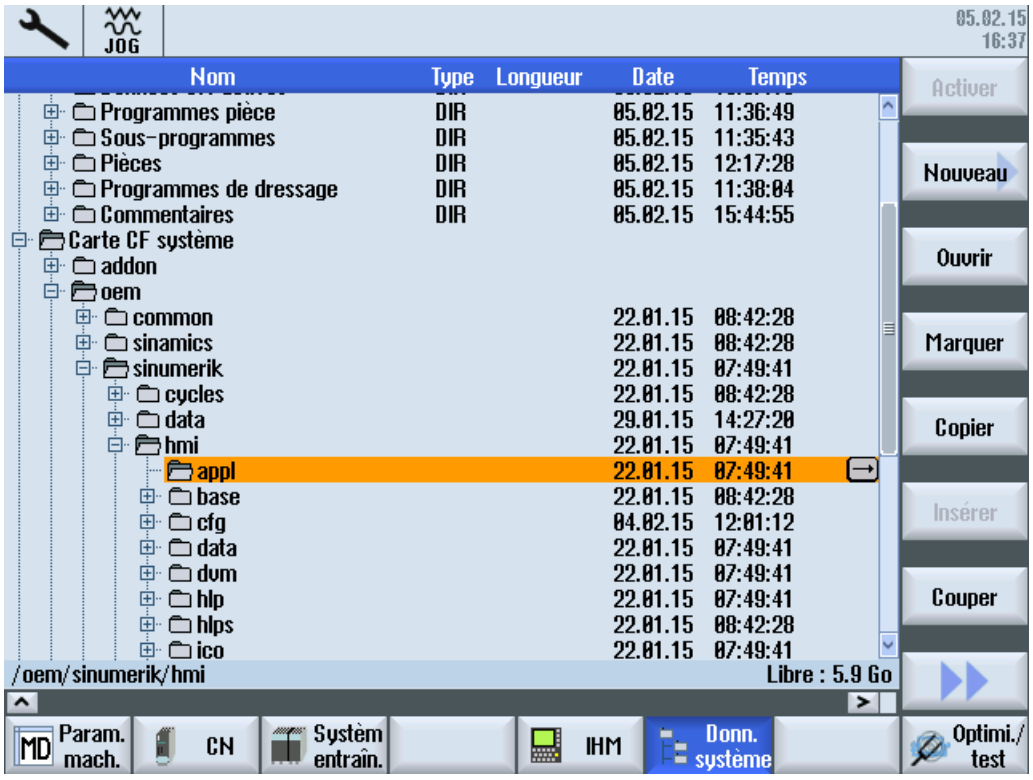


Figure 3-2 Répertoire constructeur

Les fichiers suivants du répertoire constructeur "Carte CF système/oem/sinumerik/hmi" sont requis dans la commande pour la configuration de boîtes de dialogue utilisateur :

Tableau 3-1 Fichiers pour la configuration

Type de fichier	Nom du fichier	Signification	Emplacement de stockage dans le groupe fonctionnel Mise en service > Données système
Fichier de script	"xmldial.xml"	Fichier texte standard Ce fichier de script gère au moyen de balises XML l'image des menus de touches logicielles et des masques de dialogue configurés de SINUMERIK Operate dans le groupe fonctionnel <CUSTOM>.	Répertoire constructeur > sous-répertoire "appl" pour les applications
Fichier de script spécifique à l'application	"xxx.xml"	Le nom du fichier peut être choisi librement. La référence à ces noms est établie via les fichiers INI correspondants.	

Type de fichier	Nom du fichier	Signification	Emplacement de stockage dans le groupe fonctionnel Mise en service > Données système
Fichier texte	"oem_xml_screens_xxx.ts"	Fichier texte standard pour le groupe fonctionnel Customer Ce fichier texte contient les textes des menus et masques de dialogue pour les différentes langues.	Répertoire constructeur > sous-répertoire "lng" pour les langues souhaitées
Fichier texte spécifique à l'utilisateur	"yyy_xxx.ts"	Le nom du fichier peut être choisi librement. La référence à ces noms est établie via les attributs correspondants aux balises DialogGUI, menu ou formulaire. La description se trouve au chapitre "Fichiers texte spécifiques au projet (Page 262)". Pour les applications Sidescreen ou Display Manager, le nom du fichier texte est dérivé du nom de la page ou du widget. Un exemple se trouve au chapitre "Gestion des langues et des textes (Page 237)".	
Bitmaps		La commande prend en charge les formats BMP et PNG.	Répertoire constructeur > sous-répertoire "ico"
Fichiers XML ajoutés dans le fichier de commande "xmldial.xml" avec la balise XML "INCLUDE".	par ex. "machine_settings.xml"	Ces fichiers contiennent également des instructions programmées pour la représentation des masques de dialogue et des paramètres dans SINUMERIK Operate.	Répertoire constructeur > sous-répertoire "appl" pour les applications

Points de liaison Easy XML

Les points de liaison disponibles pour les scripts Easy XML sont les suivants :

Touche matérielle/logicielle	Point de liaison

Softkey Operating Menu	Nom de fichier du script standard "xmldial.xml"
	Est activé dans le fichier "slamconfig.ini"
	
	Exemple :
	`[CustomXML]`
	`TextId=SL_AM_CUSTOM`
	`SidescreenTextId=SL_SIDESCREEN_CUSTOM`
	`TextFile=slam`
	`TextContext=SlAmAreaMenu`
	`SoftkeyPosition=12`
	`Visible=true`

Menu User	Nom de fichier du script standard "menu_user.xml"

Touche matérielle/logicielle	Point de liaison
Menu Function	Nom de fichier du script standard "menu_function.xml"

Touche matérielle AP	Le nom de fichier du script se rapporte à la description de la touche.
	
	Exemple :
	`KEY50.0 = area:=CustomXML, dialog:=SlEECustomDialog, cmdline:="-conf slagmdialog.hmi -mainModule restore.xml -entry cmc2"`
	`KEY51.0 = area:=CustomXML, dialog:=SlEECustomDialog, cmdline:="-conf slagmdialog.hmi -mainModule activate.xml -entry cmc1"`

Instruction MMC	Le nom de fichier du script se rapporte à la description de l'instruction CN. **Exemple :** `MMC("POPUPDLG, XML_ON, TEST.XML, cmd1","A)`

Touche matérielle/logicielle	Point de liaison

Touches logicielles réservées d'un groupe fonctionnel	Pour intégrer une application OEM dans un groupe fonctionnel, des modèles correspondants sont disponibles dans le répertoire siemens\sinumerik\hmi\template\cfg\ (sl<groupe_fonctionnel>_oem.xml). Copier le fichier XML appartenant au groupe fonctionnel dans le répertoire oem\sinumerik\hmi\cfg et le modifier comme suit :
	La fonction **switchToDialog** doit être programmée comme réaction de touche logicielle. Comme arguments, indiquer les mots-clés **area** avec la valeur **CustomXML** et **dialog** avec la valeur **SlEECustomDialog**.
	
	Syntaxe :
	`<FUNCTION args="-area CustomXML -dialog SlEECustomDialog -mainModule <name of the main module> -entry <menu name>" name="switchToDialog"/>`
	
	Exemple :
	Intégration d'une application Easy XML dans le groupe de diagnostic :
	"dg.xml" est utilisé comme nom de fichier du script. Dans ce fichier, le menu avec le nom **main** doit être appelé.
	`<MENU name="DgGlobalHu">`
	`<ETCLEVEL id="0">`
	`<SOFTKEYGROUP name="GroupEtc">`
	`<SOFTKEY position="7">`
	`<PROPERTY name="textID" type="QString">DG_SK</PROPERTY>`
	`<PROPERTY name="translationContext" type="QString">EASY_XML</PROPERTY>`
	`<FUNCTION args="-area CustomXML -dialog SlEECustomDialog -mainModule dg.xml -entry main" name="switchToDialog"/>`
	`</SOFTKEY>`
	`</SOFTKEYGROUP>`
	`</ETCLEVEL>`
	`</MENU>`

Sidescreen	Le nom de fichier du script se rapporte à la description Sidescreen du fichier "slsidescreen.ini".
Touche logicielle Easy Extend	Nom de fichier du script standard "agm.xml"

3.3 Fichiers de configuration

Introduction

La figure suivante illustre le répertoire constructeur "Carte CF Système/oem/sinumerik/hmi" sur la commande.

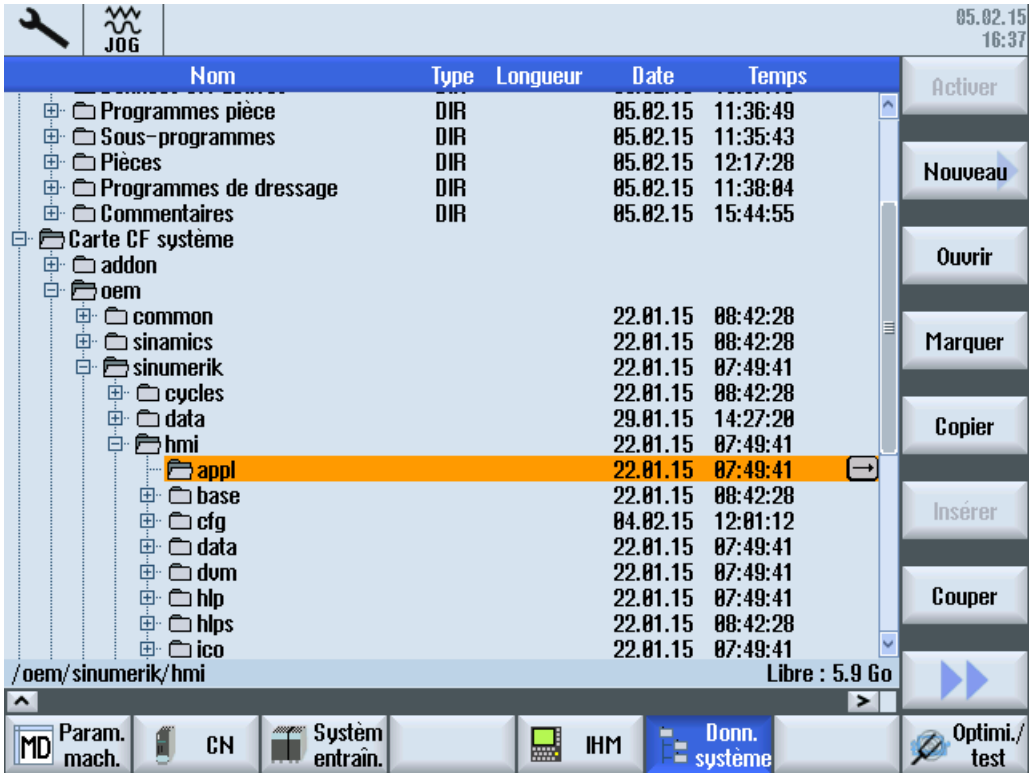


Figure 3-3 Répertoire constructeur

Les fichiers suivants du répertoire constructeur "Carte CF système/oem/sinumerik/hmi" sont requis dans la commande pour la configuration de boîtes de dialogue utilisateur :

Tableau 3-2 Fichiers pour la configuration

Type de fichier	Nom du fichier	Signification	Lieu d'archivage dans le groupe fonctionnel Mise en service > Données système
Fichier de script	"xmldial.xml"	Ce fichier de script gère au moyen de balises XML l'image des menus de touches logiciel-les et des masques de dialogue configurés de SINUMERIK Ope-rate dans le groupe fonctionnel <CUSTOM>.	Répertoire constructeur > sous-répertoire "appl" pour les applications
Fichier texte	"oem_xml_screens_xxx.ts"	Ce fichier texte contient les tex-tes des menus et masques de dialogue pour les différentes langues.	Répertoire constructeur > sous-répertoire "lng" pour les langues

Type de fichier	Nom du fichier	Signification	Lieu d'archivage dans le groupe fonctionnel Mise en service > Données système
Bitmaps		La commande prend en charge les formats BMP et PNG.	Répertoire constructeur > sous-répertoire "ico" Les bitmaps sont enregistrés dans les sous-répertoires pour la résolution d'écran inhérente à la commande. Remarque : Si un chemin d'accès au fichier bitmap est indiqué, les fichiers peuvent être enregistrés directement dans ce répertoire.
Fichiers XML ajoutés dans le fichier de commande "xmldial.xml" avec la balise XML "INCLUDE".	par ex. "machine_settings.xml"	Ces fichiers contiennent également des instructions programmées pour la représentation des masques de dialogue et des paramètres dans SINUMERIK Operate.	Répertoire constructeur > sous-répertoire "appl" pour les applications

Dépendances des fichiers pour la configuration des boîtes de dialogue utilisateur

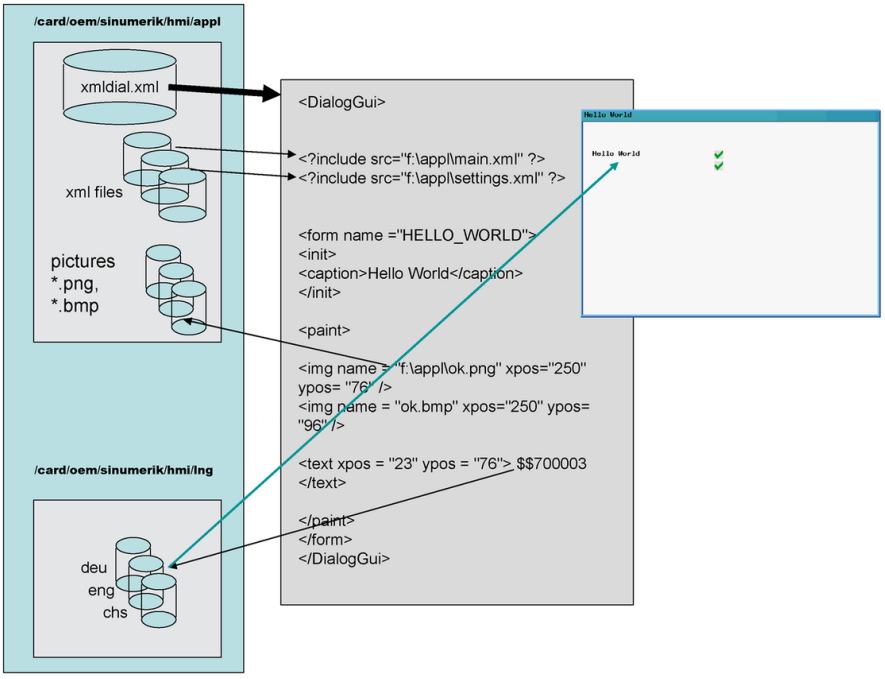


Figure 3-4 Dépendances

Chargement de la configuration

Comme décrit dans le tableau précédent "Fichiers pour la configuration" dans la colonne "Lieu d'archivage dans le groupe fonctionnel", les fichiers créés doivent être copiés dans les sous-répertoires correspondants du répertoire constructeur.

Remarque

Dès qu'un fichier de script "xmldial.xml" se trouve dans le sous-répertoire pour les applications, l'utilisateur peut démarrer cette boîte de dialogue utilisateur dans le groupe fonctionnel <CUSTOM>.

Après la toute première copie, une réinitialisation de l'IHM doit être effectuée à l'aide d'un "démarrage normal".

Exemple de boîte de dialogue utilisateur dans SINUMERIK Operate

Lors de l'appel du groupe fonctionnel <CUSTOM>, les touches logicielles configurées s'affichent. L'utilisateur peut utiliser les masques de dialogue configurés correspondants.

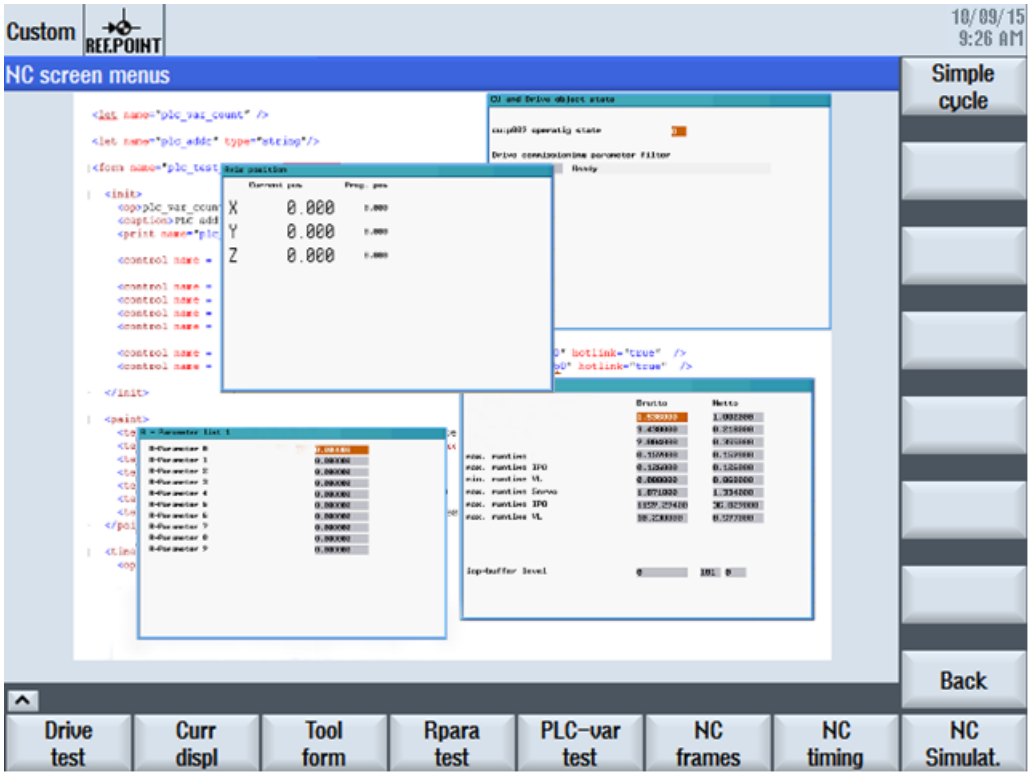


Figure 3-5 Exemple de boîte de dialogue utilisateur dans le groupe fonctionnel <CUSTOM>

D'autres exemples sont disponibles dans la Toolbox.

Voir aussi

Fonctions prédéfinies (Page 158)

3.4 Structure du fichier de configuration

Vue d'ensemble

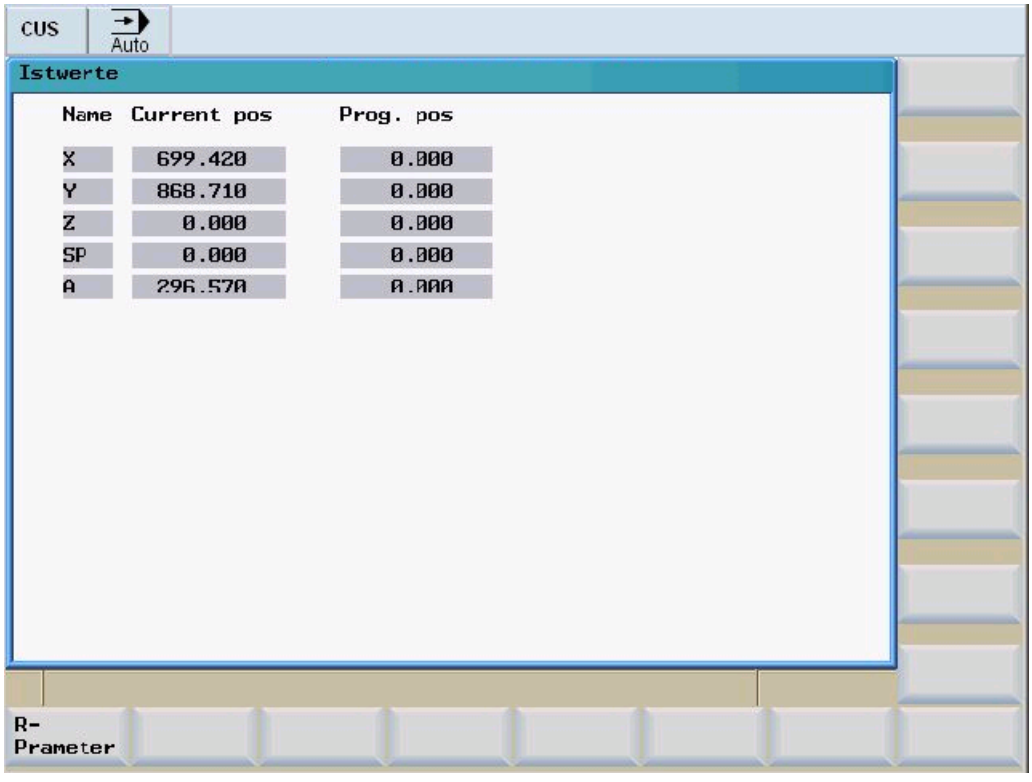
Un fichier de configuration se compose des éléments suivants :

- Description du menu principal "main" avec touches logicielles d'accès
- Définition des boîtes de dialogue
- Définition des variables
- Description des blocs
- Définition des barres de touches logicielles

Les exemples suivants montrent le script XML du fichier "xmldial.xml" et les captures d'écran correspondantes.

Le script contient les boîtes de dialogue pour l'affichage des valeurs réelles et des parcours restants ainsi qu'une liste de paramètres R.

Fenêtre de masque de dialogue Valeur réelle



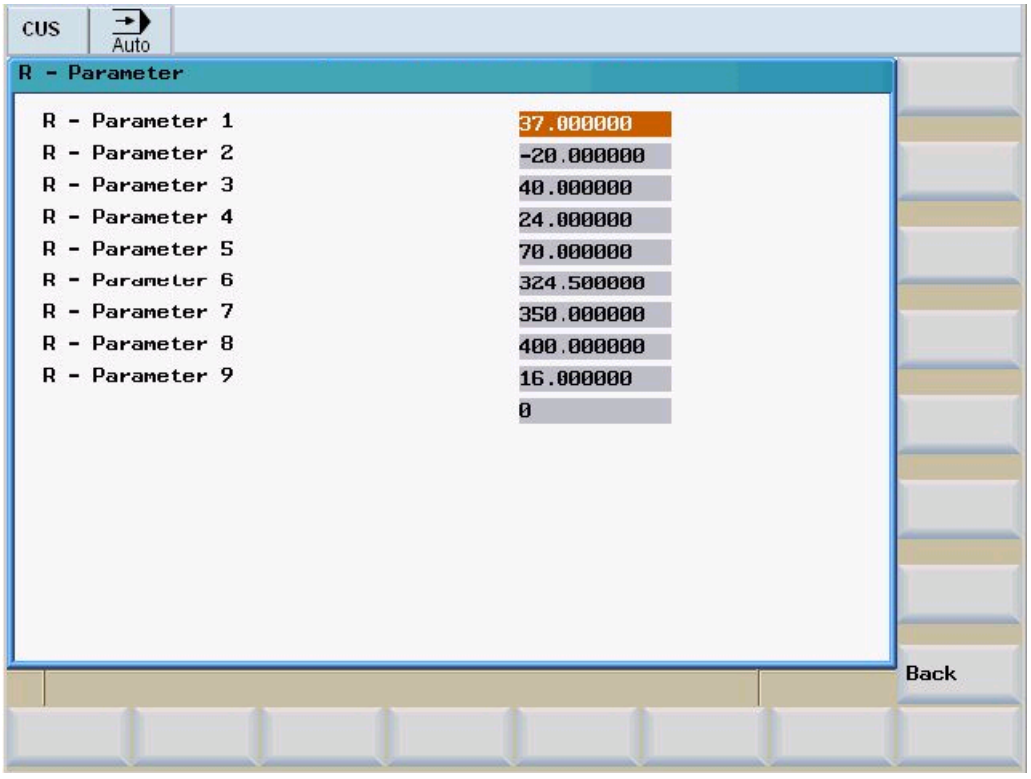
xmldial.xml

```
<DialogGui>

<!--
main menu
It is called by the system software. It starts the application.
The menu tag manages the soft key reactions. One input form can be assigned
to a menu tag.
-->
<menu name = "MAIN">
<OPEN_FORM name = "CURRENT_DISPLAY" />

<softkey POSITION="1">
<caption>R-%nPrameter</caption>
<navigation>MENU_R_PARAMETER</navigation> <!-- opens the menu R parameter --
>
</softkey>


</menu>


<form name = "CURRENT_DISPLAY">

<init>
<caption>Istwerte</caption>

<control name = "label1" xpos = "36" ypos = "56" width = "32"
fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[0]" />
<control name = "label2" xpos = "36" ypos = "76" width = "32"
fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[1]" />
<control name = "label3" xpos = "36" ypos = "96" width = "32"
fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[2]" />
<control name = "label4" xpos = "36" ypos = "116" width = "32"
fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[3]" />
<control name = "label5" xpos = "36" ypos = "136" width = "32"
fieldtype="readonly" refvar="nck/Channel/GeometricAxis/name[4]" />

<control name = "edit1" xpos = "80" ypos = "56" refvar="nck/Channel/
GeometricAxis/actProgPos[0]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
<control name = "edit2" xpos = "80" ypos = "76" refvar="nck/Channel/
GeometricAxis/actProgPos[1]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
<control name = "edit3" xpos = "80" ypos = "96" refvar="nck/Channel/
GeometricAxis/actProgPos[2]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
<control name = "edit4" xpos = "80" ypos = "116" refvar="nck/Channel/
GeometricAxis/actProgPos[3]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
<control name = "edit5" xpos = "80" ypos = "136" refvar="nck/Channel/
GeometricAxis/actProgPos[4]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
```

xmldial.xml

```
<control name = "edit11" xpos = "210" ypos = "56" refvar="nck/Channel/
GeometricAxis/progDistToGo[0]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
<control name = "edit12" xpos = "210" ypos = "76" refvar="nck/Channel/
GeometricAxis/progDistToGo[1]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
<control name = "edit13" xpos = "210" ypos = "96" refvar="nck/Channel/
GeometricAxis/progDistToGo[2]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
<control name = "edit14" xpos = "210" ypos = "116" refvar="nck/Channel/
GeometricAxis/progDistToGo[3]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>
<control name = "edit15" xpos = "210" ypos = "136" refvar="nck/Channel/
GeometricAxis/progDistToGo[4]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>

</init>

<paint>
<text xpos= "36" ypos="30">Name</text>
<text xpos= "80" ypos="30">Current pos</text>
<text xpos= "210" ypos="30">Prog. pos</text>

</paint>
</form>

.....
```

Fenêtre de masque de dialogue Paramètre R



xmldial.xml

```
.....

<!-- ************************************************************
Menu R- Parameter
-->

<menu name ="MENU_R_PARAMETER">
<OPEN_FORM name = "R-Parameter" />

<softkey POSITION="16">
<caption>Back</caption>
<navigation>MAIN</navigation>
</softkey>

</menu>

<form name = "R-Parameter">

<init>
<DATA_ACCESS type="true" />
<caption>R - Parameter</caption>

<control name = "edit1" xpos = "322" ypos = "34" refvar="nck/Channel/
Parameter/R[1]" />
<control name = "edit2" xpos = "322" ypos = "54" refvar="nck/Channel/
Parameter/R[2]" />
<control name = "edit3" xpos = "322" ypos = "74" refvar="nck/Channel/
Parameter/R[3]" />
<control name = "edit4" xpos = "322" ypos = "94" refvar="nck/Channel/
Parameter/R[4]" />
<control name = "edit5" xpos = "322" ypos = "114" refvar="nck/Channel/
Parameter/R[5]" />
<control name = "edit6" xpos = "322" ypos = "134" refvar="nck/Channel/
Parameter/R[6]" />
<control name = "edit7" xpos = "322" ypos = "154" refvar="nck/Channel/
Parameter/R[7]" />
<control name = "edit8" xpos = "322" ypos = "174" refvar="nck/Channel/
Parameter/R[8]" />
<control name = "edit9" xpos = "322" ypos = "194" refvar="nck/Channel/
Parameter/R[9]" />
<control name = "edit10" xpos = "322" ypos = "214" refvar="nck/Channel/
Parameter/R[10]" />
<control name = "edit10" xpos = "322" ypos = "214" refvar="plc/mb170" />
</init>

<paint>
<text xpos = "23" ypos = "34">R - Parameter 1</text>
<text xpos = "23" ypos = "54">R - Parameter 2</text>
<text xpos = "23" ypos = "74">R - Parameter 3</text>
<text xpos = "23" ypos = "94">R - Parameter 4</text>
<text xpos = "23" ypos = "114">R - Parameter 5</text>
<text xpos = "23" ypos = "134">R - Parameter 6</text>
<text xpos = "23" ypos = "154">R - Parameter 7</text>
<text xpos = "23" ypos = "174">R - Parameter 8</text>
<text xpos = "23" ypos = "194">R - Parameter 9</text>
```

xmldial.xml

```
</paint>
</form>

</DialogGui>
```

3.5 Textes localisés

Les textes localisés sont utilisés pour :

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

Les textes localisés sont enregistrés dans des fichiers texte.

Remarque

Les manipulations suivantes sont requises pour l'utilisation de ces fichiers texte :

- Mise à disposition des fichiers dans les langues requises.
- Transfert dans les répertoires de langue correspondants de la commande.

3.6 Diagnostic XML

Pour le diagnostic et la détection d'erreurs de script, le système propose la fonction Diagnostic Easy XML.

L'application de la fonction de diagnostic vérifie la syntaxe XML et parcourt tous les scripts faisant partie du projet. Le système vérifie également les fonctions, les menus, les formes, ainsi que l'existence et la validité des variables. Les erreurs détectées sont affichées dans une liste.

La fonction Diagnostic Easy XML peut être activée soit avec un attribut dans la balise **DialogGui**, soit via le paramètre machine d'affichage.

Activer la fonction Diagnostic Easy XML

Exemple :

```
<DialogGui diagnose="true" >
...
...
</DialogGui >
```

ou

Paramètre machine d'affichage

MD9113 $MM_EASY_XML_DIAGNOSE		Prise en charge de l'aide au diagnostic et à la correction pour scripts Easy XML
= 0	Aucun diagnostic actif	
= 1	Vérification de syntaxe active	
= 0x100	Les messages sont également enregistrés dans le fichier error_log.txt.	

Diagnostic par enregistrement des traces

L'enregistrement des traces peut être intégré à un script à l'aide de la balise **debug**. L'enregistrement n'a lieu que si l'attribut **debug_msg** est indiqué avec la valeur **on** dans la balise **DialogGui**.

Présentation fonctionnelle des touches logicielles

Boîte de dialogue	Touche logicielle	Fonction
Menu principal "Diagnostic EasyXML"	Démarrer script	Le projet chargé démarre directement.
	Démarrer vérification	La vérification de syntaxe démarre. Tous les messages générés peuvent être consultés. Il est possible de procéder à un nouveau contrôle.

Boîte de dialogue	Touche logicielle	Fonction
Boîtes de dialogue "Erreurs" et "Alarmes"	Erreurs	Le résultat est trié par erreurs et alarmes.
	Alarmes	
	Aller à l'erreur	Si des erreurs ou des alarmes ont été détectées, la touche logicielle est affichée. La touche logicielle ouvre le fichier XML marqué dans la liste des erreurs afin de l'éditer. Le curseur est placé automatiquement sur la ligne présentant une erreur.
	Résultat de la vérification	Tous les messages générés peuvent être consultés.
Boîte de dialogue "Document"	Enregistrer	Les modifications apportées au fichier XML sont enregistrées.
	Abandon	Le fichier XML est fermé sans être modifié. Le résultat de la vérification s'affiche à nouveau.

3.7 Descripteur XML

3.7.1 Structure générale

Structure et instructions du fichier de script pour la configuration de boîtes de dialogue

Toutes les configurations de boîtes de dialogue doivent être enregistrées à l'intérieur de la balise **DialogGui**.

```
<DialogGui>
…
</DialogGui>
```

Exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<DialogGui>
…
<FORM name ="Hello_World">
<INIT>
<CAPTION>Hello World</CAPTION>
</INIT>
...
</FORM>

</DialogGui>
```

Instructions

Le langage propose les instructions suivantes pour l'exécution des instructions conditionnelles et des boucles de programme :

- Boucle FOR
- Boucle WHILE
- Boucle DO_WHILE
- Traitement conditionnel
- Instructions SWITCH et CASE
- Éléments de commande d'un masque de dialogue
- Descriptions de touches logicielles
- Définition des variables

Les instructions sont décrites en détail dans le chapitre "Descriptions des instructions/ descripteurs (Page 47)".

3.7.2 Descriptions des instructions/descripteurs

Les **balises XML** suivantes sont définies pour la création des boîtes de dialogue et des menus ainsi que pour l'exécution de séquences de programme :

Remarque

Les valeurs d'attribut caractérisées par "<...>" entre guillemets doivent être remplacées par les expressions utilisées actuellement.

Exemple :
<DATA_LIST action="read/write/append" id="<list name>">
est programmé comme suit :

<DATA_LIST action="read/write/append" id="my datalist">

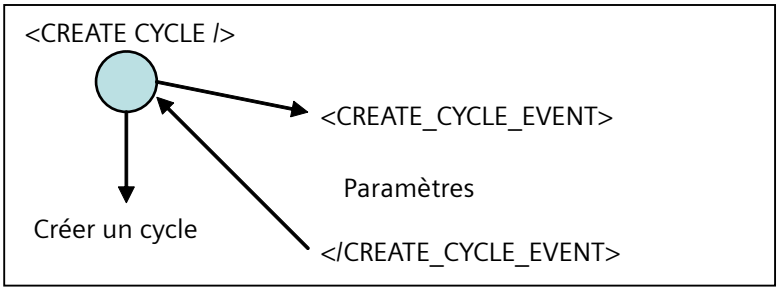
Lors de la copie de balises XML et d'exemples de plusieurs lignes, veiller à ce que les balisages ne soient pas séparés par des sauts de ligne indésirables.

Descripteur de balise	Signification
BREAK	Arrêt conditionnel d'une boucle.

CONDITION	La balise doit être programmée sur une ligne ou dans le cas d'une notation sur plusieurs lignes, les conditions doivent être spécifiées séparément du nom de la balise.
	
	Exemple :
	`<if>`
	`<condition>test &lt;= 2</condition>`
	`...`
	
	ou
	
	`<if>`
	`<condition>`
	`test &lt;= 2`
	`</condition>`
	`...`

CONTROL	La balise sert à la création d'éléments de commande. La description est donnée au chapitre "Création de menus de touches logicielles et de masques de dialogue (Page 79)".
CONTROL_RESET	La balise permet de redémarrer un ou plusieurs constituants de la commande. **Syntaxe :** `<CONTROL_RESET resetnc="TRUE" />` **Attributs :** • **RESETNC = "TRUE"** Le constituant de la CN est redémarré • **RESETDRIVE = "TRUE"** Les constituants de l'entraînement sont redémarrés.

Descripteur de balise	Signification

CREATE_CYCLE_EVENT	Si l'analyseur syntaxique lance le traitement de la balise CREATE_CYCLE, le message <CREATE_CYCLE_EVENT> est d'abord envoyé à la forme active. Ce message peut être utilisé pour le traitement des paramètres de cycle avant que l'analyseur syntaxique ne génère l'instruction CN à partir de la liste des paramètres et de la consigne de création.
	 
	
	Syntaxe :
	`<CREATE_CYCLE_EVENT>`
	`…`
	`…`
	`</CREATE_CYCLE_EVENT>`
	
	Exemple :
	`<SOFTKEY_OK>`
	`…`
	`<CREATE_CYCLE />`
	`…`
	`…`
	`<SOFTKEY_OK>`
	`…`
	`…`
	
	`<FORM>`
	`<NC_INSTRUCTION>MY_CYCLE($P1, $P2)</ NC_INSTRUCTION >`
	`…`
	`…`
	`<CREATE_CYCLE_EVENT>`
	`<type_cast name="P1" type="int"/>`
	
	`<OP> P1 = P1 * 150 </OP>`
	
	`</CREATE_CYCLE_EVENT>`
	`…`
	`…`
	
	`</FORM>`

Descripteur de balise	Signification

DATA	La balise permet l'écriture sur la CN, l'AP, les données globales utilisateur et les données d'entraînement.
	La formation de l'adresse est expliquée au chapitre "Adressage des composants (Page 144)".
	
	Attribut :
	• **name**
	Adresse de la variable
	
	Valeur de la balise :
	Une constante, **une variable ou un terme** sont attendus comme valeur de balise.
	
	Syntaxe :
	`<DATA name="<variable name>"> value </DATA>`
	`<DATA name="<variable name>"> <term> </DATA>`
	
	Exemple :
	`<DATA name = "plc/mb170"> 1 </DATA>`
	…
	`<LET name = "tempVar"> 7 </LET>`
	`<!– le contenu de la variable locale "tempVar" est écrit sur l'octet de mémento 170 →`
	`<DATA name = "plc/mb170">$tempVar</DATA>`

DATA Suite	**Exemple :**
	Calcul d'un terme. Le résultat est affecté aux variables spécifiées.
	`<let name="a" >#f</let>`
	`<let name="b" >7</let>`
	`<data name = "b"> a & b</data>`
	`<let name="c" type="string"></let>`
	`<data name = "c"> _T" test" + _T" and test"</data>`

Descripteur de balise	Signification

DATA_LIST	La balise permet la sauvegarde ou la restauration des paramètres d'entraînement et paramètres machine répertoriés.
	Le listage des adresses s'effectue ligne par ligne. La formation de l'adresse est expliquée au chapitre "Adressage des composants (Page 144)".
	Jusqu'à 20 listes de données temporaires peuvent être créées.
	
	Attributs :
	• **action**
	read – les valeurs des variables répertoriées sont enregistrées dans une mémoire temporaire
	append – les valeurs des variables répertoriées sont ajoutées à une liste existante
	write – les valeurs sauvegardées sont copiées dans les paramètres machine correspondants
	• **id**
	le descripteur sert à identifier la mémoire temporaire
	Syntaxe :
	`<DATA_LIST action="<read/write/append>"  id="<list name>">`
	`NC/PLC Listage d'adresses`
	`</DATA_LIST>`
	
	Exemple :
	`<DATA_LIST action ="read" id="<name>">`
	`nck/channel/parameter/r[2]`
	`nck/channel/parameter/r[3]`
	`nck/channel/parameter/r[4]`
	`$MN_USER_DATA_INT[0]`
	`…`
	`</ DATA_LIST>`
	`<DATA_LIST action ="write" id="<name>" />`

DIALOGGUI	Cette balise représente la balise racine pour la fonction Easy XML. Toutes les instructions incluses sont traitées par l'analyseur Easy XML. **Attributs :** Les attributs listés peuvent être spécifiés en option pour activer des fonctions centrales : • **diagnose** - Valeur **true** active le diagnostic XML • **debug_msg** - Valeur "on", l'enregistrement de messages traces est actif **Plus d'informations**, voir chapitre "Diagnostic XML (Page 44)" • **textfile** - Nom du fichier texte à utiliser • **textcontext** - Contexte de texte à utiliser, uniquement valable en relation avec l'attribut textfile • **textfilelist** - Permet de charger une liste avec différents fichiers texte **Plus d'informations**, voir chapitre "Fichiers texte spécifiques au projet (Page 262)"
DIALOGGUI Suite	• **restoresession** - Valeur **true** Si le projet Easy XML est sélectionné à plusieurs reprises, l'analyseur syntaxique ouvre le menu sélectionné en dernier et la forme sélectionnée en dernier. Ce comportement est maintenu jusqu'au redémarrage de l'IHM. **Plus d'informations**, voir chapitre "Restaurer la session (Page 268)"

Descripteur de balise	Signification
DEBUG_MSG	Cet attribut commande l'enregistrement des traces. La description est donnée au chapitre "Création de menus de touches logicielles et de masques de dialogue (Page 79)" sous la rubrique DEBUG. Si la valeur est définie sur **on**, l'analyseur enregistre tous les résultats dans un fichier. Cela peut entraîner une exécution plus lente du script. Par défaut, la valeur est définie sur **off**.

DO_WHILE	Boucle DO_WHILE
	`DO`
	`  Instructions`
	`WHILE (Test)`
	
	Syntaxe :
	`<DO_WHILE>`
	`Instructions`
	…
	`<CONDITION>…</CONDITION>`
	`</DO_WHILE>`
	La boucle Do-While est constituée d'un bloc d'instructions et d'une condition. Le code à l'intérieur du bloc d'instructions est d'abord exécuté, puis la condition est traitée. Si la condition est vraie (true), la fonction exécute la fraction de code. Cela se répète jusqu'à ce que la condition devienne fausse (false).
	
	Exemple :
	`<DO_WHILE>`
	`<DATA name = "PLC/qb11"> 15 </DATA>`
	`<CONDITIION> "plc/ib9" == 0 </CONDITION>`
	`</DO_WHILE>`

DYNAMIC_INCLUDE	La balise inclut un fichier de script XML.
	Contrairement à la balise INCLUDE, la lecture a seulement lieu lors du traitement de l'instruction correspondante.
	Pour les gros projets, l'utilisation de cette balise permet un chargement plus rapide du groupe Customer ou de l'assistance pour cycles. En outre, les besoins moyens en ressources diminuent, compte tenu que les boîtes de dialogue ne sont pas toujours toutes appelées au cours d'une session.
	
	Syntaxe :
	<DYNAMIC_INCLUDE src="path name"/>
	
	Exemple :
	`<SOFTKEY POSITION="3">`
	`  <CAPTION>MY_MENU</CAPTION>`
	`  <DYNAMIC_INCLUDE src="f:\appl\sk3_script.xml"/>`
	`  <NAVIGATION>MY_MENU</NAVIGATION>`
	</SOFTKEY>

ELSE	Instruction lorsque la condition n'a pas été remplie (**IF, THEN, ELSE**)

Descripteur de balise	Signification

FOR	Boucle FOR
	`for (initialisation ; test ; suite) instruction(s)`
	
	Syntaxe :
	`<FOR>`
	`<INIT>…</INIT>`
	`<CONDITION>…</CONDITION>`
	`<INCREMENT>…</INCREMENT>`
	`Instructions`
	`…`
	`</FOR>`
	La boucle FOR est réalisée comme suit :
	1. Traitement de l'expression **Initialisation** (INIT).
	2. Traitement de l'expression **Test** (CONDITION) en tant qu'expression booléenne. Si la valeur est "faux" (false), la boucle FOR est terminée.
	3. Exécution des instructions suivantes.
	4. Traitement de l'expression **Suite** (INCREMENT).
	5. Poursuivre avec l'étape 2.
	Toutes les variables utilisées qui sont exploitées dans la branche INIT, CONDITION et INCREMENT doivent être créées à l'extérieur de la boucle FOR.
	
	Exemple :
	`<LET name = "count">0</LET>`
	`<FOR>`
	`<INIT>`
	`<OP> count = 0</OP>`
	`</INIT>`
	`<CONDITION> count <= 7 </CONDITION>`
	`<INCREMENT>`
	`<OP> count = count + 1 </OP>`
	`</INCREMENT>`
	`<OP> "plc/qb10" = 1+ count </OP>`
	`</FOR>`

FORM	La balise contient la description d'une boîte de dialogue utilisateur. La description est donnée au chapitre "Création de menus de touches logicielles et de masques de dialogue (Page 79)".
HMI_RESET	La balise lance un redémarrage de l'IHM. L'interprétation est interrompue après cette instruction.

Descripteur de balise	Signification

IF	Instruction conditionnelle (**IF, THEN, ELSE**)
	Les balises THEN et ELSE sont incluses dans la balise IF.
	La balise IF est suivie de la condition exécutée par la balise CONDITION. Le résultat de l'opération décide du traitement complémentaire des instructions. Si le résultat de la fonction est vrai, la branche THEN est exécutée et la branche ELSE est ignorée. Si le résultat de la fonction est faux, l'analyseur syntaxique traite la branche ELSE.
	
	Syntaxe :
	`<IF>`
	`<CONDITION> Condition != 7 </CONDITION>`
	`<THEN>`
	`Instruction pour le cas suivant : Condition remplie`
	`</THEN>`
	`<ELSE>`
	`Instruction pour le cas suivant : Condition non remplie`
	`</ELSE>`
	`</IF>`
	
	Exemple :
	`<IF>`
	`<CONDITION> "plc/mb170" != 7 </CONDITION>`
	`<THEN>`
	`<OP> "plc/mb170" = 7 </OP>`
	…
	`</THEN>`
	`<ELSE>`
	`<OP> "plc/mb170" = 2 </OP>`
	…
	`</ELSE>`
	`</IF>`

IF Suite	Si seules des variables locales sont utilisées dans la condition, la condition peut être indiquée comme attribut dans la balise IF.
	
	Exemple :
	`<let name="var1">1</let>`
	`<if condition="var == 1">`
	`<then>`
	`...`
	`...`
	`</then>`
	`</if>`
	
	Remarque :
	Les contrôles qui ne sont pas liés à une variable de référence reçoivent le type de données nombre entier.
	Exceptions :
	Les contrôles de type **imagebox** et **multiline** sont affectés au type de données **string**.

Descripteur de balise	Signification
INCLUDE	L'instruction inclut une description XML. (voir aussi DYNAMIC_INCLUDE dans ce tableau) **Attribut :** • **src** contient le nom de chemin. **Syntaxe :** `<?INCLUDE src="<Path name>" ?>`

Descripteur de balise	Signification
LET	L'instruction crée une variable locale sous le nom spécifié. **Tableaux :** L'attribut **dim** (dimension) permet de créer des tableaux à une ou deux dimensions. L'adressage des différents éléments de tableau s'effectue via l'indice de tableau. Pour un tableau à deux dimensions, l'indice de ligne est d'abord indiqué, puis l'indice de colonne. • Tableau à une dimension : Indice 0 à 4 0 1 2 3 4 • Tableau à deux dimensions : Indice ligne 0 à 3 et indice colonne 0 à 5 0,0 0,1 0,2 0,3 0,4 0,5 1,0 1,1 1,2 1,3 1,4 1,5 2,0 2,1 2,2 2,3 2,4 2,5 3,0 3,1 3,2 3,3 3,4 3,5 **Attributs :** • **name** Nom de variable • **type** Le type de variable peut être un nombre entier (integer) (INT), un nombre entier non signé (unsigned) (UINT), double (DOUBLE), float (FLOAT), string (STRING) ou STRUCT. Il est par ailleurs possible d'utiliser une structure générée par **typedef** comme type de variable (voir le descripteur de balise TYPEDEF). Si aucune instruction de type n'est spécifiée, le système crée une variable Integer. `<LET name = "VAR1" type = "INT" />` • **permanent** Si l'attribut est défini sur **true**, la valeur de la variable est enregistrée de manière permanente. Cet attribut n'a d'effet que pour les variables globales. • **poweroff** Si la valeur de l'attribut est **true**, les valeurs sont conservées jusqu'à la mise à l'arrêt de la commande. **Plus d'informations**, voir chapitre "Restaurer la session (Page 268)" • **dim** Il convient d'indiquer le nombre d'éléments de tableau. Pour un tableau à deux dimensions, la deuxième dimension est spécifiée après la première, séparée par une virgule. L'accès à un élément de tableau s'effectue via l'indice de tableau spécifié après le nom de variable entre crochets. name[index] ou name[row,column] – Tableau à une dimension : dim="<Nombre d'éléments>" – Tableau à deux dimensions : dim="<Nombre de lignes>,<Nombre de colonnes>" Les éléments de tableau non initialisés se voient affecter par défaut la valeur "0".

Descripteur de balise	Signification

LET Suite	**Exemple :**
	Tableau à une dimension :
	`<let name="array" dim="10"></let>`
	Tableau à deux dimensions :
	`<let name="list_string" dim="10,3" type="string"></let>`
	
	Valeur par défaut :
	Une variable peut être initialisée avec une valeur.
	`<LET name = "VAR1" type = "INT"> 10 </LET>`
	
	Si les valeurs sont enregistrées dans une variable locale à partir de la CN ou de variables d'AP, l'opération d'affectation adapte le format automatiquement au format des variables lues.
	• Affectation par défaut d'une variable String :
	Il est possible d'affecter des textes de plusieurs lignes à une variable String en transmettant le texte formaté sous forme de valeur. Si une ligne doit être terminée par un line feed <LF> (saut de ligne) , il convient d'ajouter les caractères "\\n" à la fin de la ligne.
	`<LET name = "text" type = "string"> F4000 G94\\n`
	`G1 X20\\n`
	`Z50\\n`
	`M2\\n`
	`</LET>>`
	
	`Tableaux (Arrays) :`
	`<let name="list" dim="10,3">`
	`{1,2,3},`
	`{1,20}`
	`</let>`
	
	`<let name="list_string" dim="10,3" type="string">`
	`{"text 10","text 11"},`
	`{"text 20","text 21"}`
	`</let>`
	
	Affectation :
	Les valeurs provenant des paramètres machine ou des sous-routines peuvent être affectées à une variable à l'aide de l'opération d'affectation "=".
	La validité d'une variable s'étend jusqu'à la fin du bloc XML de niveau supérieur.
	Les variables qui doivent être disponibles globalement, doivent être créées directement après la balise **DialogGUi**.
	Pour une boîte de dialogue, il convient d'observer les points suivants :
	• Le traitement du message ouvre la balise correspondante.
	• La balise est fermée après l'exécution du message.
	• Toutes les variables contenues dans la balise sont supprimées avec la fermeture.

Descripteur de balise	Signification

LET Suite	**Type de variable struct** :
	Ce type de variable contient une liste de variables qui peuvent entrer en action lorsqu'on utilise le nom de la structure. Une structure peut contenir tous les types de variable ainsi que des structures.
	Une variable est déclarée à l'intérieur de la structure au moyen de la balise "element". Les attributs de la balise et l'initialisation correspondent aux attributs et à l'initialisation de l'instruction let.
	
	`<let name="name" type="struct">`
	`<element name="<nom>" type="<type de variable>" />`
	…
	…
	…
	`</let>`
	
	Une valeur par défaut peut être attribuée à un élément de structure utilisé comme valeur initiale pour la création d'une variable.
	Plus d'informations à la section "Initialisation de structures"
	Cette valeur est écrasée si une valeur initiale est indiquée lors de la création de la variable.

Descripteur de balise	Signification

LET Suite	L'accès à une variable de la structure s'effectue via le nom de structure et le nom de variable. Les deux noms sont séparés par l'opérateur point (.).
	
	Syntaxe :
	
	`<op>`
	`nom_structure.nom_variable = valeur ;`
	`</op>`
	
	Exemple :
	`<let name="info" type="struct">`
	`  <element name="id" type="int" />`
	`  <element name="name" type="string" />`
	`  <element name="phone" type="string" />`
	`</let>`
	
	`<op>`
	`  info.id = 1;`
	`  info.name = _T"my name";`
	`  info.phone = _T"0034 45634";`
	`</op>`

Descripteur de balise	Signification

LET Suite	**Initialisation de structures :**
	Les structures peuvent être initialisées lors de la création des variables en indiquant une valeur initiale par élément de structure. Dans le cas d'un champ de structures, il faut séparer chacune d'entre elles par des accolades.
	
	Exemple :
	`<typedef name="StructRect" type="struct" >`
	`  <element name="left" type="int">22</element>`
	`  <element name="top" type="int">122</element>`
	`  <element name="right" type="int">220</element>`
	`  <element name="bottom" type="int">56</element>`
	`</typedef>`
	`<let name="rect" type="StructRect" >`
	Lors de la création de la variable **rect**, les éléments de structure sont initialisés avec les valeurs suivantes :
	`rect.left = 22`
	`rect.top = 122`
	`rect.right = 220`
	`rect.bottom = 56`
	
	Si des valeurs initiales sont attribuées à une structure, celles-ci remplacent les valeurs par défaut.
	`<let name="rect" type="StructRect" >`
	`  {11,12}`
	`</let>`
	Lors de la création de la variable rect, les éléments de structure sont initialisés avec les valeurs suivantes :
	`rect.left = 11`
	`rect.top = 12`
	`rect.right = 220`
	`rect.bottom = 56`

Descripteur de balise	Signification

LET Suite	**Structures imbriquées :**
	Les structures sont des variables d'un type de variable spécifié par l'utilisateur et peuvent donc être utilisées comme éléments de structure dans une structure.
	Les éléments sont initialisés selon la règle décrite.
	
	Exemple :
	`<typedef name="StructRectRect" type="struct" >`
	`  <element name="r1" type="StructRect">77, 99, 232, 23</element>`
	`  <element name="r2" type="StructRect">77, 88, 45, 12</element>`
	`</typedef>`
	
	`<let name="rect_rect_array" type="StructRectRect" >`
	Lors de la création de la variable **rect_rect_array**, les éléments de structure sont initialisés avec les valeurs suivantes :
	`r1.left = 77`
	`r1.top = 99`
	`r1.right = 232`
	`r1.bottom = 23`
	`r2.left = 77`
	`r2.top = 88`
	`r2.right = 45`
	`r2.bottom = 12`
	
	`<let name="rect_rect_array1" type="StructRectRect" >`
	`  {{11,12,13,14},{111,188,199,114}}`
	`</let>`
	
	Lors de la création de la variable **rect_rect_array**, les éléments de structure sont initialisés avec les valeurs suivantes :
	`r1.left = 11`
	`r1.top = 12`
	`r1.right = 13`
	`r1.bottom = 14`
	`r2.left = 111`
	`r2.top = 188`
	`r2.right = 199`
	`r2.bottom = 114`

LET Suite	**Initialisation de variables String globales :** Si un identifiant texte est utilisé pour initialiser une variable String globale, l'identificateur et le texte à remplacer sont attendus dans le fichier texte standard **oem_xml_screens_xxx.ts** ou dans le fichier texte spécifié dans la balise **DialogGUI**. Au chargement de l'application, le texte de la langue active remplace l'identificateur de texte. Si un changement de langue est effectué pendant que l'application est active, il n'y a aucune nouvelle initialisation des variables String globales. Le texte peut être remplacé dans le message **LANGUAGE_CHANGED**.
LOCK_OPERATING_AREA	Le changement de groupe fonctionnel est verrouillé. La balise **UNLOCK_OPERATING_AREA** permet d'annuler le verrouillage du changement de groupe fonctionnel. **Syntaxe :** `<LOCK_OPERATING_AREA />`

Descripteur de balise	Signification
MSG	Le composant de commande affiche le message spécifié dans la balise. Si un numéro d'alarme est utilisé, la boîte de dialogue indique le texte enregistré sous ce numéro. **Exemple :** `<MSG text ="my message" />`

MSGBOX	L'instruction ouvre une boîte de message dont la valeur de retour peut être utilisée pour l'embranchement.
	
	Syntaxe :
	`<MSGBOX text="<Message>" caption="<caption>" retvalue="<variable name>" type="<button type>" />`
	
	Attributs :
	• **text**
	Texte
	• **caption**
	Titre
	• **retvalue**
	Nom de la variable dans laquelle est copiée la valeur de retour :
	1 – OK
	0 – CANCEL
	• **type**
	Possibilités d'acquittement :
	"BTN_OK"
	"BTN_CANCEL"
	"BTN_OKCANCEL"
	Si un numéro d'alarme est utilisé pour l'attribut **"text"** ou **"caption"** , la boîte de message indique le texte enregistré sous ce numéro.
	
	Exemple :
	`<MSGBOX text="Test message" caption="Information" retvalue="result" type="BTN_OK" />`

Descripteur de balise	Signification

OP	La balise exécute les opérations spécifiées.
	Les opérations qui peuvent être exécutées sont présentées au chapitre "Opérateurs (Page 77)".
	Pour l'accès aux paramètres d'entraînement et aux données de la CN / de l'AP, il convient de définir le nom de variable complet entre **guillemets**. La formation de l'adresse est expliquée au chapitre "Adressage des composants" (Page 144).
	`PLC: "PLC/MB170"`
	`NC: "NC/Channel/…"`
	
	Exemple :
	`<LET name = "tmpVar" type="INT"> </LET>`
	`<OP> tmpVar = "plc/mb170" </OP>`
	`<OP> tmpVar = tmpVar *2 </OP>`
	`<OP> "plc/mb170" = tmpVar </OP>`
	
	A l'intérieur d'une balise d'opération, plusieurs équations peuvent être utilisées. Un point-virgule marque la fin de l'instruction.
	
	Exemple :
	`<op>`
	`  x = x+1;`
	`  y = y+1;`
	`</op>`
	
	Traitement de la chaîne de caractères :
	L'instruction d'opération est en mesure de traiter des chaînes de caractères et d'affecter la chaîne de caractères résultante aux variables String spécifiées dans l'équation.
	Pour caractériser les expressions de texte, il convient de faire précéder le texte de l'identifiant _T. En outre, le formatage de valeurs de variables est possible. La règle de formatage doit être introduite avec l'identifiant _F, suivi de l'instruction de format. L'adresse de la variable est ensuite spécifiée.
	
	Exemple :
	`<LET name="buffer" type="string"></LET>`
	`…`
	`…`
	`…`
	`<OP> buffer = _T"unformatted value R0= " + "nck/Channel/Parameter/R[0]" + _T" and " + _T"$$85051" + _T" formatted value R1 " + _F %9.3f"nck/Channel/Parameter/R[1]" </OP>`

Descripteur de balise	Signification

OPERATION	**Opération**
	Une opération de décalage peut être utilisée à l'intérieur d'une équation.
	Opérateur Décaler vers la gauche "<<"
	La fonction << permet de décaler les bits vers la gauche. Vous pouvez définir la valeur à décaler et le nombre d'incréments de décalage directement ou à l'aide d'une variable. Si la limite du format de données est atteinte, les bits sont décalés au-delà de la limite sans qu'un message d'erreur ne s'affiche.
	Principe de l'exécution
	Exécution de gauche à droite et '+' et '-' avant '<<' ou '>>'
	
	Exemple :
	`<op> idx = idx << 2 </op>`
	`<op> idx = 3 + idx << 2 </op>`
	
	Opérateur Décaler vers la droite ">>"
	La fonction >> permet de décaler les bits vers la droite. Vous pouvez définir la valeur à décaler et le nombre d'incréments de décalage directement ou à l'aide d'une variable. Si la limite du format de données est atteinte, les bits sont décalés au-delà de la limite sans qu'un message d'erreur ne s'affiche.
	Principe de l'exécution
	Exécution de gauche à droite et '+' et '-' avant '<<' ou '>>'
	
	Exemple :
	`<op> idx = idx >> 2 </op>`
	`<op> idx = 3+idx >> 2</op>`

PASSWORD	Pour la fonction "EasyExtend", cette balise sert à l'activation de modules supplémentaires.
	La chaîne de caractères saisie est écrite dans la variable de référence indiquée et doit être traitée dans le programme utilisateur API.
	
	Syntaxe :
	`<PASSWORD refVar ="<variable name>" />`
	
	Attributs :
	• **refVar**
	Nom de la variable de référence
	• **text**
	L'indication d'un attribut remplace le texte standard (facultatif)
	
	Exemple :
	`<PASSWORD refvar="plc/mw107" />`
	
	Exemple facultatif :
	`<password refVar = "plc/MD108" text="Password" />`

POWER_OFF	Un message demande à l'opérateur d'arrêter la machine. Le texte du message est enregistré de manière permanente dans le système.

Descripteur de balise	Signification

PRINT	La balise génère un texte dans la ligne de la boîte de dialogue ou copie le texte dans la variable spécifiée.
	Si le texte contient des identifiants de formatage, les valeurs des variables sont insérées aux emplacements appropriés.
	
	Syntaxe :
	`<PRINT name="Nom de variable" text="text %Formatage"> Variable, … </PRINT>`
	`<PRINT text="text %Formatage"> Variable, … </PRINT>`
	Attributs :
	• **name**
	Nom de la variable dans laquelle le texte doit être enregistré (facultatif)
	• **text**
	Texte
	
	Formatage :
	Le caractère "%" entraîne le formatage des variables spécifiées en tant que valeur.
	%[Mémentos] [Largeur] [.Chiffres après la virgule] Type
	• Mémentos :
	Caractère facultatif pour définir le formatage d'édition :
	– justifié à droite ou justifié à gauche ("-" pour justifié à gauche)
	– Ajouter des zéros de tête ("0")
	– Remplir avec des espaces
	• Largeur :
	L'argument définit la largeur minimale d'affichage d'un nombre non négatif. Si la valeur à afficher 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, le paramètre facultatif définit le nombre de chiffres après la virgule.
	• Type :
	Le caractère de type définit les formats de données transmis à l'instruction d'impression (Print). Ces caractères doivent être spécifiés.
	– d : valeur du type Integer
	– f : nombre à virgule flottante
	– s : valeur du type string

Descripteur de balise	Signification

PRINT Suite	**Valeurs :**
	Nombre de variables dont les valeurs doivent être insérées dans le texte.
	Les types de variable doivent coïncider avec l'identifiant de type correspondant de la règle de formatage et être séparés les uns des autres par des virgules.
	
	Exemple :
	Affichage d'un texte dans la ligne d'information
	`<PRINT text="Texte d'info" />`
	Edition d'un texte avec formatage de variable
	`<LET name="trun_dir"></LET>`
	`<PRINT text="M%d">trun_dir</PRINT>`
	Edition d'un texte dans une variable String avec formatage de variable
	`<LET name="trun_dir"></LET>`
	`<LET name="str" type="string" ></LET>`
	`<print name="str" text="M%d ">trun_dir</print>`

PROGRESS_BAR	La balise ouvre ou ferme une barre de progression. La barre est affichée en dessous de la fenêtre de l'application.
	
	Syntaxe :
	`<PROGRESS_BAR type="<true/false>"> value </ PROGRESS_BAR>`
	
	Attributs :
	• **type =** "TRUE" - ouvre la barre de progression
	• **type** = "FALSE" - ferme la barre de progression
	• **min**
	(facultatif) - valeur minimale
	• **max**
	(facultatif) - valeur maximale
	
	Valeur :
	• **Value**
	Pourcentage de progression de la barre
	Exemple :
	`<PROGRESS_BAR type="true" min="0" max="101" >20< /`
	`PROGRESS_BAR>......<PROGRESS_BAR >50< /`
	`PROGRESS_BAR>......<PROGRESS_BAR type="false" >100< /PROGRESS_BAR>`

Descripteur de balise	Signification

SEND_MESSAGE	La balise envoie un message avec deux paramètres relatifs à la forme active traitée dans la balise Message.
	
	Syntaxe :
	`<SEND_MESSAGE>p1, p2</SEND_MESSAGE>`
	
	Exemple :
	`<SOFTKEY POSITION="3">`
	`  <caption>Set%nParameter</caption>`
	`  <send_message>1, 0</send_message>`
	`</SOFTKEY>`
	
	`<FORM>`
	…
	…
	`<MESSAGE>`
	`<SWITCH>`
	`<CONDITION>$message_par1</CONDITION>`
	`<CASE value="1">`
	…
	…
	`</CASE>`
	`</SWITCH>`
	`</MESSAGE>`
	…
	…
	
	`</FORM>`

SHOW_CONTROL	La visibilité d'un élément de commande peut être gérée à l'aide de la balise.
	
	Syntaxe :
	`<SHOW_CONTROL name="<name>" type="<type>" />`
	
	Attributs :
	• **name**
	Nom de l'élément de commande
	• **type =** "TRUE" - L'élément de commande est visible
	• **type** = "FALSE" - L'élément de commande n'est pas visible (masqué)
	
	Exemple :
	`<SHOW_CONTROL name="myEditfield" type="false" />`
	…
	…
	`<SHOW_CONTROL name="myEditfield" type="true" />`

Descripteur de balise	Signification

SLEEP	La balise interrompt l'exécution du script pour l'intervalle de temps spécifié. Le temps d'interruption résulte de la valeur transmise multipliée par la base temps de 50 ms.
	
	Syntaxe :
	`<SLEEP value="Temps d'interruption" />`
	
	Exemple :
	Temps d'attente 1,5 s
	`<SLEEP value="30" />`

STOP	L'interprétation est interrompue à cet endroit.

SWITCH	L'instruction **SWITCH** décrit une sélection multiple. Une expression est évaluée une fois et comparée avec un nombre de constantes. Si l'expression coïncide avec les constantes, les instructions sont exécutées à l'intérieur de l'instruction **CASE**.
	L'instruction **DEFAULT** est traitée si aucune constante ne coïncide avec l'expression.
	
	Syntaxe :
	`<SWITCH>`
	`<CONDITION> Valeur </CONDITION>`
	`<CASE value="<Konstante 1>">`
	`Instructions`
	`…`
	`</CASE>`
	`<CASE value="<Konstante 2>">`
	`Instructions`
	`…`
	`</CASE>`
	`<DEFAULT>`
	`Instructions`
	`…`
	`</DEFAULT>`
	`</SWITCH>`

Descripteur de balise	Signification

SWITCHTOAREA	La balise SWITCHTOAREA permet de passer au groupe fonctionnel indiqué à partir du groupe Customer.
	Le paramètre est indiqué comme valeur d'attribut.
	
	Syntaxe :
	`<switchToArea name="area" args="argument "/>`
	
	Attributs :
	name
	Les noms de groupes fonctionnels suivants ont été accordés pour les groupes fonctionnels :
	• **AreaMachine** - Machine
	• **AreaParameter** - Paramètres
	• **AreaProgramEdit** - Editeur
	• **AreaProgramManager** - Gestionnaire de programmes
	• **AreaDiagnosis** - Diagnostic
	• **AreaStartup** - Mise en service
	args (réservé)
	Les arguments de durée peuvent être également indiqués avec le groupe fonctionnel pour l'activation des boîtes de dialogue.
	
	Exemple :
	`<switchToArea name="AreaMachine" />`

SWITCHTODYNAMICTARGET	Si le dialogue précédent se définit comme destination dynamique de saut, la balise SWITCHTODYNAMICTARGET active ce dialogue et met fin au traitement du script. **Syntaxe :** `<switchToDynamicTarget />`
THEN	Instruction si la condition a été remplie (**IF, THEN, ELSE**)
TYPE_CAST	La balise sert à convertir le type de données d'une variable locale. **Syntaxe :** <type_cast name="variable name" type=" new type" /> **Attributs :** • **name** Nom de la variable • **type** La variable est affectée au nouveau type de données. • **convert** La variable est affectée au nouveau type de données. En outre, il se produit une conversion de la valeur de la variable dans le nouveau type de données.

Descripteur de balise	Signification

TYPEDEF	La balise permet de définir un nouveau descripteur pour un type de données. Pour les définitions de structures, l'avantage est que l'on définit une fois le type de données et que celui-ci peut ensuite être utilisé comme type de données dans une instruction LET.
	Le descripteur et le type sont attendus comme attributs.
	L'analyseur syntaxique prend en charge uniquement la mise en place de définitions de structures.
	
	Dans la définition de type, une variable est déclarée au moyen de la balise "element". Les attributs de la balise correspondent aux attributs de l'instruction let.
	
	`<typedef name="<descripteur>" type="struct">`
	`  <element name="<nom>" type="<type de variable>" />`
	…
	…
	…
	`</typedef>`
	
	Après définition, le descripteur peut être utilisé comme type de données pour l'instruction LET.
	
	`<let name="<nom de variable>" type="<descripteur>"></let>`
	
	Exemple :
	`<typedef name="my_struct" type="struct">`
	`  <element name="id" type="int" />`
	`  <element name="name" type="string" />`
	`  <element name="phone" type="string" />`
	`</typedef>`
	
	`<let name="info" type="my_struct"></let>`
	…
	…
	`<op>`
	`  info.id = 1;`
	`  info.name = _T"my name";`
	`  info.phone= _T"0034 45634";`
	`</op>`

Descripteur de balise	Signification

TYPEDEF Suite	Certains fonctions prédéfinies attendent comme paramètres d'appel des variables de type de structure RECT, POINT ou SIZE. Ces structures sont définies dans le fichier struct_def.xml.
	Plus d'informations, voir chapitre "Créer un fichier struct_def.xml (Page 142)"
	
	RECT :
	`<typedef name="StructRect" type="struct" >`
	`  <element name="left" type="int">0</element>`
	`  <element name="top" type="int">0</element>`
	`  <element name="right" type="int">0</element>`
	`  <element name="bottom" type="int">0</element>`
	`</typedef>`
	
	POINT :
	`<typedef name="StructPoint" type="struct" >`
	`  <element name="x" type="int">0</element>`
	`  <element name="y" type="int">0</element>`
	`</typedef>`
	
	SIZE :
	`<typedef name="StructSize" type="struct" >`
	`  <element name="width" type="int">0</element>`
	`  <element name="height" type="int">0</element>`
	`</typedef>`

UNLOCK_OPERATING_AREA	Annulation du verrouillage du changement de groupe fonctionnel **Syntaxe :** `<UNLOCK_OPERATING_AREA />`

WAITING	La balise attend le redémarrage du constituant après une réinitialisation de la CN ou de l'entraînement.
	
	Attributs :
	• **WAITINGFORNC** = "TRUE" - en attente de redémarrage de la CN
	• **WAITINGFORDRIVE** = "TRUE" - en attente de redémarrage des entraînements
	
	Syntaxe :
	`<WAITING WAITINGFORNC = "TRUE"/>`
	
	Exemple :
	...
	`<CONTROL_RESET resetnc = "true" resetdrive = "true"/>`
	`<WAITING waitingfornc = "true" waitingfordrive = "true" />`
	...

Descripteur de balise	Signification

WHILE	Boucle While
	`WHILE (Test)`
	`  Instruction`
	
	Syntaxe :
	`<WHILE>`
	`<CONDITION>…</CONDITION>`
	`Instructions`
	…
	`</WHILE>`
	La boucle While sert à exécuter plusieurs fois une séquence d'instructions tant qu'une condition est remplie. Cette condition est vérifiée avant l'exécution de la séquence d'instructions.
	
	Exemple :
	`<WHILE>`
	`<CONDITION> "plc/ib9" == 0 </CONDITION>`
	`<DATA name = "PLC/qb11"> 15 </DATA>`
	`</WHILE>`

XML_PARSER	La balise "XML_PARSER" peut être utilisée pour l'analyse syntaxique de fichiers XML. L'analyseur syntaxique interprète un fichier XML et appelle des fonctions de rappel définies. Chaque fonction de rappel fait partie d'un événement prédéfini. Le programmeur peut traiter les données XML à l'intérieur de cette fonction. **Evénements prédéfinis :** • start document L'analyseur syntaxique ouvre le document et commence l'analyse. • end document L'analyseur syntaxique ferme le document. • start element L'analyseur syntaxique a trouvé un élément et crée une liste avec tous les attributs et valeurs d'attribut. Ces listes sont transmises à la fonction de rappel. • end element La fin de l'élément a été trouvée. • characters L'analyseur syntaxique transmet tous les caractères d'un élément. • error L'analyseur syntaxique a détecté une erreur de syntaxe. Lorsqu'un événement se produit, l'analyseur syntaxique appelle la fonction de rappel et vérifie la valeur de retour de la fonction. Si la fonction retourne la valeur "true", l'analyseur syntaxique poursuit le processus.

Descripteur de balise	Signification

XML_PARSER Suite	**Interfaces**
	La valeur de l'attribut **name** contient le chemin d'accès au fichier XML.
	Pour affecter des événements aux fonctions de rappel, il convient de définir les propriétés suivantes :
	Standard
	startElementHandler
	endElementHandler
	charactersHandler
	Optionnel
	errorHandler
	documentHandler
	La valeur d'un attribut définit le nom de la fonction de rappel.
	
	Exemple :
	`<XML_PARSER name="f:\appl\xml_test.xml">`
	`<!-- standard handler -->`
	`<property startElementHandler="startElementHandler" />`
	`<property endElementHandler="endElementHandler" />`
	`<property charactersHandler="charactersHandler" />`
	`<!-- optional handler -->`
	`<property errorHandler="errorHandler" />`
	`<property documentHandler="documentHandler" />`
	`</XML_PARSER>`

Descripteur de balise	Signification

XML_PARSER Suite	En outre, l'analyseur syntaxique fournit des variables afin que les fonctions de rappel puissent accéder aux données d'événement.
	
	startElementHandler :
	Paramètres de fonction
	tag_name - Nom de balise
	num - Nombre d'attributs trouvés
	Variables système
	$xmlAttribute Tableau String avec le nombre d'éléments indiqués par num.
	$xmlValue Tableau String qui contient la plage de valeurs d'attribut 0-num.
	
	Exemple :
	`<function_body name="startElementHandler" return="true" parameter="tag_name, num">`
	`<print text="attribute name %s"> $xmlAttribute[0]</print>`
	`<print text="attribute value %s"> $xmlValue[0]</print>`
	`</function_body>`
	
	endElementHandler :
	Paramètres de fonction
	tag_name - Nom de balise
	
	Exemple :
	`<function_body name="endElementHandler" parameter="tag_name">`
	`<print text="name %s"> tag_name </print>`
	`</function_body>`

Descripteur de balise	Signification

XML_PARSER Suite	**charactersHandler :**
	Variables système
	$xmlCharacters Chaîne avec les données
	$xmlCharactersStart toujours 0
	$xmlCharactersLength Nombre d'octets
	
	Exemple :
	`<function_body name="charactersHandler" return="true" >`
	`  <print text="chars %s"> $xmlCharacters </print>`
	`</function_body>`
	
	documentHandler :
	Paramètres de fonction
	state 1 start document, 2 end document
	
	errorHandler :
	Variables système
	$xmlErrorString Contient la ligne non valide (variable système)
	
	Exemple :
	`<function_body name="errorHandler" >`
	`  <print text="error %s">$xmlErrorString</print>`
	`</function_body>`
	
	Résultat de rappel :
	$return si 1 (true), l'analyseur syntaxique poursuit l'analyse du fichier

3.7.3 Code couleur

L'attribut color utilise le schéma de code couleur du langage HTML.

Du point de vue syntaxique, une indication de couleur se compose du caractère "#" (losange) et de six chiffres du système hexadécimal, chaque couleur étant représentée par deux chiffres.

R – rouge

G – vert

B – bleu

#RRGGBB

Exemple :

color= "#ff0011"

Exemples de couleurs :

rouge	vert	bleu	jaune	blanc	noir
#FF0000	#00FF00	#0000FF	#ffff00	#FFFFFF	#000000

3.7.4 Syntaxe XML spéciale

Les caractères qui ont une signification particulière pour la syntaxe XML doivent être réécrits si ceux-ci doivent être représentés correctement par un éditeur XML général.

Les caractères concernés sont les suivants :

Caractère	Notation en XML
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;

3.7.5 Caractères spéciaux HTML

Pour la représentation des caractères ou l'utilisation de caractères de commande, l'analyse de caractères spéciaux HTML est proposée.

Les codages décimaux et hexadécimaux de la page de code Latin 1 peuvent être utilisés.

Exemple

Caractères	Description	Notation décimale	Notation hexadécimale
"	Guillemets en haut	"	"
LF	Line Feed : présentation de ligne

3.7.6 Opérateurs

L'instruction d'opération traite les opérateurs suivants :

Opérateur	Signification
=	Affectation
==	égal à
<, <	inférieur à
>, >	supérieur à
<=, <=	inférieur ou égal à
>=, >=	supérieur ou égal à
|	opération OU bit par bit
||	opération OU logique
&, &	opération ET bit par bit
&&, &&	opération ET logique
+	Addition
-	Soustraction
*	Multiplication
/	Division
!	Non
!=	différent de

Les instructions d'opération sont traitées de gauche à droite. Selon les cas, la mise entre parenthèses des expressions peut être utile afin de définir la priorité d'exécution des sous-expressions.

3.7.7 Variables système

Les variables système sont des variables disponibles dans chaque script et servant à l'échange de données entre l'analyseur syntaxique et l'exécution du script.

Le tableau suivant donne un aperçu des variables générées automatiquement et des balises auxquelles elles sont associées :

Nom de variable	Signification	Valide dans la balise
$actionresult	Indique à l'analyseur syntaxique si celui-ci doit traiter l'événement	KEY_EVENT
$focus_name	Contient le nom du tableau qui est activé	FOCUS_IN INDEX_CHANGED EDIT_CHANGED
$focus_item_data	Contient la valeur numérique item_data ayant été affectée au tableau	
$gestureinfo	Pour un geste du doigt, l'analyseur syntaxique fournit les informations sur les gestes dans la variable et exécute la balise gesture_event.	GESTURE_EVENT
$return	La variable sert à transporter la valeur de retour d'une sous-fonction	FUNCTION_BODY
$message_par1 $message_par2	Contient les paramètres d'appel de la fonction SEND_MESSAGE	MESSAGE
$xmlAttribute	Contient une liste des attributs de balise trouvés	XML_PARSER startElementHandler
$xmlValue	Contient une liste des valeurs de balise trouvées	
$xmlCharacters	Contient le flux de données	XML_PARSER charactersHandler
$xmlCharactersStart	Contient l'indice de départ	
$xmlCharactersLength	Contient le nombre de caractères enregistrés dans le flux de données	
$mouse_event.type $mouse_event.x $mouse_event.y $mouse_event.id $mouse_event.buttons $mouse_event.button	Structure pour le transfert des paramètres de l'événement de souris	MOUSE_EVENT

3.7.8 Création de menus de touches logicielles et de masques de dialogue

Pour l'intégration de masques de dialogue, une balise de menu avec le nom "main" doit être présente dans la description XML. Cette balise est appelée par le système après l'activation du groupe fonctionnel <CUSTOM>. D'autres branches de masques de dialogue ainsi que l'activation d'une boîte de dialogue peuvent être définies à l'intérieur de la balise.

```
<menu name= "MAIN">
<OPEN_FORM name= "main dialogue">
<softkey POSITION="1">
<caption>sub menu 1</caption>
<navigation>sub menu 1</navigation>
</softkey>
<softkey POSITION="8">
<caption>sub menu 8</caption>
<navigation>sub menu 8</navigation>
</softkey>
</menu>

<menu name= "sub menu 1">
<OPEN_FORM name= "dialogue 1">
</menu>

<menu name= "sub menu 8">
<OPEN_FORM name= "dialogue 8“>
</menu>
```

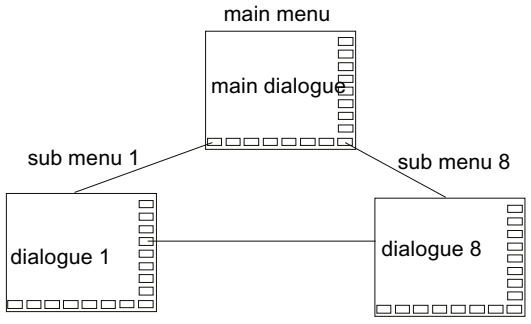


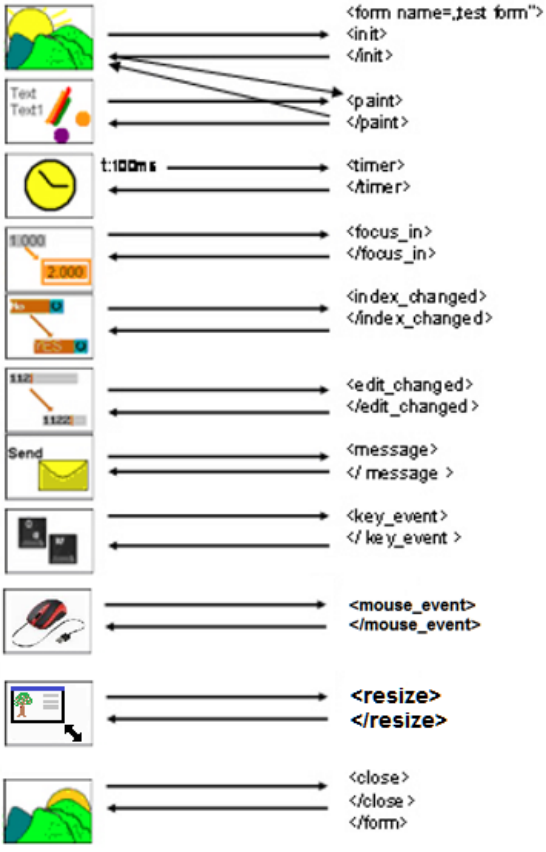
Figure 3-6 Structure du menu

Descripteur de balise	Signification
FORM	La balise contient la description d'une boîte de dialogue utilisateur. Les balises correspondantes sont décrites au chapitre Création de menus et de masques de dialogue. **Syntaxe :** `<FORM name="<dialog name>" color="#ff0000">` **Attributs :** • **color** Couleur d'arrière-plan du masque de dialogue **Plus d'informations**, voir chapitre "Code couleur (Page 75)" – Blanc par défaut • **name** Descripteur de la forme • **type** La valeur admissible est *cycle* , qui caractérise un masque de cycle utilisateur • **xpos** Position X du coin supérieur gauche de la boîte de dialogue (facultatif) • **ypos** Position Y du coin supérieur gauche (facultatif) • **width** Extension dans la direction X (en pixels) (facultatif) • **height** Extension dans la direction Y (en pixels) (facultatif)

FORM Suite	Avec l'attribut AUTOSCALE_CONTENT, il est possible de déterminer le comportement des éléments de commande d'un formulaire pour différentes résolutions d'écran. En standard, les éléments de commande sont automatiquement adaptés à la résolution d'écran détectée. Si l'on désire piloter soi-même le positionnement et le dimensionnement, il faut attribuer dans la balise form la valeur OFF à l'attribut.
	
	Attribut :
	autoscale_content
	
	Valeurs :
	• **on**
	Les coordonnées des éléments de commande sont automatiquement adaptées à la résolution d'écran (Standard)
	• **off**
	Les coordonnées des éléments de commande sont utilisées sans changement
	
	Exemple :
	`<form name="main_form" autoscale_content="off">`
	…
	…
	…
	`</form>`

Descripteur de balise	Signification
FORM Suite	Attributs : • **textfile** L'attribut permet la saisie d'un fichier texte lié à la forme. Le système utilise le descripteur EASY_XML comme contexte de texte par défaut. • **textcontext** L'attribut permet la saisie d'un descripteur pour le contexte de texte à utiliser. Cet attribut est valable en relation avec l'attribut **textfile**.

FORM Suite	L'attribut LAYOUT doit être utilisé pour appliquer des mises en page standard ou celles stockées dans le fichier easyscreen.ini. Sa valeur est le nom de la mise en page à utiliser.
	Remarque :
	Une modification des réglages par défaut n'est utile que pour les boîtes de dialogue contextuelles, puisque la boîte de dialogue appelée ici n'est pas masquée.
	
	Attribut :
	layout
	
	Syntaxe :
	`<form name="Nom du formulaire" layout="Nom de la mise en page" >`
	…
	…
	`</form>`
	
	Exemple :
	`<form name="form0" layout="slstandardscreenlayout.`
	`SlStandardScreenLayout.LowerForm" >`
	…
	…
	`</form>`
	
	Outre les positions des masques et les dimensions prédéfinies, il est également possible de stocker vos propres définitions dans le fichier easyscreen.ini.
	Entrée dans easyscreen.ini :
	`[640x480]`
	`MyPanel = x:=0, y:=220, width:=340, height:=174`
	`[800x480]`
	`MyPanel = x:=0, y:=220, width:=420, height:=174`
	
	Exemple :
	`<form name="form0" layout="MyPanel" >`
	…
	…
	`</form>`

Descripteur de balise	Signification
FORM Suite	**Messages de dialogue :** • **INIT** • **PAINT** • **TIMER** • **CLOSE** • **FOCUS_IN** • **INDEX_CHANGED** • **EDIT_CHANGED** • **GESTURE_EVENT** • **KEY_EVENT** • **MESSAGE** • **MOUSE_EVENT** • **RESIZE** • **CHANNEL_CHANGED** • **LANGUAGE_CHANGED**
FORM Suite	

Descripteur de balise	Signification

FORM Suite	**Syntaxe :**
	`<FORM name = "<dialog name>" color = "#ff0000">`
	Exemple :
	`<FORM name = "R-Parameter">`
	` <INIT>`
	` <DATA_ACCESS type = "true" />`
	` <CAPTION>R - Parameter</CAPTION>`
	` <CONTROL name = "edit1" xpos = "322" ypos = "34" refvar = "nck/Channel/Parameter/R[1]" />`
	` <CONTROL name = "edit2" xpos = "322" ypos = "54" refvar = "nck/Channel/Parameter/R[2]" />`
	` <CONTROL name = "edit3" xpos = "322" ypos = "74"`
	` </INIT>`
	
	`  <PAINT>`
	` <TEXT xpos = "23" ypos = "34">R - Parameter 1</TEXT>`
	` <TEXT xpos = "23" ypos = "54">R - Parameter 2</TEXT>`
	` <TEXT xpos = "23" ypos = "74">R - Parameter 3</TEXT>`
	
	` </PAINT>`
	
	`</FORM>`

INIT	**Message de boîte de dialogue** La balise est exécutée immédiatement après la création de la boîte de dialogue. Il convient de créer ici tous les éléments de saisie ainsi que les "hyperliens" du masque de dialogue.

Descripteur de balise	Signification

KEY_EVENT	Message de dialogue
	La balise KEY_EVENT peut être intégrée dans la forme pour l'évaluation des événements clavier. Si la balise est présente dans une forme, le système envoie le code clavier MF2 à la forme active. Si la variable **$actionresult** n'est pas mise à zéro, le système traite ensuite l'événement clavier.
	Le code clavier est mis à disposition dans la variable **$keycode** en tant que valeur Integer.
	
	Exemple :
	Le caractère saisi dans la variable **exclude_key** doit être filtré à partir du flux d'entrée.
	
	`<LET name="stream" type="string"/>`
	`<LET name="exclude_key" type="string"/>`
	
	`<FORM name = "keytest_form">`
	`  <INIT>`
	`  <CONTROL name = "p1" xpos = "120" ypos = "84" width ="200" refvar="stream" hotlink="true" />`
	`  <CONTROL name = "p2" xpos = "160" ypos = "104" width ="8" refvar="exclude_key" hotlink="true" />`
	`  </INIT>`
	
	`  <PAINT>`
	`  <text xpos = "8" ypos = "84">data stream</text>`
	`  <text xpos = "8" ypos = "104">exclude key</text>`
	`  </PAINT>`
	
	`  <KEY_EVENT>`
	`  <LET name="excl_keycode" type="string"/>`
	`  <OP>excl_keycode = exclude_key</OP>`
	`  <type_cast name="excl_keycode" type="int" />`
	`  <PRINT text="%d %d">$keycode, excl_keycode</PRINT>`
	
	`  <IF>`
	`  <CONDITION>$keycode == excl_keycode</CONDITION>`
	`  <THEN>`
	`  <op> $actionresult = 0</op>`
	`  </THEN>`
	`  </IF>`
	
	`  </KEY_EVENT>`
	
	`</FORM>`

Descripteur de balise	Signification

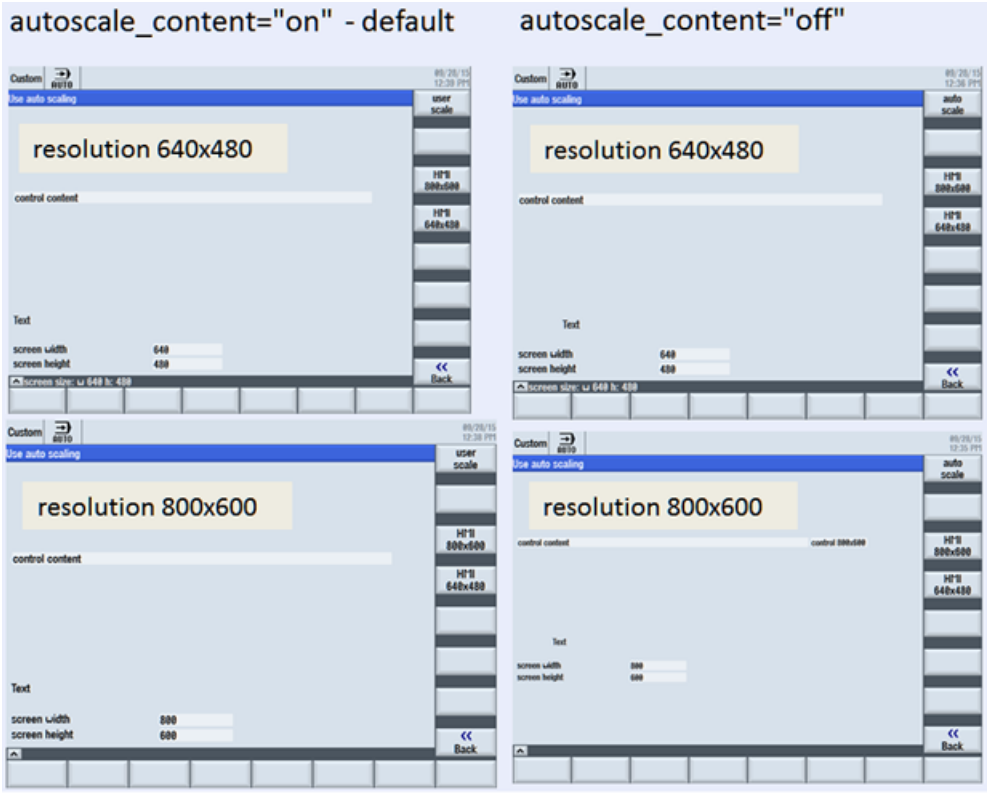
MOUSE_EVENT	La balise peut être intégrée dans le script pour le traitement des événements de souris. Celui-ci est exécuté lorsque les activités suivantes sont effectuées à l'aide de la souris :
	• Appui sur une touche
	• Relâchement d'une touche
	• Déplacement de la souris
	L'analyseur syntaxique met à disposition les informations dans une structure et crée la variable de structure **$mouse_event** avec les éléments suivants :
	
	Eléments de structure :
	• **type**
	Codage de l'activité
	– 2 - Pression d'un bouton
	– 3 - Relâchement d'un bouton
	– 5 - Déplacement de la souris
	• **x**
	Position X du curseur en pixels ; fait référence à la résolution actuelle de l'écran
	• **y**
	Position Y du curseur en pixels ; fait référence à la résolution actuelle de l'écran
	• **id**
	Descripteur
	– -1, s'il est impossible d'affecter la position à un élément de commande
	– != -1, si le curseur de la souris se trouve au sein d'un élément de commande, le contenu de l'attribut **idemdata** est fourni
	• **button**
	Comprend l'état des boutons au moment de l'événement
	– 0 - aucun bouton
	– 1 - bouton gauche
	– 2 - bouton droit
	– 4 - bouton central
	Les boutons peuvent être reliés à une opération OU bit par bit.
	
	Exemple :
	`<MOUSE_EVENT>`
	`  <print text="button %d type %d x %d y %d ">$mouse_event.button, $mouse_event.type, $mouse_event.x, $mouse_event.y</print>`
	`</MOUSE_EVENT>`

Descripteur de balise	Signification

MESSAGE	**Message de dialogue**
	Si l'instruction Send_message est exécutée dans le script, l'analyseur syntaxique exécute la balise Message. Les valeurs p1 et p2 sont mises à disposition dans les variables $message_par1 et $message_par2 (voir balise "SEND_MESSAGE").
	
	Syntaxe :
	`<MESSAGE>`
	`</MESSAGE>`
	
	Exemple :
	`<LET name="user_selection" />`
	
	`<SOFTKEY POSITION="3">`
	`<CAPTION>Set%nParameter</CAPTION>`
	`<SEND_MESSAGE>1, 10</SEND_MESSAGE>`
	`</SOFTKEY>`
	…
	…
	
	`<FORM>`
	…
	…
	`<MESSAGE>`
	`<SWITCH>`
	`<CONDITION>$message_par1</CONDITION>`
	`<CASE value="1">`
	`<OP> user_selection = $message_par2 </OP>`
	…
	…
	`</CASE>`
	`</SWITCH>`
	`</MESSAGE>`
	…
	…
	`</FORM>`

Descripteur de balise	Signification

SEND_MESSAGE	La balise envoie un message avec deux paramètres relatifs à la forme active traitée dans la balise Message (voir aussi MESSAGE).
	
	Syntaxe :
	`<SEND_MESSAGE>p1, p2</SEND_MESSAGE>`
	
	Exemple :
	`<LET name="user_selection" />`
	
	`<SOFTKEY POSITION="3">`
	`  <CAPTION>Set%nParameter</CAPTION>`
	`  <SEND_MESSAGE>1, 10</SEND_MESSAGE>`
	`</SOFTKEY>`
	…
	…
	`<FORM>`
	…
	…
	`<MESSAGE>`
	`<SWITCH>`
	`<CONDITION>$message_par1</CONDITION>`
	`<CASE value="1">`
	`<OP> user_selection = $message_par2 </OP>`
	…
	`</CASE>`
	`</SWITCH>`
	`</MESSAGE>`
	…
	…
	`</FORM>`

Descripteur de balise	Signification
RESIZE	**Message de boîte de dialogue** La balise peut être intégrée dans le script pour le traitement d'un événement RESIZE. Cet événement est créé par une commutation dynamique de la résolution. 

Descripteur de balise	Signification

RESIZE Suite	**Exemple :**
	`<let name="screen_size" type="StructSize" />`
	`<form name="menu_userscale_form" autoscale_content="off">`
	` <init>`
	` <caption>Use auto scaling</caption>`
	
	` <function name="hmi.get_hmi_resolution">screen_size</function>`
	` <print text="screen size: w %d h: %d">screen_size.width, screen_size.height</print>`
	
	` <data_access type="true" />`
	` <control name="s_c" xpos="8" ypos="140" fieldtype="readonly" width="500" refvar="string_var" hotlink="true"/>`
	` <if>`
	` <condition>screen_size.width == 800</condition>`
	` <then>`
	` <control name="s_1" xpos="508" ypos="140" fieldtype="readonly" width="100" refvar="string_var1" hotlink="true"/>`
	` </then>`
	` </if>`
	` <control name="s_w" xpos="200" ypos="350" fieldtype="readonly" refvar="screen_size.width" hotlink="true"/>`
	` <control name="s_h" xpos="200" ypos="370" fieldtype="readonly" refvar="screen_size.height" hotlink="true"/>`
	` </init>`
	` <paint>`
	` <text xpos ="68" ypos="310">Text</text>`
	` <text xpos ="8" ypos="350">screen width</text>`
	` <text xpos ="8" ypos="370">screen height</text>`
	` </paint>`
	
	` <resize>`
	` <function name="hmi.get_hmi_resolution">screen_size</function>`
	` <print text="resize_event screen size: w %d h: %d">screen_size.width, screen_size.height</print>`
	` <switch>`
	` <condition>screen_size.width</condition>`
	` <case value="640">`
	` <function name="control.delete">_T"s_1"</function>`
	` </case>`
	` <case value="800">`
	` <control name="s_1" xpos="508" ypos="140" fieldtype="readonly" width="100" refvar="string_var1" hotlink="true"/>`
	` </case>`
	` </switch>`
	` </resize>`
	`</form>`

Descripteur de balise	Signification

CHANNEL_CHANGED	**Message de dialogue**
	Cette balise permet le traitement de l'événement de changement de canal. Elle est exécutée lorsque l'utilisateur actionne la touche de présélection de canal.
	
	Syntaxe :
	`<channel_changed>`
	`</channel_changed>`
	
	Exemple :
	Lors d'un changement de canal, le formulaire doit être fermé et ouvert à nouveau afin de pouvoir afficher les données du canal sélectionné.
	`<channel_changed>`
	`  <open_form name = "form1" reopen="true"/>`
	`</channel_changed>`
	
	ou
	
	Lors d'un changement de canal, un autre menu est activé.
	`<channel_changed>`
	`  <navigation>menu2</navigation>`
	`</channel_changed>`

LANGUAGE_CHANGED	**Message de dialogue**
	Cette balise permet le traitement d'une demande de changement de langue. Elle est exécutée lorsque l'utilisateur change la langue dans un masque de dialogue ouvert.
	
	Syntaxe :
	`<language_changed>`
	`</language_changed>`
	
	Exemple :
	Lors d'un changement de langue, le formulaire doit être fermé et ouvert à nouveau afin de fournir aux éléments de commande les éléments de texte de la langue activée.
	`<language_changed>`
	`  <open_form name = "form1" reopen="true"/>`
	`</language_changed>`

FOCUS_IN	**Message de boîte de dialogue**
	La balise est appelée lorsque le système active un élément de commande. Pour l'identification de l'élément de commande, le système copie le nom de l'élément de commande dans la variable **$focus_name** et la valeur de l'attribut **item_data** dans la variable **$focus_item_data**.
	Ce message peut être utilisé par ex. pour pouvoir afficher des images en fonction de l'élément activé (curseur).
	Exemple :
	`<focus_in>`
	`<PRINT text="focus on filed:%s, %d">$focus_name,`
	`$focus_item_data </PRINT>`
	`</focus_in>`

Descripteur de balise	Signification
PAINT	**Message de boîte de dialogue** La balise est exécutée avec l'affichage de la boîte de dialogue. Il convient de spécifier ici tous les textes et images que la boîte de dialogue doit afficher. En outre, la balise est exécutée si le système constate que des parties de la boîte de dialogue doivent être affichées à nouveau. Cela peut être initié par ex. par la fermeture de fenêtres superposées.
TIMER	**Message de boîte de dialogue** La balise est exécutée de manière cyclique. Une temporisation qui déclenche l'exécution de la balise Timer toutes les 100 ms env. est affectée à chaque forme.

CAPTION	La balise contient le titre de la boîte de dialogue.
	Cette balise doit être utilisée à l'intérieur de la balise INIT.
	La ligne de titre peut être subdivisée en plusieurs colonnes.
	A cet effet, la balise caption doit être programmée avec l'attribut define_section avant l'édition du texte.
	L'attribut define_section définit le nombre de colonnes
	
	L'attribut property permet de définir pour chaque colonne la longueur, la position de départ et l'orientation du texte. L'indication de position et de longueur est une valeur en pourcentage qui se rapporte à la largeur de la forme.
	La valeur de l'attribut indexe le numéro de colonne (en commençant par zéro).
	
	`<caption define_sections="3">`
	`<property value="0" length="20" position="2" alignment="left" />`
	`<property value="1" length="20" position="79" alignment="right" />`
	`</caption>`
	
	Le texte peut ensuite être affecté à la colonne en spécifiant en outre l'attribut index avec l'indice de colonne.
	`<caption index="0">colonne1</caption>`
	`<caption index="1">colonne2</caption>`
	
	Syntaxe :
	`<CAPTION>Titel</CAPTION>`
	
	Exemple :
	`<CAPTION>my first dialogue</CAPTION>`

CLOSE	**Message de boîte de dialogue** Cette balise est exécutée avant la fermeture de la boîte de dialogue.

Descripteur de balise	Signification

CLOSE_FORM	La balise ferme la boîte de dialogue active.
	Cette instruction est seulement requise lorsque la boîte de dialogue a été ouverte par l'instruction MMC et qu'une fonction de touche logicielle est proposée à l'opérateur pour la fermeture de la boîte de dialogue. En principe les boîtes de dialogue sont gérées automatiquement et il n'est pas nécessaire de les fermer de manière explicite.
	
	Syntaxe :
	`<CLOSE_FORM/>`
	
	Exemple :
	`<softkey_ok>`
	` <caption>OK</caption>`
	` <CLOSE_FORM />`
	` <navigation>main_menu</navigation>`
	`</softkey_ok>`

Descripteur de balise	Signification
CONTROL	La balise sert à la création d'éléments de commande. **Syntaxe :** `<CONTROL name = "<control name>" xpos = "<Position X>" ypos = "<Position Y>" refvar = "<Variable CN>" hotlink = "true" format = "<Format>" />` **Remarque :** Les variables utilisées pour définir les valeurs d'attributs doivent être déclarées en tant que variables globales. **Attributs :** • **name** Descripteur du tableau. Le descripteur représente en même temps une variable locale et ne doit pas être utilisé plusieurs fois dans la forme. • **xpos** Position X du coin supérieur gauche • **ypos** Position Y du coin supérieur gauche • **fieldtype** Type de tableau Aucun type n'est spécifié, le tableau est configuré en tant que tableau d'édition (edit). – **edit** Les données peuvent être modifiées – **readonly** Les données ne peuvent pas être modifiées **combobox** À la place de valeurs numériques, le tableau présente les descripteurs correspondants. Si le type de tableau "**combobox**" est sélectionné, les expressions à représenter doivent également être affectées au tableau. Pour ce faire, il convient d'utiliser la balise <ITEM> . La zone de liste déroulante enregistre l'indice du texte actuellement sélectionné dans les variables inhérentes à l'élément de commande (voir attribut **refvar**). Après la création de l'élément de commande, il est possible d'insérer d'autres éléments à l'aide des fonctions **addItem** ou **insertItem**. – **progressbar** L'affichage d'une barre de progression s'effectue dans la plage de valeurs 0 à 100. Cette dernière peut être adaptée au paramètre à représenter à l'aide des propriétés Valeur minimale et Valeur maximale.

Descripteur de balise	Signification

CONTROL Suite	• **fieldtype edit** et **readonly**
	Dans le type de champ **edit** et **readonly**, un affichage de textes sur plusieurs lignes est possible avec l'attribut **multiline**. Le saut de ligne se produit à la limite d'un mot ou avec le caractère de saut de ligne <LF>.
	Exemple :
	`<control name="mulitline_edit" xpos="280" ypos="60" width="200" height="100" type="string">`
	`  <property multiline="true" />`
	`</control>`
	`<control name="c2" xpos="280" ypos="260" width="200" height="100" type="string" fieldtype="readonly">`
	`  <property multiline="true" />`
	`</control>`

Descripteur de balise	Signification

CONTROL Suite	• **fieldtype**
	– **graphicbox**
	Le type de tableau crée un élément de commande de graphique à traits en 2D. La balise <ITEM> permet d'insérer un élément graphique dans l'élément de commande. Les paramètres **width** et **height** indiquent la largeur et la hauteur de la boîte.
	Après la création de l'élément de commande, il est possible d'insérer d'autres éléments à l'aide des fonctions **addItem** ou **insertItem**. Le paramètre **itemdata** n'est pas exploité pour cet élément de commande.
	Si une variable de référence est affectée à l'élément de commande, l'enregistrement de ses valeurs a lieu dans une grille de 100 ms. Il est possible d'enregistrer au maximum 10000 valeurs. La propriété **max** provoque un enregistrement infini des valeurs. Une fois atteinte la durée maximale d'enregistrement, la valeur la plus ancienne est respectivement supprimée. La durée d'enregistrement doit être indiquée en millisecondes.
	L'élément de commande représente les valeurs enregistrées de manière discrète dans le temps sous forme de lignes reliées entre elles. L'attribut **channel** permet l'enregistrement de jusqu'à trois autres variables de référence. L'indice commence par un. L'indice nul (zéro) sert à définir les propriétés des variables affectées dans la balise de l'élément de commande.
	Exemple :
	`<control name= "c_gbox1" xpos = "250" ypos="24" width="240" height="356" fieldtype="graphicbox" refvar="val" hotlink="true" COLOR_BK="#ffffff">`
	`  <property max="60000" /> <!-- steps of ms -->`
	`  <property min="0" />`
	`  <property channel="0" color="#00ffff" penwidth="3"/>`
	`  <property ORDINATE="Y sinus" />`
	`  <property ABSCISSA="time *100 ms" />`
	`  <property channel="1" refvar="val2" color="#000000" />`
	`</control>`
	
	`<request name="nck/Channel/Parameter/R[2]" />`
	`<control name= "c_gbox" xpos = "0" ypos="5"  width="260" height="200" fieldtype="graphicbox" refvar="nck/Channel/Parameter/R[1]" hotlink="true" COLOR_BK="#ffffff">`
	`  <property max="400" />`
	`  <!-- steps of  ms  -->`
	`  <property min="0" />`
	`  <property channel="0" color="#00ff00" penwidth="1"/>`
	`  <property ORDINATE="Power" />`
	`  <property ABSCISSA="Time *100 ms" />`
	`  <property channel="1" refvar="nck/Channel/Parameter/R[2]" color="#FF0000" penwidth="1"/>`
	`  <property FIX_TIME_AREA="on" />`
	`</control>`

Descripteur de balise	Signification
CONTROL Suite	**Exemple graphicbox :** `<CONTROL name= "graphic" xpos = "8" ypos="23" width="300" height="352" fieldtype="graphicbox" />` • Ajout d'éléments : Les éléments sont ajoutés à l'aide de la fonction **additem** ou **loaditem**. Les éléments 2D suivants peuvent être utilisés : – Ligne - l(inc) – Secteur de cercle - c(ircle) – Point - p(oint) **Structure d'un élément :** <type d'élément>; coordonnées • Ligne : l; xs; ys; xe, ye I - Identification de ligne Xs - Position de départ X Ys - Position de départ Y Xe - Position de fin X Ye - Position de fin Y • Cercle : C, xs, ys, xe, ye, cc_x, cc_y, r C - Identification de secteur de cercle Xs - Position de départ X Ys - Position de départ Y Xe - Position de fin X Ye - Position de fin Y Cc_x - Coordonnées X du centre du cercle CC_y - Coordonnées Y du centre du cercle • Rayon : R • Point : P, x, y P - Identification de point X - Position X Y - Position Y • Suppression du graphique : Le contenu est supprimé à l'aide de la fonction **empty**.

Descripteur de balise	Signification
CONTROL Suite	Les types de champs suivants sont décrits au paragraphe "Configuration personnalisée des boutons (Page 215)". • **fieldtype** – **pushbutton** Le type de champ est utilisé comme touche ou commutateur. – **radiobutton** Le type de champ permet de sélectionner une option parmi plusieurs. – **checkbox** Le type de champ permet de sélectionner plusieurs options. – **groupbox** Le type de champ contient visuellement un groupe d'éléments de commande avec un cadre et un intitulé. – **switch** Le type de champ est un élément graphique indiquant un état sur deux par une icône. – **scrollarea** Le type de champ est utilisé pour afficher les éléments de commande au sein d'une zone donnée.

Descripteur de balise	Signification
CONTROL Suite	• **fieldtype** – **listbox** Le type de tableau génère une commande de menu déroulant vide. La balise <ITEM> permet d'insérer un élément dans le menu déroulant. L'attribut ITEM **value** permet d'affecter une valeur univoque à cet élément. Cela peut servir p. ex. à identifier l'élément. Les paramètres **width** et **height** indiquent la largeur et la hauteur du menu déroulant. Après la création de l'élément de commande, il est possible d'insérer d'autres éléments de menu déroulant à l'aide des fonctions **addItem**, **insertItem** ou **loadItem**. Après la création de l'élément de commande, il est possible d'insérer d'autres éléments de menu déroulant à l'aide de la fonction **addItem** ou **insertItem**. Le paramètre **item-data** n'est pas exploité pour cet élément de commande. – **itemlist** Le type de tableau génère un élément commande statique qui affiche les descripteurs correspondants à la place des valeurs numériques. La balise <ITEM> permet d'affecter un descripteur au tableau. • **item_data** Une valeur Integer spécifique à l'utilisateur peut être affectée à l'attribut. Cette valeur est fournie au message FOCUS_IN pour l'identification du tableau de l'élément activé. • **refvar** Descripteur des variables de référence, qui peut être combiné au tableau (facultatif). • **hotlink** = "TRUE" " Si la valeur des variables de référence change, le tableau est mis à jour automatiquement (facultatif). • **format** L'attribut définit le format d'affichage des variables spécifiées. Informations de formatage, voir print-Tag (facultatif). L'attribut **format** permet également la représentation d'une valeur entière dans d'autres systèmes de numération ou la représentation comme valeur booléenne pour les types de champs **edit** et **readonly**. Le formatage via le nombre de caractères et l'alignement n'est pas pris en charge. On peut utiliser les valeurs suivantes : • **%o** - octal • **%x** - hexadécimal • **%n** - binaire • **%b** - booléen

CONTROL Suite	**Exemple de création de colonnes, attribut property :**
	`<control name="lb1" xpos = "10" ypos = "94" width="200" height="250" fieldtype="listbox" refvar="tmp" hotlink="true">`
	`  <property columns="4" />`
	`  <property column="0" type="width">190</property>`
	`  <property column="1" type="width">100</property>`
	`  <property column="2" type="width">190</property>`
	`  <property column="3" type="width">16</property>`
	`</control>`

Descripteur de balise	Signification

CONTROL Suite	**Exemple de format d'attribut :**
	`<let name="val_test">255</let>`
	`<form name="main_form">`
	`<init>`
	`<caption>bool hex octal bin display</caption>`
	`<control name="test_oVal" xpos="250" ypos="60" refvar="val_test" hotlink = "true" />`
	`<control name="test_bool" xpos="250" ypos="90" refvar="val_test" hotlink = "true" format="%b"/>`
	`<control name="test_hex" xpos="250" ypos="120" refvar="val_test" hotlink = "true" format="%x"/>`
	`<control name="test" xpos="250" ypos="150" refvar="val_test" hotlink = "true" format="%o"/>`
	`<control name="test_b" xpos="200" ypos="180" width="300" refvar="val_test" hotlink = "true" format="%n"/>`
	`</init>`
	`<paint>`
	`<text xpos="16" ypos="60">integer value</text>`
	`<text xpos="16" ypos="90">boolean display</text>`
	`<text xpos="16" ypos="120">hexadecimal display</text>`
	`<text xpos="16" ypos="150">octal display</text>`
	`<text xpos="16" ypos="180">binary display</text>`
	`</paint>`
	`</form>`

CONTROL Suite	**Attributs :** • **color_bk** L'attribut définit la couleur d'arrière-plan de l'élément de commande. • **color_fg** L'attribut définit la couleur de premier plan de l'élément de commande. **Plus d'informations**, voir chapitre "Code couleur (Page 75)" • **display_format** L'attribut définit le format de traitement des variables spécifiées. Cet attribut doit être appliqué pour accéder à une variable Float de l'AP compte tenu que l'accès s'effectue au moyen de la lecture d'un mot double. Les formats de données admissibles sont les suivants : – FLOAT – INT – DOUBLE – STRING Affecter des expressions (par ex. texte ou élément graphique à afficher) d'un menu déroulant, d'une boîte graphique ou d'une zone de liste déroulante : **Syntaxe :** <ITEM>Expression</ITEM> <ITEM value ="<Valeur>">Expression</ITEM>

Descripteur de balise	Signification

CONTROL Suite	**Exemple :**
	`<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = " combobox ">`
	`<ITEM>text1</ITEM>`
	`<ITEM>text2</ITEM>`
	`<ITEM>text3</ITEM>`
	`<ITEM>text4</ITEM>`
	`</CONTROL>`
	Si une valeur Integer quelconque doit être affectée à une expression, l'attribut **value = "valeur"** doit être ajouté à la balise.
	A la place de la numérotation continue, la variable Control contient à présent la valeur affectée de l'élément.
	
	Exemple :
	`<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = " combobox ">`
	`<ITEM value = "10">text1</ITEM>`
	`<ITEM value = "20">text2</ITEM>`
	`<ITEM value = "12">text3</ITEM>`
	`<ITEM value = "1">text4</ITEM>`
	</CONTROL>
	Exemple de barre de progression :
	`<CONTROL name = "progress1" xpos = "10" ypos = "10" width = "100" fieldtype = "progressbar" hotlink = "true" refvar = "nck/Channel/GeometricAxis/actProgPos[1]">`
	`<PROPERTY min = "0" />`
	`<PROPERTY max = "1000" />`
	`</CONTROL>`
	Exemple de menu déroulant :
	`<let name="item_string" type="string"></let>`
	`<let name="item_data" ></let>`
	
	`<CONTROL name="listbox1" xpos = "360" ypos="150" width="200" height="200" fieldtype="listbox" />`
	• Ajout d'éléments :
	Les éléments sont ajoutés à l'aide de la fonction **additem** ou **loaditem**.
	• Suppression de contenu :
	Le contenu est supprimé à l'aide de la fonction **empty**.
	
	`<op> item_string = _T"text1\\n" </op>`
	`<function name="control.additem">_T"listbox1", item_string, item_data </function>`
	`<op> item_string = _T"text2\\n" </op>`
	`<function name="control.additem">_T"listbox1", item_string, item_data </function>`

Descripteur de balise	Signification

CONTROL Suite	**Exemple itemlist :**
	`<CONTROL name = "itemlist1" xpos = "10" ypos = "10" fieldtype = "itemlist">`
	`<ITEM value = "10">text1</ITEM>`
	`<ITEM value = "20">text2</ITEM>`
	`<ITEM value = "12">text3</ITEM>`
	`<ITEM value = "1">text4</ITEM>`
	`</CONTROL>`

CONTROL Suite	L'élément de commande Imagebox gère un graphique au format bitmap ou GIF. Si le graphique est plus grand que la zone représentée, l'élément de commande affiche des barres de défilement. Pour la commande de la zone visible, le système met à disposition la fonction CONTROL.IMAGEBOXSET. **Plus d'informations**, voir chapitre "Fonctions prédéfinies (Page 158)" **Syntaxe :** `<function name="control.imageboxget"> ... </function>`
CONTROL Suite	**Attributs :** • **disable** L'attribut bloque/permet la saisie dans un élément de commande edit. • **tooltip** Un texte de consigne est affiché lorsque le curseur est placé sur l'élément de commande. • **factor** Facteur de conversion • **font** Indication d'un type de police • **fontpixelsize** Hauteur de caractère - La valeur est à indiquer en nombre de pixels et fait référence à une résolution de 640x480 pixels. La hauteur des caractères affichés est déterminée à partir du rapport entre la résolution réelle et la résolution de référence. **Exemple :** `<control name="c2" xpos="180" ypos="120" fontname="arial" fontpixelsize="26" height="100" type="string" />`

Descripteur de balise	Signification
CONTROL Suite	**Attribut :** • **fontname** L'attribut permet de déterminer la police de caractères à utiliser. Les polices installées peuvent être déterminées par la fonction HMI.GET_FONT_FAMILIES. **Valeur :** Nom de la police Les polices de caractères Siemens suivantes sont installées dans le système : • Siemens AD CH Simplified • Siemens AD CH Traditional • Siemens AD JP • Siemens AD KS • Siemens AD Mono • Siemens AD Mono CH Simplified • Siemens AD Mono CH Traditional • Siemens AD Mono JP • Siemens AD Mono KS • Siemens AD Mono TH • Siemens AD Mono VN • Siemens AD Sans • Siemens AD TH • Siemens AD VN • Siemens Logo • Siemens Sans Cond Global • Siemens Sans Cond Mono • Siemens Sans Global Les systèmes basés sur MS Windows peuvent contenir des polices supplémentaires. **Exemple :** `<control name="c2" xpos="180" ypos="120" fontname="arial" fontpixelsize="26" height="100" type="string" />`

Descripteur de balise	Signification
CONTROL Suite	**Modifier un élément de commande après la création** Une balise d'élément de commande peut être utilisée pour modifier les propriétés d'un élément de commande existant après la création. La balise doit être spécifiée avec le nom de l'élément de commande à modifier et les nouvelles propriétés. Cela ne peut être effectué qu'à l'intérieur d'une balise form. Les propriétés suivantes peuvent par ex. être modifiées : • name • xpos • ypos • width • height • color_bk • color_fg • access level • fieldtype • itemdata • min • max • default • disable • tooltip • font • factor La variable de référence ne peut pas être modifiée. Si une propriété doit être modifiée par déclenchement au moyen d'un événement de touche logicielle, utiliser la balise send message pour transmettre cette demande dans le contexte form. La balise message est utilisée pour saisir le message.

Descripteur de balise	Signification

CONTROL Suite	**Modification de l'élément de commande dans une instruction d'opération**
	Pour modifier les propriétés d'un élément de commande pendant l'exécution, il est également possible de procéder à la modification d'une instruction d'opération. Pour cela, il convient d'indiquer le nom de l'élément de commande et la propriété à laquelle doit être affectée une nouvelle valeur. La propriété est séparée du nom de l'élément de commande par un point.
	Syntaxe :
	<Nom>.<Propriété>
	
	Exemple :
	`...`
	`...`
	`<let name="value" />`
	`<let name="w" />`
	`<let name="h" />`
	`...`
	`...`
	`<control name="c_move" xpos="$xpos" ypos="124" />`
	`...`
	`...`
	`<op>`
	`c_move.xpos = 300;`
	`value = c_move.xpos;`
	`h = c_move.height;`
	`w = c_move.width;`
	`</op>`

Descripteur de balise	Signification

HELP_CONTEXT	Cette balise définit la rubrique d'aide à appeler. Elle doit être programmée dans le bloc INIT.
	Système 808/802D sl
	Le nom indiqué dans l'attribut est complété par le préfixe XmlUserDlg_ et transmis au système d'aide. La structure ainsi constituée du fichier d'aide est décrite à la rubrique "Création de l'aide en ligne".
	Déroulement lors de l'activation du système d'aide :
	1. Activation de la touche "Info".
	2. La boîte de dialogue délivre l'expression "my_dlg_help".
	3. L'analyseur syntaxique convertit l'expression en "XmlUserDlg_my_dlg_help" .
	4. Activation du système d'aide.
	5. Transmission du terme recherché "XmlUserDlg_my_dlg_help".
	Système 840D sl/828D
	Plus d'informations sur la structure et la génération d'un manuel d'aide, voir :
	Manuel de mise en service SINUMERIK Operate
	
	Attributs :
	• **name**
	Le nom indiqué dans l'attribut est transmis au système d'aide. Le nom de fichier utilisé doit être indiqué dans le manuel d'aide dans la balise **entry**.
	• **anchor**
	L'attribut permet de spécifier le mot-clé.
	
	Syntaxe :
	`<HELP_CONTEXT name="<file name>" anchor="key word"/>`
	
	Exemple :
	`<form name = "form1">`
	`<init>`
	`<help_context name="chapter_1.html" anchor="Keyword_1" />`
	`...`
	`...`
	`<control name = "cpwd" xpos = "322" ypos = "64" >`
	`<property password="true" />`
	`<property help_context="chapter_1.html" anchor="Keyword_2" />`
	`</control>`
	`</form>`

Descripteur de balise	Signification
DATA_ACCESS	La balise gère le comportement des masques de dialogue lors de l'enregistrement des entrées utilisateur. Le comportement doit être défini à l'intérieur de la balise INIT. Si la balise n'est pas utilisée, l'enregistrement temporaire des entrées a toujours lieu. Exception : les éléments de commande pour lesquels l'attribut **hotlink** est défini sur **true** sont toujours écrits et lus directement. **Attribut :** • **type** = "TRUE" – aucun enregistrement temporaire des valeurs d'entrée n'a lieu. Le masque de dialogue copie les valeurs saisies directement dans les variables de référence. • **type =** "FALSE" – les valeurs sont copiées à l'aide de la balise **UPDATE_CONTROLS** type = "FALSE" dans la variable de référence. **Exemple :** `<DATA_ACCESS type = "true" />`

Descripteur de balise	Signification

DEBUG	Cette balise sert à enregistrer les traces qui peuvent être programmées dans le script. Utiliser l'attribut **text** pour décrire le texte à enregistrer. Cet attribut et ses valeurs correspondent à l'attribut **text** et aux valeurs de l'instruction d'impression.
	Par défaut, la balise de débogage n'est pas exécutée car elle ralentit le système. Pour activer les résultats de débogage, l'attribut **debug_msg** doit être programmé avec la valeur **on** dans la balise **DialogGui** ou **AGM**.
	Les traces sont enregistrées avec les messages d'erreur de l'analyseur dans le fichier suivant :
	card/oem/sinumerik/hmi/dvm/log/dvm.log
	
	Syntaxe :
	`<debug text="<Texte voir Instruction d'impression>">Variables voir Instruction d'impression</debug>`
	
	Exemple Easy XML :
	`<DialogGui debug_msg="on">`
	`...`
	`...`
	`<control name="test_bool" xpos="250" ypos="90" refvar="val_test" hotlink = "true" format="%b"/>`
	`<debug text="main_form control test_bool created value:%d">test_bool</debug>`
	
	Entrée dans le fichier dvm.log :
	`date: 10/22/ 2018 time: 09:38:44:351`
	`Debug xmldial.xml 17: main_form control test_bool created value:255`
	
	Exemple Easy Extend :
	`<AGM debug_msg="on">`
	`...`
	`...`
	`<control name="test_bool" xpos="250" ypos="90" refvar="val_test" hotlink = "true" format="%b"/>`
	`<debug text="main_form control test_bool created value:%d">test_bool</debug>`
	
	Entrée dans le fichier dvm.log :
	`date: 10/22/ 2018 time: 09:38:44:351`
	`Debug xmldial.xml 17: main_form control test_bool created value:255`

Descripteur de balise	Signification

EDIT_CHANGED	**Message de boîte de dialogue**
	La balise est appelée si le contenu d'un élément de commande edit a été modifié.
	Pour l'identification de l'élément de commande, le système copie le nom de l'élément de commande dans la variable **$focus_name** et la valeur de l'attribut **item_data** dans la variable **$focus_item_data**.
	
	Exemple :
	`<EDIT_CHANGED>`
	`<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>`
	…
	…
	`</EDIT_CHANGED>`

GESTURE_EVENT	La balise sert à exécuter des gestes pour la commande Multitouch. La balise est décrite au chapitre "Commande multitactile (Page 207)".

INDEX_CHANGED	**Message de boîte de dialogue**
	La balise est appelée si l'opérateur modifie la sélection d'une zone de liste déroulante.
	Pour l'identification de l'élément de commande, le système copie le nom de l'élément de commande dans la variable
	$focus_name et la valeur de l'attribut **item_data** dans la variable **$focus_item_data**.
	Remarque :
	Une variable de référence affectée à l'élément de commande n'a pas encore été comparée pour le moment à la variable de l'élément de commande et contient l'indice de la sélection précédente de la zone de liste déroulante.
	
	Exemple :
	`<INDEX_CHANGED>`
	`<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>`
	…
	…
	`</INDEX_CHANGED>`

MENU	La balise définit un menu qui contient la description des touches logicielles et la boîte de dialogue à ouvrir.
	
	Attribut :
	• **name**
	Nom de menu
	
	Syntaxe :
	`<MENU name = "<menu name>">`
	…
	`<open_form …>`
	…
	`<SOFTKEY …>`
	`</SOFTKEY>`
	`</MENU>`

Descripteur de balise	Signification

NAVIGATION	La balise définit le menu à appeler. Elle peut être utilisée au sein d'un bloc de touches logicielles, d'un bloc menu ou dans un formulaire. Si un nom de variable est affecté à la balise comme valeur, l'analyseur syntaxique active le menu enregistré dans les variables.
	Dans un bloc menu, la navigation s'effectue à la position correspondante à l'instruction. Les instructions suivantes ne sont plus exécutées.
	
	Remarque :
	Lorsque la cible de navigation doit être définie par le contenu d'une variable, cette dernière doit être déclarée en tant que variable globale.
	
	Syntaxe :
	`<NAVIGATION>menu name</NAVIGATION>`
	
	Exemple :
	`<menu name = "main">`
	`  <softkey POSITION="1">`
	`  <caption>sec. form</caption>`
	`  <navigation>sec_menu</navigation>`
	`  </softkey>`
	`</menu>`
	
	`<menu name = "sec_menu">`
	`  <open_form name = "sec_form" />`
	`  <softkey_back>`
	`  <navigation>main</navigation>`
	`  </softkey_back>`
	</menu>

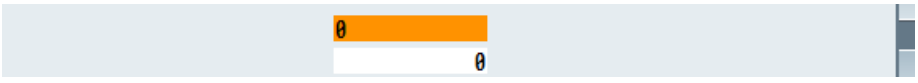
Descripteur de balise	Signification

OPEN_FORM	La balise ouvre le masque de dialogue indiqué par le nom.
	Si le masque de dialogue spécifié est déjà actif, cette balise ne sera pas exécutée.
	Une ouverture répétée du masque de dialogue peut être obtenue en spécifiant l'attribut **reopen**. Cela peut s'avérer nécessaire si des changements d'état du système nécessitent une modification du contenu du masque.
	
	Attribut :
	• **name**
	Nom du masque de dialogue
	• **reopen**
	Si la valeur de l'attribut est définie sur **true**, un appel répété de la balise **open_form** vérifie s'il s'agit du masque de dialogue actif. Si c'est le cas, l'analyseur ferme le formulaire actif et l'ouvre à nouveau.
	
	Syntaxe :
	`<OPEN_FORM name = "<form name>" />`
	
	Exemple :
	`<menu name = "main">`
	`  <open_form name = "main_form" />`
	`  <softkey POSITION="1">`
	`  <caption>main form</caption>`
	`  <navigation>main</navigation>`
	`  </softkey>`
	`</menu>`
	
	`<form name="main_form">`
	`  <init>`
	`  </init>`
	
	`  <paint>`
	`  </paint>`
	
	`</form>`

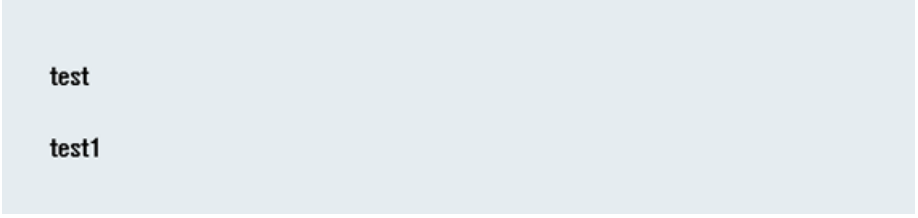
Descripteur de balise	Signification

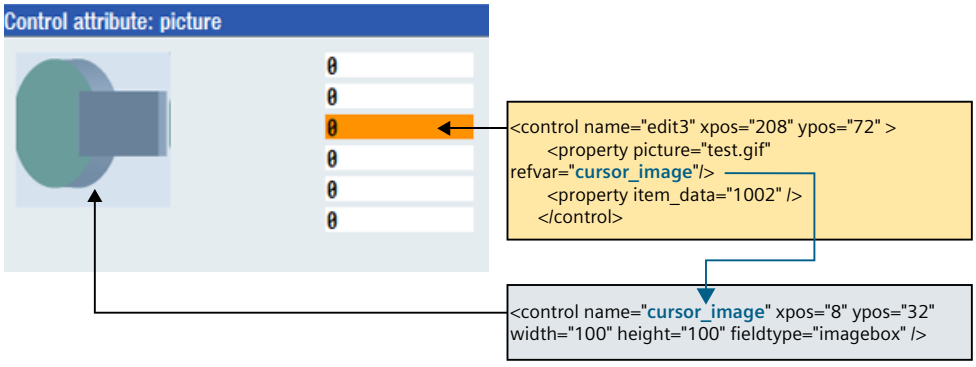
PROPERTY	Cette balise permet de définir des propriétés supplémentaires pour un élément de commande. La balise est insérée dans la balise control.
	
	Attributs :
	• **max** = "<valeur maximale>"
	• **min** = "<valeur minimale>"
	• **default** = "<valeur par défaut>"
	• **factor** = "facteur de conversion"
	• **color_bk** = "<code couleur arrière-plan>"
	• **color_fg** = "<code couleur police>"
	• **password** = "<true>" - les caractères saisis sont indiqués par "*"
	• **multiline** = "<true>" - permet la saisie sur plusieurs lignes dans une commande Edit
	• **disable** = "<true/false>" - bloque/permet la saisie dans un élément de commande Edit
	• **transparent** = "couleur transparente d'un bitmap"
	Plus d'informations, voir chapitre "Code couleur (Page 75)"
	• **tooltip** = le texte de consigne est affiché lorsque le curseur est placé sur l'élément de commande.
	La syntaxe est la suivante : <property tooltip="texte remarque" />
	• **abscissa** = "nom du premier axe de coordonnées" (uniquement pour la boîte graphique)
	• **ordinate** = "nom du deuxième axe de coordonnées" (uniquement pour la boîte graphique)
	• **fix_time_area** = "on" - Les attributs **min** et **max** définissent la période dans laquelle les signaux doivent être affichés. Les signaux représentés sont déplacés de droite à gauche.
	Exemple :
	`<CONTROL name = "progress1" xpos = "10" ypos = "10" width = "100" fieldtype = "progressbar" hotlink = "true" refvar = "nck/Channel/GeometricAxis/actProgPos[1]">`
	`<PROPERTY min = "0" />`
	`<PROPERTY max = "1000" />`
	`</CONTROL>`
	`<CONTROL name = "edit1" xpos = "10" ypos = "10">`
	`<PROPERTY min = "20" />`
	`<PROPERTY max = "40" />`
	`<PROPERTY default = "25" />`
	`</CONTROL>`

Descripteur de balise	Signification

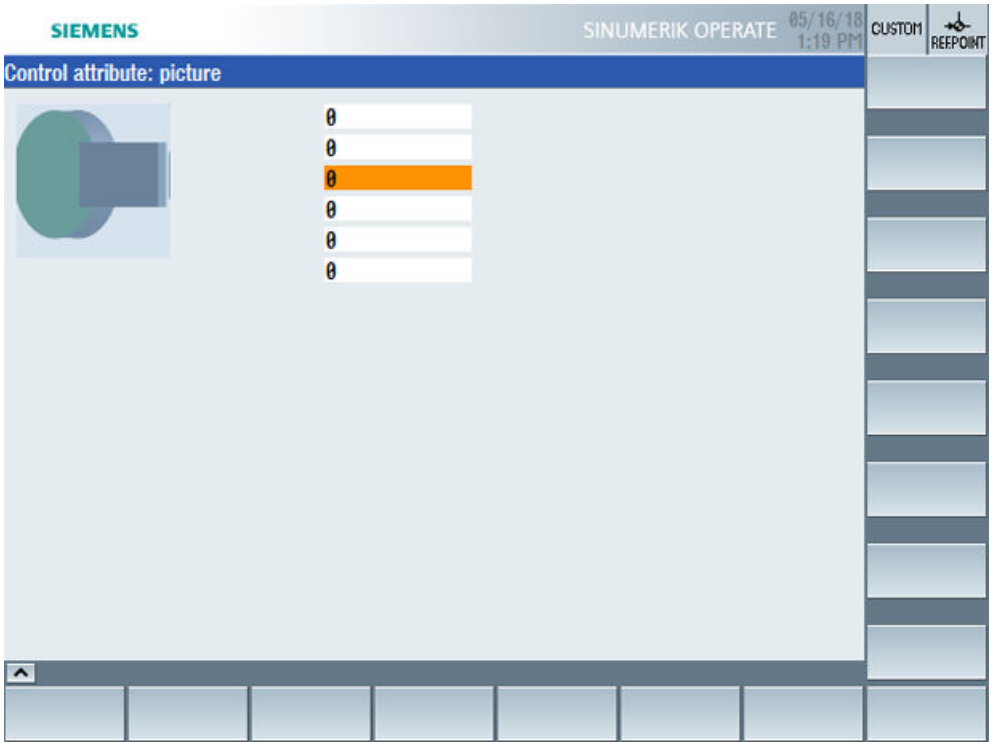
PROPERTY Suite	**Attribut :**
	alignment - La propriété permet un alignement du texte à gauche ou à droite. Par défaut, un texte est aligné à gauche.
	
	Valeurs :
	• **left**
	• **right**
	
	Syntaxe :
	`alignment="<right/left>"`
	
	Exemple :
	`<control name="edit2" xpos="208" ypos="52" >`
	`<property alignment="right"/>`
	`</control>`
	
	 

Descripteur de balise	Signification

PROPERTY Suite	**Attribut :**
	parent - La propriété définit l'affectation de l'élément de commande. Par défaut, les éléments de commande sont affectés à la forme. Si des éléments de commande doivent être affectés à une vue de défilement, cette dernière doit être spécifiée comme fenêtre parent.
	
	Valeur :
	Nom de la fenêtre parent
	
	Exemple :
	`...`
	`<control name="area" xpos="8" ypos="72" width="554" height="320" fieldtype="scrollarea" >`
	`<control name="test" xpos="20" ypos="40" width="20" fieldtype="readonly" />`
	`</control >`
	`<control name="test1" xpos="20" ypos="80" width="20" fieldtype="readonly" parent="area"/>`
	
	 
	
	`<init>`
	`<control name="area" xpos="8" ypos="72" width="554" height="120" fieldtype="scrollarea">`
	`<control name="test" xpos="20" ypos="40" width="80" fieldtype="readonly" type="string"/>`
	`</control >`
	
	`<control name="test1" xpos="20" ypos="80" width="80" fieldtype="readonly" type="string" parent="area"/>`
	`<op>test = _T"test"</op>`
	`<op>test1 = _T"test1"</op>`
	`<update_controls type="true" />`
	`...`

Descripteur de balise	Signification
PROPERTY Suite	**Attribut :** **picture** - L'attribut définit un graphique à afficher Le graphique indiqué s'affiche ou le nom du graphique est enregistré dans une variable, lorsque ce champ spécifié pour la saisie est activé. Cet attribut doit être utilisé avec l'attribut **refvar** pour pouvoir définir le puits de données. Pour l'affichage d'un graphique ou d'une animation, le nom d'un élément de commande de type **imagebox** doit être indiqué. Cet élément de commande se charge de la gestion des graphiques. Si le nom d'une variable locale est indiqué, cette variable doit être du type de données **String**. Le graphique reste affiché jusqu'à ce qu'un nouveau nom soit attribué à la variable de référence. **Syntaxe :** `<property picture="<Name der Grafik>" refvar="<Datensenke>"/>`
PROPERTY Suite	**Exemple de correspondance de champ :** 

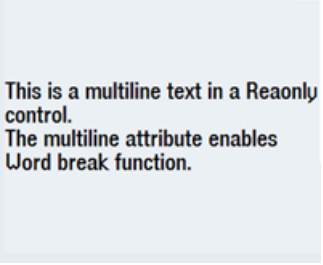
Descripteur de balise	Signification

PROPERTY Suite	**Exemple :**
	`<let name="cursor_icon" type="string" />`
	`<form name="main_form1">`
	`<init>`
	`<caption>Control attribute: picture</caption>`
	`<control name="cursor_image" xpos="8" ypos="32" width="100" height="100" fieldtype="imagebox" />`
	`<control name="edit1" xpos="208" ypos="32" >`
	`<property picture="red_led_off.bmp" refvar="cursor_image"/>`
	`</control>`
	`<control name="edit2" xpos="208" ypos="52" >`
	`<property picture="red_led_on.bmp" refvar="cursor_image"/>`
	`</control>`
	`<control name="edit3" xpos="208" ypos="72" >`
	`<property picture="test.gif" refvar="cursor_image"/>`
	`</control>`
	`<control name="edit4" xpos="208" ypos="92" >`
	`<property picture="red_led_off.bmp" refvar="cursor_image"/>`
	`</control>`
	`<control name="edit5" xpos="208" ypos="112" >`
	`<property picture="red_led_off.bmp" refvar="cursor_icon"/>`
	`</control>`
	`<control name="edit6" xpos="208" ypos="132" >`
	`<property picture="red_led_on.bmp" refvar="cursor_icon"/>`
	`</control>`
	`</init>`
	`<timer><print text="%s">cursor_icon</print></timer>`
	`</form>`
	 

Descripteur de balise	Signification

PROPERTY Suite	**Attribut :**
	cursortext - L'attribut définit le texte du curseur à afficher
	
	Le texte s'affiche dans la ligne de titre, lorsque ce champ spécifié pour la saisie est activé.
	En plus de cet attribut, il est possible d'indiquer deux autres attributs qui définissent l'alignement du texte et la zone de texte du curseur. Par défaut, le texte est aligné à gauche. La zone de texte du curseur prend par défaut 50 % de la largeur de la ligne de titre.
	
	`alignment="<right/left>"` - Alignement du texte à droite/à gauche
	`length="<Largeur>"` - Pourcentage que doit occuper le texte de curseur dans la ligne de titre
	
	Syntaxe :
	`<property cursortext="<text>" />`
	ou
	`<property cursortext="<text>" alignment="<right/left>" />`
	ou
	`<property cursortext="text2" alignment="r="<text>" alignment="<right/left>" length="<Rapport>"/>`
	
	Exemple :
	`<form name="main_form2">`
	`<init>`
	`<caption>Control attribute: cursortext</caption>`
	`<control name="edit1" xpos="208" ypos="32" >`
	`<property cursortext="cursor text field edit1" />`
	`</control>`
	`<control name="edit2" xpos="208" ypos="52" >`
	`<property cursortext="cursor text field edit2" alignment="right"/>`
	`</control>`
	`<control name="edit3" xpos="208" ypos="72" >`
	`<property cursortext="cursor text field edit3" alignment="right" length="40"/>`
	`</control>`
	`<control name="edit4" xpos="208" ypos="92" >`
	`<property cursortext="cursor text field edit4" />`
	`</control>`
	`</init>`
	`</form>`
	
	 

Descripteur de balise	Signification

PROPERTY Suite	**Attribut :**
	multiline - Permet l'affichage d'un texte sur plusieurs lignes dans un élément de commande Edit ou Readonly
	
	Syntaxe :
	`<property multiline="true" />`
	
	Exemple :
	`...`
	`...`
	`<op>`
	`textlist[2] = _T"This is a multiline text in a Reaonly control.\`
	`\nThe multiline attribute enables Word break function.";`
	`</op>`
	`<control name="lstring" xpos="8" ypos="120" width="200" height="160"`
	`fieldtype="readonly" refvar="textlist[2]" >`
	`<property multiline="true" />`
	`</control>`
	
	 

PROPERTY Suite	**Attributs :**
	• **help_context**
	Cet attribut permet d'affecter une rubrique d'aide à un élément de commande. Le nom de fichier utilisé doit être indiqué dans le manuel d'aide dans la balise **entry**.
	• **anchor**
	L'attribut permet de spécifier le mot-clé. Cet attribut n'est valable qu'avec l'attribut **help_context**.
	
	Syntaxe :
	`<property help_context="<file name>" anchor="<key word>"/>`
	
	Exemple :
	`<control name = "cpwd" xpos = "322" ypos = "64" >`
	`<property password="true" />`
	`<property help_context="chapter_1.html" anchor="Keyword_2" />`
	`</control>`

Descripteur de balise	Signification
SOFTKEY	La balise définit les propriétés et les réactions d'une touche logicielle. **Attributs :** • **position** Numéro de la touche logicielle. 1 à 8 touches logicielles horizontales, 9 à 16 touches logicielles verticales Les attributs suivants entrent en ligne de compte : • **type** Définit la propriété de la touche logicielle. user_controled - La représentation de la touche logicielle est définie par le script. toggle_softkey - La touche logicielle est représentée alternativement enfoncée ou non enfoncée. • **refvar** A utiliser uniquement en liaison avec le type toggle_softkey . Variable de référence dans laquelle la propriété actuelle de touche logicielle est copiée. Il s'agit de spécifier une variable du type "String", qui contient les propriétés "enfoncée", "non enfoncée" ou "bloquée" (voir balise state). • **picture** Un bitmap peut être édité à l'aide de l'attribut, justifié à gauche sur la touche logicielle. Le nom de chemin complet doit être indiqué. Le nombre de caractères de texte représentables se réduit de la largeur des bitmaps.

Descripteur de balise	Signification
SOFTKEY Suite	Les autres actions suivantes peuvent être définies à l'intérieur du bloc de touches logicielles : • **picturealignment** La justification de l'image est déterminée par cet attribut. Par défaut, l'image est justifiée sur la partie droite de la touche logicielle. Les valeurs suivantes peuvent être indiquées pour la justification : – **top** bord supérieur – **bottom** bord inférieur – **left** bord gauche – **right** bord droit – **center** centré • **caption** Texte des touches logicielles • **state** A utiliser uniquement en liaison avec le type user_controlled . La balise affecte la représentation souhaitée de la touche logicielle au système. **Syntaxe :** <state type="<state>" /> Les chaînes suivantes peuvent être spécifiées : – **notpressed** La touche logicielle est représentée non enfoncée. – **pressed** La touche logicielle est représentée enfoncée. – **disabled** La touche logicielle est représentée bloquée et grisée. • **navigation** • **update_controls** • **function**

Descripteur de balise	Signification
SOFTKEY Suite	**Syntaxe :** Touche logicielle standard : <state type="<softkey state>" /> <softkey position = "<1>"> … … </softkey> ou Touche logicielle commandée par script : <softkey position = "<1>" type="<user_defined>" > <state type="<softkey state>" /> … … </softkey> ou Touche logicielle de commutation : <softkey position = "<1>" type="<toggle_softkey>" refvar="<variable name>" > … … </softkey> **Exemple :** <let name="define_sk_type" type="string">PRESSED</let> <let name="sk_type">1</let> <softkey POSITION="1" type="user_controled" > <caption>Toggle%nSK</caption> <if> <condition>sk_type == 0 </condition> <then> <op> sk_type = 1 </op> <op> define_sk_type = _T"PRESSED" </op> </then> <else> <op> define_sk_type = _T"NOTPRESSED" </op> <op> sk_type = 0 </op> </else> </if> <state type="$$$define_sk_type" /> … … </softkey>

Descripteur de balise	Signification

SOFTKEY Suite	**Exemple :**
	ou
	
	`<let name="curr_softkey_state" type="string">PRESSED</let>`
	`</softkey>`
	
	`<softkey POSITION="3" type="toggle_softkey" refvar="curr_softkey_state">`
	`<caption>Toggle%nSK</caption>`
	...
	
	</softkey>
	
	 
	
	 

SOFTKEY_OK	La balise définit la réaction de la touche logicielle "OK".
	
	 
	
	Les autres actions suivantes peuvent être définies à l'intérieur du bloc de touches logicielles :
	• **navigation**
	• **update_controls**
	• **function**
	
	Syntaxe :
	`<SOFTKEY_OK>`
	...
	...
	`</SOFTKEY_OK>`

SOFTKEY_CANCEL	La balise définit la réaction de la touche logicielle "Interruption".
	
	 
	
	Les autres actions suivantes peuvent être définies à l'intérieur du bloc de touches logicielles :
	• **navigation**
	• **update_controls**
	• **function**
	
	Syntaxe :
	`<SOFTKEY_CANCEL>`
	...
	...
	`</SOFTKEY_CANCEL>`

Descripteur de balise	Signification

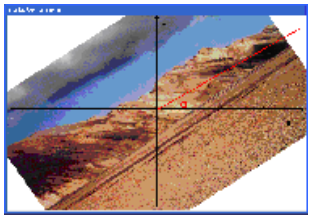
SOFTKEY_BACK	La balise définit la réaction de la touche logicielle "Retour".
	 
	Les autres actions suivantes peuvent être définies à l'intérieur du bloc de touches logicielles :
	• **navigation**
	• **update_controls**
	• **function**
	Syntaxe :
	`<SOFTKEY_BACK>`
	...
	...
	`</SOFTKEY_BACK>`

SOFTKEY_ACCEPT	La balise définit la réaction de la touche logicielle "Valider".
	 
	Les autres actions suivantes peuvent être définies à l'intérieur du bloc de touches logicielles :
	• **navigation**
	• **update_controls**
	• **function**
	Syntaxe :
	`<SOFTKEY_ACCEPT>`
	...
	...
	`</SOFTKEY_ACCEPT>`

Descripteur de balise	Signification
TEXT	La balise sert à afficher un texte à la position indiquée. Si un numéro d'alarme est utilisé, la boîte de dialogue indique le texte enregistré sous ce numéro. **Syntaxe :** `<TEXT xpos = "<Position X>" ypos = "<Position Y>"> Text </TEXT>` **Attributs :** • **xpos** Position X du coin supérieur gauche • **ypos** Position Y du coin supérieur gauche • **color** Couleur de texte **Plus d'informations**, voir chapitre "Code couleur (Page 75)" **Valeur :** Texte à afficher

IMG	La balise sert à afficher une image à la position indiquée. Les formats d'image BMP et PNG sont pris en charge.
	
	Syntaxe :
	`<IMG xpos = "<Position X>" ypos = "<Position Y>" name = "<name>" />`
	`IMG … xrot="Angle axe X" yrot="Angle axe Y" zrot="Angle axe Z" />`
	
	Attributs :
	• **xpos**
	Position X du coin supérieur gauche
	• **ypos**
	Position Y du coin supérieur gauche
	• **name**
	Nom de chemin complet. L'indication du chemin doit se faire en minuscules.
	• **transparent**
	Couleur transparente du bitmap
	Plus d'informations, voir chapitre "Code couleur (Page 75)"

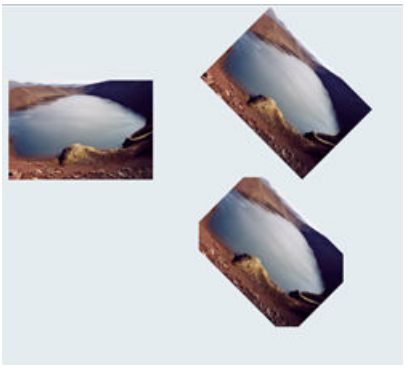
Descripteur de balise	Signification

IMG Suite	**Exemple** :
	L'image pivote de 34 degrés autour de l'axe Z :
	`<img xpos = "5" ypos = "23" name = "f:/appl/pic3.bmp" height="355" width="550" zrot="34" color_bk="#ffffff" ></img>`
	
	Remarque :
	La désignation des lecteurs varie selon le système.
	 
	
	Facultatif :
	Si la représentation de l'image diffère de la taille d'origine, la dimension peut être définie à l'aide des attributs **width** et **height**.
	• **width**
	Largeur en pixels
	• **height**
	Hauteur en pixels
	
	Exemples :
	`<IMG xpos = "20" ypos = "40" name = "f:/appl/test.bmp" />`
	`<IMG xpos = "5" ypos = "23" name = "f:/appl/test.bmp" height = "355" width = "550"/>`

Descripteur de balise	Signification

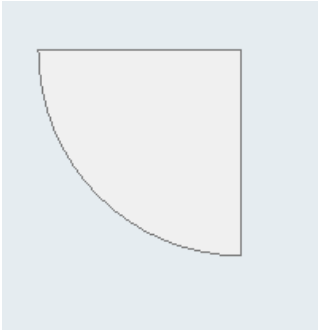
IMG Suite	**Attribut :**
	AspectRatioMode
	L'attribut permet de commander le rapport largeur/hauteur d'une image quand un zoom non proportionnel est effectué.
	
	Valeurs :
	• **Ignore**
	Le rapport largeur/hauteur du bitmap est adapté à la hauteur et à la largeur spécifiées.
	• **Keep** (par défaut)
	Le rapport largeur/hauteur est conservé et permet de garantir que l'image soit mise à l'échelle d'un rectangle occupant l'espace le plus étendu possible dans les limites de la hauteur et de la largeur spécifiées.
	• **KeepByExpanding**
	Le rapport largeur/hauteur est conservé et permet de garantir que l'image soit mise à l'échelle d'un rectangle occupant l'espace le plus réduit possible en dehors de la hauteur et de la largeur spécifiées.
	
	Exemple :
	`<img xpos = "8" ypos = "30" name = "test.bmp" height="155" width="150" AspectRatioMode="KeepbyExpanding">`
	`  <property transparent="#00ff00" />`
	`</img>`
	`<img xpos = "8" ypos = "200" name = "test.bmp" height="155" width="150" AspectRatioMode="Ignore">`
	`  <property transparent="#00ff00" />`
	`</img>`
	`<img xpos = "200" ypos = "200" name = "test.bmp" height="155" width="150" AspectRatioMode="Keep">`
	`  <property transparent="#00ff00" />`
	`</img>`
	 
	• En haut - Propriété **KeepbyExpanding** définie
	• En bas à gauche - Propriété **Ignore** définie
	• En bas à droite - Propriété **Keep** définie

Descripteur de balise	Signification

IMG Suite	**Attributs :**
	ExpandingForRotation
	Le rapport largeur/hauteur est conservé. L'image est mise à l'échelle d'un rectangle qui correspond au cadre de l'image tournée et mise à l'échelle.
	
	Exemple :
	`<img xpos = "8" ypos = "100" name = "pic1.bmp" height="155" width="150" >`
	`  <property transparent="#ffffff" />`
	`</img>`
	`<op>`
	`  zrot = 45.0;`
	`</op>`
	`<img xpos = "205" ypos = "23" name = "pic1.bmp" height="155" width="150" zrot="$zrot" AspectRatioMode="ExpandingForRotation">`
	`  <property transparent="#ffffff" />`
	`</img>`
	`<img xpos = "205" ypos = "200" name = "pic1.bmp" height="155" width="150" zrot="$zrot" >`
	`  <property transparent="#ffffff" />`
	`</img>`
	 
	• À gauche - Image mise à une échelle de 150 x 155 pixels
	• En haut à droite - Image mise à une échelle de 150 x 155 pixels et tournée de 45 degrés avec la propriété **ExpandingForRotation**
	• En bas à droite - Image mise à une échelle de 150 x 155 pixels et tournée de 45 degrés

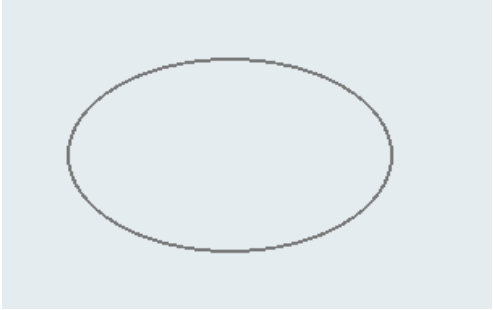
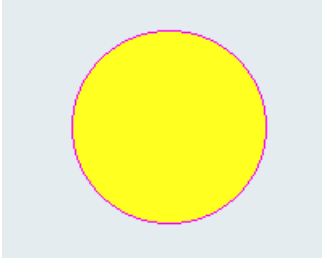
Descripteur de balise	Signification
ARC	La balise sert à dessiner un secteur de cercle dans une forme. **Attributs :** Position à l'intérieur de la forme en pixels : • **xpos1** - Point de départ du segment de cercle coordonnées X • **ypos1** - Point de départ du segment de cercle coordonnées Y • **xpos2** - Point final du segment de cercle coordonnées X • **ypos2** - Point final du segment de cercle coordonnées Y Position à l'intérieur de la forme en pixels : • **ccx** - Centre du segment de cercle coordonnées X • **ccy** - Centre du segment de cercle coordonnées Y • **radius** - Rayon du cercle, valeur en pixels **Remarque :** **Centre ou rayon** En spécifiant les deux paramètres, le centre du cercle est utilisé.
ARC Suite	• **penwidth** L'attribut définit l'épaisseur du trait de la ligne en pixels. • **color** Couleur de trait • **color_bk** Couleur de remplissage du segment de cercle **Plus d'informations**, voir chapitre "Code couleur (Page 75)" • **style** L'attribut définit le type de ligne : – **"SolidLine"** - Ligne continue (par défaut) – **"DashLine"** - Ligne tiretée – **"DotLine"** - Ligne pointillée – **"DashDotLine"** - Ligne points-tirets – **"DashDotDotLine"** - Ligne tiret-point-point

Descripteur de balise	Signification
ARC Suite	• **type** – **cw** - Dans le sens horaire – **ccw** - Dans le sens anti-horaire • **arrow** Des flèches sont représentées à la fin des segments : – **"P1"** - Flèche au point de départ de la ligne – **"P2"** - Flèche au point final de la ligne – **"P1P2"** - Flèches au point de départ et au point final de la ligne – **"P1_FILLED"** - Flèche au point de départ de la ligne remplie – **"P2_FILLED"** - Flèche au point final de la ligne remplie – **"P1P2_FILLED"** - Flèches au point de départ et au point final de la ligne remplies • **arrow_size** Si l'attribut Arrow est utilisé, la longueur de la flèche peut être indiquée en pixels.

ARC Suite	**Exemples :**
	 
	`<arc xpos1="20" ypos1="200" xpos2="180" ypos2="200" ccx="100" ccy="200" type="ccw" PENWIDTH="1" color="#777777" arrow="p1p2_filled"/>`
	
	ou
	
	`<arc xpos1="20" ypos1="200" xpos2="180" ypos2="200" radius="80" type="ccw" PENWIDTH="1" color="#777777" arrow="p1p2_filled"/>`
	
	 
	`<arc xpos1="20" ypos1="200" xpos2="100" ypos2="280" radius="80" type="ccw" PENWIDTH="1" color="#777777" color_bk="#f0f0f0"/>`

Descripteur de balise	Signification
BOX	La balise dessine un angle droit plein à la position indiquée et dans la couleur spécifiée. **Syntaxe :** `<BOX xpos = "<Position X>" ypos = "<Position Y>" width = "<Extension X>" height = "<Extension Y>" color = "<Code couleur>" />` **Attributs :** • **xpos** Position X du coin supérieur gauche • **ypos** Position Y du coin supérieur gauche • **width** Extension dans la direction X (en pixels) • **height** Extension dans la direction Y (en pixels) • **color** Code couleur **Plus d'informations**, voir chapitre "Code couleur (Page 75)"
ELLIPSE	La balise sert à dessiner une ellipse dans une forme. **Attributs :** Position à l'intérieur de la forme en pixels : • **xpos** - Point de départ de la ligne coordonnées X • **ypos** - Point de départ de la ligne coordonnées Y • **width** - Largeur du rectangle englobant (bounding box) • **height** - Hauteur du rectangle englobant (bounding box) • **penwidth** L'attribut définit l'épaisseur du trait de la ligne en pixels. • **color** Couleur de trait • **color_bk** Couleur de remplissage de l'ellipse **Plus d'informations**, voir chapitre "Code couleur (Page 75)" • **style** L'attribut définit le type de ligne : – **"SolidLine"** - Ligne continue (par défaut) – **"DashLine"** - Ligne tiretée – **"DotLine"** - Ligne pointillée – **"DashDotLine"** - Ligne points-tirets – **"DashDotDotLine"** - Ligne tiret-point-point

Descripteur de balise	Signification

ELLIPSE Suite	**Exemples :**
	 
	`<ellipse xpos="302" ypos="160" width="100" height="60" PENWIDTH="2" color="#777777" />`
	 
	`<ellipse xpos="402" ypos="360" width="60" height="60" PENWIDTH="1" color="#f800ff"  color_bk="#ffff20"/>`

Descripteur de balise	Signification

FUNCTION	**Appel de fonction**
	La balise exécute le corps de fonction spécifié
	sous l'attribut "**name**".
	
	Attributs :
	• **name** = "Nom du corps de fonction"
	• **return** = "Nom de variable pour l'enregistrement du résultat de la fonction"
	
	Valeurs :
	Liste des variables qui doivent être transmises au corps de fonction. Les variables doivent être séparées les unes des autres par une virgule. Il est possible de transférer au maximum 10 paramètres.
	En outre, il est possible de spécifier des constantes ou des expressions de texte en tant que paramètres d'appel. Pour caractériser une expression de texte, il convient de faire précéder le texte de l'identifiant _T.
	
	Syntaxe :
	`<FUNCTION name = "<function name>" />`
	La fonction appelante attend une valeur de retour
	`<FUNCTION name = "<function name>" return = "<Variablenname>" />`
	Transfert de paramètres
	`<FUNCTION name = "<function name>"> var1, var2, var3 </FUNCTION>`
	`<FUNCTION name = "<function name>"> _T"Text", 1.0, 1 </FUNCTION>`
	
	Exemples :
	Voir "FUNCTION_BODY"

Descripteur de balise	Signification

FUNCTION_BODY	**Corps de fonction**
	La balise contient le corps de fonction d'une sous-fonction. Le corps de fonction doit être programmé à l'intérieur de la balise DialogGui.
	
	Attributs :
	• **name** = "Nom du corps de fonction"
	• **parameter** = "Liste de paramètres" (facultatif)
	L'attribut liste les paramètres de transfert requis. Les paramètres doivent être séparés les uns des autres par une virgule.
	Lors de l'appel du corps de fonction, les valeurs des paramètres spécifiés dans l'appel de fonction sont copiées dans les paramètres de transfert répertoriés.
	• **return** = "true" (facultatif)
	Si l'attribut est défini sur true, la variable locale $return est créée. Il convient de copier dans cette variable la valeur de retour de la fonction transmise à la fonction appelante en quittant la fonction.
	
	Syntaxe :
	
	Corps de fonction sans paramètres
	`<FUNCTION_BODY name = "<function name>">`
	…
	…
	…
	`</ FUNCTION_BODY>`
	
	Corps de fonction avec paramètres
	`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>">`
	…
	`<LET name = "tmp></LET>`
	`<OP> tmp = p1 </OP>`
	…
	`</FUNCTION_BODY>`
	
	Corps de fonction avec valeur de retour
	`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>" return = "true">`
	…
	`<LET name = "tmp></LET>`
	`<OP> tmp = p1 </OP>`
	…
	`<OP> $return = tmp </OP>`
	`</FUNCTION_BODY>`

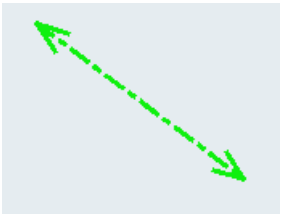
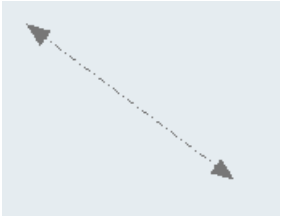
Descripteur de balise	Signification

FUNCTION_BODY Suite	**Exemple :**
	`<function_body name = "test" parameter = "c1,c2,c3" return = "true">`
	`<LET name = "tmp">0</LET>`
	`<OP> tmp = c1+c2+c3 </OP>`
	`<OP> $return = tmp </OP>`
	`</function_body>`
	…
	…
	…
	`<LET name = "my_var"> 4 </LET>`
	`<function name = "test" return = " my_var "> 2, 3,4</function>`
	`<print text = "result = %d"> my_var </print>`
	…
	…

LINE	La balise sert à dessiner une ligne dans une forme. **Attributs :** Position à l'intérieur de la forme en pixels : • **xpos1 -** Point de départ de la ligne coordonnées X • **ypos1** - Point de départ de la ligne coordonnées Y • **xpos2** - Point final de la ligne coordonnées X • **ypos2** - Point final de la ligne coordonnées Y • **penwidth** L'attribut définit l'épaisseur du trait de la ligne en pixels. • **color** Couleur de trait **Plus d'informations**, voir chapitre "Code couleur (Page 75)" • **style** L'attribut définit le type de ligne : – **"SolidLine"** - Ligne continue (par défaut) – **"DashLine"** - Ligne tiretée – **"DotLine"** - Ligne pointillée – **"DashDotLine"** - Ligne points-tirets – **"DashDotDotLine"** - Ligne tiret-point-point

Descripteur de balise	Signification
LINE Suite	• **arrow** Des flèches sont représentées aux fins de ligne : – **"P1"** - Flèche au point de départ de la ligne – **"P2"** - Flèche au point final de la ligne – **"P1P2"** - Flèches au point de départ et au point final de la ligne – **"P1_FILLED"** - Flèche au point de départ de la ligne remplie – **"P2_FILLED"** - Flèche au point final de la ligne remplie – **"P1P2_FILLED"** - Flèches au point de départ et au point final de la ligne remplies • **arrow_size** Si l'attribut Arrow est utilisé, la longueur de la flèche peut être indiquée en pixels. **Syntaxe :** `<line xpos1="<Point de départ coordonnées X>" ypos1="<Point de départ coordonnées Y>" xpos2="<Point final coordonnées X>" ypos2="<Point final coordonnées Y>" PENWIDTH="<Épaisseur de trait>" color="<Couleur de trait>" style="<Style de trait>" arrow="<Type de flèche>" arrow_size="<Taille de flèche>"/>`

Descripteur de balise	Signification

LINE Suite	**Exemples :**
	`<line xpos1="20" ypos1="40" xpos2="100" ypos2="100" PENWIDTH="3" color="#0ff00f" style="DashDotLine" arrow="p1p2" arrow_size="12"/>`
	 
	`<line xpos1="20" ypos1="40" xpos2="100" ypos2="100" PENWIDTH="1" color="#777777" style="DashDotLine" arrow="p1p2_filled" arrow_size="9"/>`
	 
	`<line xpos1="20" ypos1="40" xpos2="100" ypos2="100" PENWIDTH="2" color="#777777" arrow="p2_filled" arrow_size="9"/>`
	 

Descripteur de balise	Signification

REQUEST	La balise permet de reprendre une variable dans la fonction cyclique de lecture (hyperlien). Le temps d'accès aux variables non liées à un élément de commande s'en trouve réduit.
	Si une fonction doit être appelée automatiquement en cas de changement de valeur, le nom de la fonction doit être indiqué comme attribut complémentaire.
	Cette balise est exclusivement traitée à l'intérieur de l'instruction INIT.
	
	Attributs :
	• **name**
	Descripteur d'adresse
	• **function**
	Nom de la fonction
	
	Syntaxe :
	`<REQUEST name = "<NC-Variable>" />`
	ou
	`<REQUEST name = "<NC-Variable>" function="<function name>"/>`
	
	Exemple :
	`<request name ="plc/mb10" />`
	
	ou
	
	`<function_body name="my_function" >`
	`<print text="value changed" />`
	`</function_body>`
	…
	…
	…
	`<request name ="plc/mb10" function="my_function"/>`

Descripteur de balise	Signification

RECALL	Cette balise peut être utilisée dans un menu si une navigation doit être réalisée via la touche Recall. Si la balise est programmée, le symbole Recall est affiché et le traitement de la touche Recall est autorisé par l'analyseur syntaxique.
	
	Syntaxe :
	`<recall>`
	...
	...
	`</recall>`

UPDATE_CONTROLS	La balise effectue une comparaison entre les éléments de commande et les variables de référence.
	
	Attribut :
	• **type**
	L'attribut détermine le sens de la comparaison de données.
	= TRUE – Les données sont lues à partir des variables de référence et copiées dans les éléments de commande.
	= FALSE – Les données sont lues à partir des éléments de commande et copiées dans les variables de référence.
	
	Syntaxe :
	`<UPDATE_CONTROLS type = "<Sens>"/>`
	
	Exemple :
	`<SOFTKEY_OK>`
	`< UPDATE_CONTROLS type="false"/>`
	`</SOFTKEY_OK>`

3.8 Création de menus utilisateur

3.8.1 Création de masques de cycle de traitement

La fonction Prise en charge de cycles permet la création automatique et la décompilation d'un appel de cycle au moyen de la boîte de dialogue des formes.

Les balises suivantes sont disponibles pour la manipulation de cette fonctionnalité :

- **NC_INSTRUCTION**
- **CREATE_CYCLE**

Pour identifier un masque de cycle, il convient de spécifier dans la balise **FORM** l'attribut **type** avec la valeur **cycle**. Cette identification permet le traitement de l'instruction **NC_INSTRUCTION**.

Exemple

```
<FORM name = "cycle100_form" type= "CYCLE">
…
…
</FORM>
```

La balise **NC_INSTRUCTION** contient l'appel de cycle à générer. Tous les paramètres de cycle doivent être réservés au moyen de variables génériques.

Exemple

```
<FORM name = "cycle100_form" type= "CYCLE">
<NC_INSTRUCTION refvar= "cyc_string" >Cycle100 ($p1, $p2, $p3)</
NC_INSTRUCTION>
…
…
…
</FORM
```

La balise **CREATE_CYCLE** prépare les valeurs enregistrées dans les variables génériques et génère l'instruction CN.

Celle-ci est ensuite copiée dans la variable spécifiée.

Descripteur de balise	Signification

NC_INSTRUCTION	L'instruction CN à créer est définie avec cette balise.
	Tous les paramètres de cycle représentés sont créés automatiquement en tant que variables String de **FORM** et sont à la disposition de **FORM**.
	
	Condition : L'attribut FORM **type** est défini sur la valeur **CYCLE**.
	
	Attribut :
	• **refvar**
	Si une variable de référence est affectée à la balise, tous les paramètres se voient affecter les valeurs issues du bloc CN enregistré dans la variable de référence.
	Syntaxe :
	`<NC_INSTRUCTION> Instruction CN avec marques de réservation </ NC_INSTRUCTION>`
	
	Exemple :
	`<let name="cyc_string" type="string"> Cycle100(0, 1000, 5)</ let>`
	…
	…
	…
	`<FORM name = "cycle100_form" type= "CYCLE">`
	`<NC_INSTRUCTION refvar= "cyc_string" >Cycle100($p1, $p2, $p3)</ NC_INSTRUCTION>`
	…
	…
	…
	`</FORM>`

Descripteur de balise	Signification

CREATE_CYCLE	La balise génère un bloc de CN dont la syntaxe est définie par la valeur de la balise **NC_INSTRUCTION** .
	Avant de générer l'instruction CN, l'analyseur syntaxique appelle la balise CYCLE_CREATE_EVENT de FORM. Cette balise peut être utilisée pour calculer les paramètres de cycle.
	Syntaxe :
	`<CREATE_CYCLE/>`
	
	Option :
	Si une variable de référence est fournie, l'instruction copie l'appel produit dans cette variable.
	`<create_cycle refvar="name" />`
	
	Attribut :
	• **refvar**
	Si une variable de référence est affectée à la balise, l'instruction CN est copiée dans cette variable.
	
	Exemple :
	`<LET name="cyc_string" type="string"> Cycle100(0, 1000, 5)</LET>`
	
	`<SOFTKEY_OK>`
	`<caption>OK</caption>`
	`<CREATE_CYCLE />`
	`<close_form />`
	`<navigation>main_menu</navigation>`
	`</SOFTKEY_OK>`
	
	ou
	
	`<SOFTKEY_OK>`
	`<caption>OK</caption>`
	`<CREATE_CYCLE refvar= "cyc_string" />`
	`<close_form />`
	`<navigation>main_menu</navigation>`
	`</SOFTKEY_OK>`

3.8.2 Caractères de remplacement

Le système permet de définir des propriétés d'un élément de commande (valeurs d'attribut) pendant l'exécution. Pour pouvoir utiliser cette fonction, il convient de mettre à disposition la propriété souhaitée dans une variable locale et de transmettre à la balise le nom de variable en le faisant précéder d'un **caractère $** comme valeur d'attribut.

Si la balise attend une variable String en tant que valeur d'attribut ou valeur, les caractères $$$ doivent être placés avant le nom de variable.

Exemple :

```
<let name="my_ypos">100</let>
<let name="field_name" type="string"></let>

<control name = "edit1" xpos = "322" ypos = "$my_ypos" refvar="nck/
Channel/Parameter/R[1]" />

<op>my_ypos = my_ypos +20 </op>

<control name = "edit2" xpos = "322" ypos = "$my_ypos" refvar="nck/
Channel/Parameter/R[2]" />

<print name ="field_name" text="edit%d">3</print>
<op>my_ypos = my_ypos +20 </op>

<control name = "$field_name" xpos = "322" ypos = "$my_ypos"
refvar="nck/Channel/Parameter/R3]" />

<caption>$$$field_name</caption>
```

3.9 Créer un fichier struct_def.xml

Procédure

Dans certaines fonctions, le fichier **struct_def.xml** est utilisé pour la définition des types. Si ce fichier XML est absent, créer une copie du fichier avec le balisage suivant et l'intégrer dans le projet fonctionnel.

Remarque

Créer le fichier XML au format UTF8.

Script

struct_def.xml

```
<!--
Copyright 2015 by Siemens AG and Wolfram Kuhnert
(wolfram.kuhnert@siemens.com)
Permission to use, copy, modify, distribute, and sell this softwareand its
documentation for any purpose is hereby granted without fee,provided that
the above copyright notice appear in all copies and that both that copyright
notice and this permission notice appearin supporting documentation, and
that the names of Siemens AG and Wolfram Kuhnertnot be used in advertising
or publicity pertaining todistribution of the software without specific,
written priorpermission.
Siemens AG and Wolfram Kuhnert make no representations about the
suitability of this software for any purpose. It isprovided "as is" without
express or implied warranty.
Required software version: Operate 4.7 Sp1 HF1
-->

<!--
    typedef struct tagRECT {
     LONG left;
     LONG top;
     LONG right;
     LONG bottom;
    } RECT;
-->

<typedef name="StructRect" type="struct" >
  <element name="left" type="int">0</element>
  <element name="top" type="int">0</element>
  <element name="right" type="int">0</element>
  <element name="bottom" type="int">0</element>
</typedef>

<typedef name="StructPoint" type="struct" >
  <element name="x" type="int">0</element>
  <element name="y" type="int">0</element>
</typedef>

<typedef name="StructSize" type="struct" >
  <element name="width" type="int">0</element>
  <element name="height" type="int">0</element>
</typedef>

<typedef name="StructMDLimits" type="struct" >
  <element name="lowerLimit" type="double">0</element>
  <element name="upperLimit" type="double">0</element>
  <element name="arrayDim1" >0</element>
  <element name="arrayDim2" >0</element>
</typedef>
```

3.10 Adressage de composants

Pour l'adressage de variables CN, de blocs d'AP ou de paramètres d'entraînement, des descripteurs d'adresse doivent être créés pour le paramètre souhaité. Une adresse est constituée des chemins partiels **Nom de composant** et **Adresse de variable**. Une barre oblique doit être utilisée comme caractère de séparation.

3.10.1 Adressage de l'AP

L'adressage de l'AP commence par la composante de chemin **plc**.

Tableau 3-3 Les adresses suivantes sont admissibles :

DBx.DB(f)	Bloc de données
I(f)x	Entrée
Q(f)x	Sortie
M(f)x	Mémento
V(f)x	Variable

DBx.DBXx.b	Bloc de données
Ix.b	Entrée
Qx.b	Sortie
Mx.b	Mémento
Vx.b	Variable

Tableau 3-4 Format de données **f** :

B	Octet
W	Mot
D	Double mot

Pour un adressage de bits, l'identification du format de données n'est pas nécessaire.

Adresse **x** :

Descripteur d'adresse S7 200 valide

Adressage de bits :

b – Numéro de bit

Exemples :

```
<data name = "plc/mb170">1</data>
<data name = "i0.1"> 1 </data>
<op> "m19.2" = 1 </op>
refvar="plc/db9062.dbb0:string"
```

3.10.2 Adressage de variables CN

L'adressage de variables CN commence par la composante de chemin **nck**,

Cette partie est suivie de l'adresse de la donnée dont la structure est décrite dans le Manuel de listes Variables CN et signaux d'interface.

Exemple :

```
<LET name = "tempStatus"></LET>
<OP> tempStatus ="nck/channel/state/chanstatus" </OP>
```

3.10.3 Adressage propre au canal

Si aucun numéro de canal n'est indiqué dans le jeton d'adresse, l'accès est toujours réalisé sur le canal 1 du logiciel de commande.

S'il est nécessaire de lire des données depuis un canal spécial, le symbole **u** (unité) est ajouté à l'adresse avec le numéro de canal souhaité.

Exemple :

```
nck/Channel/MachineAxis/actFeedRate[3]
nck/Channel/MachineAxis/actFeedRate[u1, 3]
```

3.10.4 Création d'adresses CN/AP pendant l'exécution

Il est possible de créer un descripteur d'adresse pendant l'exécution.

Pour ce faire, on utilise le contenu d'une variable String dans une instruction d'opération ainsi que dans les fonctions nc.cap.read et nc.cap.write.

Pour ce mode d'adressage, il convient de veiller à ce qui suit :

- Placer le nom de variable entre guillemets.
- Faire précéder le nom de variable de trois caractères ‚$'.

Syntaxe :

```
"$$$variable name"
```

Exemple :

```
<PRINT name="var_adr" text="DB9000.DBW%d"> 2000</PRINT>
<OP> "$$$var_adr" = 1 </OP>
```

3.10.5 Adressage de composants d'entraînement

L'adressage des composants d'entraînement commence par la composante de chemin **drive**, suivie de la spécification du groupe d'entraînement :

CU – Control Unit

DC – Drive Control (Motor Module)

CULNK – Modules d'extension (HUBS)

TM – Terminal Module

LM – Line Module

Le paramètre à définir est ajouté à cette composante.

Exemple :

```
<LET name="r0002_content"></LET>
<LET name="p107_content"></LET>

<!— Lecture de la valeur r0002 sur la CU ->

<OP> r0002_content = "drive/cu/r0002" </OP>
<OP> r0002_content = "drive/cu/r0002[CU1]" </OP>

<!— Lecture de la valeur r0002 sur la NX1 ->
<OP> r0002_content = "drive/cu/r0002[CU2]" </OP>

<!— Lecture de la valeur p107[0] sur la CU ->
<OP> p107_content = "drive/cu/p107[0]" </OP>
<PRINT text="%d"> p107_content </PRINT>

<!— Lecture de la valeur p107[0] sur la CU ->

<OP> p107_content = "drive/cu/p107[0, CU1]" </OP>
<PRINT text="%d"> p107_content </PRINT>

<!— Lecture de la valeur p107[0] sur la NX1 ->

<OP> p107_content = "drive/cu/p107[0, CU2]" </OP>
<PRINT text="%d"> p107_content </PRINT>
```

Adressage des objets entraînement :

Pour l'adressage des différents objets, il convient d'indiquer l'objet souhaité entre crochets après le paramètre.

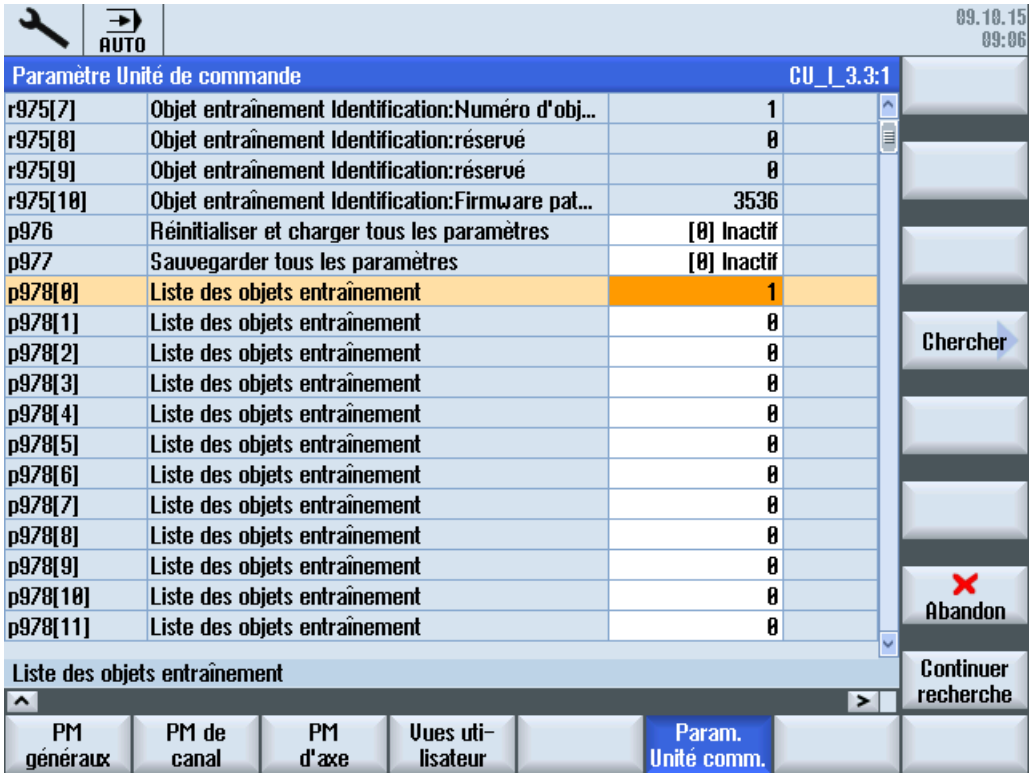


Figure 3-7 Paramètre d'entraînement p0978 [...] sur la commande

```
Numéro de paramètre[do<DO-index>]
```

Exemple :
```
p0092[do1]
```

L'indice d'entraînement peut également être lu au moyen de "caractères de remplacement" **$<Nom de variable>** à partir d'une variable locale.
```
z.B. DO$variable locale
```

Exemple :
```
<DATA name ="drive/cu/p0092">1</DATA>
<DATA name ="drive/dc/p0092[do1] ">1</DATA>
```

Adressage indirect :
```
<LET name = "driveIndex> 0 </LET>
<OP> driveIndex = $ctrlout_module_nr[0, AX1] </OP>
<DATA name ="drive/dc[do$driveIndex]/p0092">1</DATA>
```

3.10.6 Exemple : déterminer le numéro de DO pour un Motor Module

Le numéro de DO d'un Motor Module de type 11 (servo) peut être déterminé comme suit :

Tous les objets entraînement raccordés sont listés en fonction de leur numéro d'emplacement dans le tableau p0978 de la CU correspondante. Dans le même temps, les numéros de composants sont listés dans le tableau p0101 et les types de composants dans le tableau p0107.

Pour les types de composants suivants, il faut toujours utiliser un indexage propre :

CU – Control Units

DC – Drive Controls (Motor Module)

CULNK – Modules d'extension (HUBS)

TM – Terminal Module

LM – Line Module

L'indice d'adressage peut être déterminé en parcourant le tableau p0107 dans l'ordre croissant pour chaque CU raccordée et en incrémentant de 1 l'indice de type à chaque occurrence du type recherché. La valeur de base est 1. Si des composants NX sont trouvés dans ce tableau, le comptage ne reprend à un composant NX que lorsque le tableau actuel a été complètement parcouru. L'exécution des composants NX et CU se déroule dans l'ordre détecté.

Détermination d'indice : CUI avec NX

CUI		NX1		Indice d'adressage	
				DO	CU
p107[0]	[3]SINAMICS				1
p107[2]	[11]SERVO			1	
p107[3]	[11]SERVO			2	
p107[4]	[11]SERVO			3	
p107[5]	[254]CU-LINK				2
p107[6]	[11]SERVO			4	
		p107[1]	[11]SERVO	5	
		p107[2]	[11]SERVO	6	
		p107[3]	[11]SERVO	7	

Cette topologie contient sept Motor Modules. Les indices 1 à 4 adressent les Motor Modules attribués à la CU. Les indices 5 à 7 adressent les Motor Modules de la NX. L'indice 1 doit être utilisé pour accéder à la CU. La NX est adressée avec l'indice 2.

Exemples de scripts : xmldial.xml et drv_sys_hlpfunct.xml

xmldial.xml

```
<DialogGui>

 <?include src="f:\appl\drv_sys_hlpfunct.xml" ?>

 <menu name = "main">
  <open_form name = "main_form" />
  <softkey POSITION="1">
   <caption></caption>
   <navigation>main</navigation>
  </softkey>
 </menu>

 <form name="main_form">
  <init>
   <caption>Component arrangement</caption>
    <let name="count" />
    <let name="str" type="string" />
    <let name="do_name" type="string" />
    <let name="cu_name" type="string" />
    <let name="cui_idx" />

    <op>
     cui_idx = 0;
    </op>

    <function name="load_component" />
    <print text="%d CUs found">num_cus</print>

    <function name="calculate_do_index" />

    <control name="list_comp_no_do_idx" xpos="8" ypos="80"
fieldtype="listbox" width="360" height="500" >
     <property item_data="100" />
    </control>

    <op>
     count = 0;
    </op>

    <while>
    <condition>count &lt; 32 && address_idx_map[$count].comp_no != 0</
condition>

    <op>
     do_name= _T"";
    </op>

    <function name="read_do_name_fast" return="do_name">count</function>
    <function name="read_cu_name_fast"
return="cu_name">address_idx_map[$count].cu_idx</function>

    <print name="str" text="%3d %3d %3d %s
%s">address_idx_map[$count].cu_idx, address_idx_map[$count].comp_no,
address_idx_map[$count].do_idx, do_name, cu_name</print>
```

xmldial.xml

```
    <function name="control.additem">_T"list_comp_no_do_idx", str, count</
function>

    <op>
     count = count +1;
    </op>
    </while>

   <paint>
    <text xpos="8" ypos="30">Motor moduls</text>
    <text xpos="8" ypos="60">CU</text>
    <text xpos="40" ypos="60">Comp.</text>
    <text xpos="92" ypos="60">DO Index</text>
    <text xpos="172" ypos="60">Name</text>
  </paint>

 </form>
</DialogGui>
```

drv_sys_hlpfunct.xml

```
<typedef name="components" type="struct">
 <element name="cu_p978" dim="25" />
 <element name="do_p0101" dim="25" />
 <element name="do_p0107" dim="25" />
</typedef>

<typedef name="comp_no_do_idx_map" type="struct">
 <!-- cu index -->
 <element name="cu_idx" />
 <!-- component number -->
 <element name="comp_no" />
 <!-- address index -->
 <element name="do_idx" />
</typedef>

<let name="componentsList" dim="5" type="components" />
<let name="address_idx_map" dim="32" type="comp_no_do_idx_map" />
<let name="num_cus" />
<let name="_drv_sys_comp_array_size">23</let>

<!-- ------------------------------------------------------------

 function: load_components_description
 This function loads the parameter arrays p978, p101 and p107 into a local
memory

 input:
 cuno: CU index 0 based (index 0 == CUI)

 output:
 componentsList structure

 ------------------------------------------------------------------- -->

<function_body name="load_components_description" parameter="cuno">
 <let name="count" />
 <let name="next_nx" >0</let>
 <let name="cuidx"></let>
 <let name="error" />

 <op>
  count = _drv_sys_comp_array_size;
  next_nx = cuno;
  cuidx = cuno+1;
 </op>

 <print text="gather data" />

 <function name="ncfunc.cap.read" return="error"
rows="$count">componentsList[$cuno].cu_p978, "drive/cu/p0978[0, cu
$cuidx]"</function>
 <function name="ncfunc.cap.read" return="error"
rows="$count">componentsList[$cuno].do_p0101, "drive/cu/p0101[0, cu
$cuidx]"</function>
```

drv_sys_hlpfunct.xml

```
 <function name="ncfunc.cap.read" return="error"
rows="$count">componentsList[$cuno].do_p0107, "drive/cu/p0107[0, cu
$cuidx]"</function>

 <print text="gather data finished" />
 <sleep value="20" />

 <op>
  count = 0;
 </op>

 <while>
  <condition>count < _drv_sys_comp_array_size</condition>
  <if>
   <condition>componentsList[$cuno].cu_p978[$count] == 60</condition>
  <then>
   <op>
    next_nx= next_nx +1;
   </op>
  <print text="next nx %d">next_nx</print>
  <function name="load_components_description">next_nx</function>
  </then>
  </if>

 <op>
  count = count+1;
 </op>
 </while>
 <op>
  num_cus = next_nx+1;
 </op>
</function_body>

<!-- ------------------------------------------------------------

 function: calculate_do_index
 This function is looking for components of type 11 (SERVO) and lists these
in the componentsList array.

 input:
 -

 output:
 componentsList array filled

 ------------------------------------------------------------------- -->

<function_body name="calculate_do_index">
 <let name="cuno" />
 <let name="count" />
 <let name="do_index" >1</let>
 <let name="map_index" />
 <while>
  <condition>
   cuno < num_cus</condition>
   <op>
```

drv_sys_hlpfunct.xml

```
    count = 0;
   </op>
  <while>
   <condition>count < _drv_sys_comp_array_size</condition>
   <if>
    <condition> componentsList[$cuno].do_p0107[$count] == 11</condition>
    <then>
     <op>
      address_idx_map[$map_index].cu_idx = cuno+1;
      address_idx_map[$map_index].comp_no =
componentsList[$cuno].do_p0101[$count];
      address_idx_map[$map_index].do_idx = do_index;
      do_index = do_index+1;
      map_index = map_index +1;
     </op>
    </then>
   </if>
   <op>
    count = count +1;
   </op>
  </while>
  <op>
   cuno = cuno +1;
  </op>
 </while>
</function_body>
```

3.10.7 Adressage des paramètres machine et des données de réglage

Les paramètres d'entraînement et les données de réglage sont identifiés par le caractère $ suivi du nom du paramètre.

Paramètres machine :

$Mx_<nom[**index, AX<numéro d'axe>**]>

Données de réglage :

$Sx_<nom[indice, AX<numéro d'axe>]>

x :

N – paramètres machine ou données de réglage généraux

C – paramètres machine ou données de réglage spécifiques au canal

A – paramètres machine ou données de réglage spécifiques à l'axe

Indice :

Pour un tableau, le paramètre indique l'indice du paramètre.

AX<numéro d'axe> :

Pour les données spécifiques à l'axe, l'axe souhaité (**<numéro d'axe>**) doit être spécifié.

L'indice d'axe peut également être lu au moyen de "caractères de remplacement" **$<nom de variable>** à partir d'une variable locale.

Par ex. AX$variable locale

Exemple :

```
<DATA name ="$MN_AXCONF_MACHAX_NAME_TAB[0] ">X1</DATA>
```

Adressage direct de l'axe :

```
<DATA name ="$MA_CTRLOUT_MODULE_NR[0, AX1] ">1</DATA>
...
...
```

Adressage indirect de l'axe :

```
<LET name ="axisIndex"> 1 </LET>
<DATA name ="$MA_CTRLOUT_MODULE_NR[0, AX$axisIndex] ">1</DATA>
```

3.10.8 Paramètres machine spécifiques à un canal

Si aucun numéro de canal n'est indiqué dans le jeton d'adresse, l'accès est toujours réalisé sur le canal actuellement réglé du logiciel de commande.

S'il est nécessaire de lire des données depuis un canal spécial, le symbole **u** (unité) est ajouté à l'adresse avec le numéro de canal souhaité entre crochets. Dans le cas d'un adressage de tableau, le canal est indiqué entre crochets en tant que dernier argument.

Exemple :

```
$MC_RESET_MODE_MASK
```

ou

```
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]
```

```
$MC_RESET_MODE_MASK[u1]
```

ou

```
$MC_AXCONF_GEOAX_ASSIGN_TAB[0, u1]
$MC_RESET_MODE_MASK
```

3.10.9 Adressage des données utilisateur

L'adressage des données utilisateur commence par la composante de chemin **gud**, suivie par le marquage indiquant si les GUD sont globaux ou spécifiques à un canal. La plage GUD et le nom GUD se joignent à cette partie de l'adresse.

Pour un tableau, il convient d'indiquer l'indice de tableau souhaité entre crochets après le nom.

Exemple :
```
<DATA name ="gud/nck/mgud/syg_rm[0]" />
<OP>"gud/nck/mgud /syg_rm[0]" = 10 </op>
```

Adressage des données utilisateur globales

L'adressage commence par la composante de chemin gud, suivie de la spécification de plage NCK. Cette partie de l'adresse est suivie de la spécification des plages GUD :

Plages GUD (données utilisateur globales)	Affectation
sgud	Données utilisateur globales de Siemens
mgud	Données utilisateur globales du constructeur de machine
ugud	Données utilisateur globales de l'utilisateur

Adressage des données utilisateur spécifiques à un canal

L'adressage commence par la composante de chemin gud, suivie de la spécification de plage CHANNEL. Cette partie de l'adresse est suivie de la spécification des plages GUD.

Indiquer ensuite le nom de GUD. Si un tableau doit être adressé, l'indice de tableau entre crochets suit le nom.

Exemple :
```
<data name ="gud/channel/mgud/syg_rm[0]">1</data>
<op>"gud/channel/mgud/syg_rm[0]" = 5*2 </op>
```

Adressage de tableaux multidimensionnels

Pour l'adressage de tableaux multidimensionnels, l'indice de ligne doit être suivi de l'indice de colonne. Les indices sont séparés par un point.

Concernant les GUD propres à un canal :

- Le numéro de canal doit être noté avec l'identification **u** (unité) suivie du numéro avant ou après la spécification de ligne/colonne.
- Les séquences doivent être séparées les unes des autres par une virgule.
- Lorsqu'aucun numéro de canal n'est indiqué, l'accès se fait au premier canal.

Exemple :
```
"gud/Channel/sgud/_WP[u2, 2.0]"
```

ou

```
"gud/Channel/sgud/_WP[2.0, u2]"
```

3.11 Fonctions prédéfinies

Le langage de script offre différentes fonctions standard mathématiques et de traitement de chaînes. Les noms de fonction présentés ci-après sont réservés et ne peuvent pas être surchargés.

Nom de la fonction	Signification

Ncfunc cap read	La fonction copie une valeur à partir de l'adresse indiquée dans une variable locale. Si la lecture s'effectue sans aucune erreur, la variable Return contient la valeur zéro.
	
	Contrairement à l'instruction d'opération, cette fonction n'interrompt pas l'exécution des instructions de script en cas d'erreur.
	
	Attributs :
	• **return** - État d'exécution
	– Valeur = 0 - Aucune erreur
	– Valeur = 1 - La lecture des variables n'a pas pu être effectuée
	• **rows** - Nombre de lignes à lire supplémentaires dans un tableau (en option)
	Si un indice de tableau est indiqué pour les variables de référence, la fonction copie les valeurs lues à partir de cet indice dans la variable cible.
	
	Syntaxe :
	`<function name="ncfunc.cap.read" return="error"> lokale variable, "address"</function>`
	
	Exemple :
	`<let name="error"></let>`
	`<function name="ncfunc.cap.read" return="error"> 3, "drive/cu/p0009"</function>`
	`<if>`
	`<condition>error != 0</condition>`
	`<then>`
	`<break />`
	`</then>`
	`</if>`
	
	ou
	
	`<let name="cu_p978" dim="25" ></let>`
	`...`
	`...`
	`<function name="ncfunc.cap.read" return="error" rows="23">cu_p978, "drive/cu/p0978[0, 1]"`
	`</function>`

Nom de la fonction	Signification

Ncfunc cap write	La fonction écrit une valeur dans la variable indiquée. Si l'écriture s'effectue sans aucune erreur, la variable Return contient la valeur zéro.
	
	Contrairement à l'instruction d'opération, cette fonction n'interrompt pas l'exécution des instructions de script en cas d'erreur.
	
	Attributs :
	• **return** - État d'exécution
	– Valeur = 0 - Aucune erreur
	– Valeur = 1 - La lecture des variables n'a pas pu être effectuée
	• **rows** - Nombre de lignes à écrire supplémentaires dans un tableau (en option)
	Si un indice de tableau est indiqué pour les variables de référence, la fonction copie les valeurs à partir de cet indice dans la variable cible.
	
	Syntaxe :
	`<function name="ncfunc.cap.read" return="error"> local variable or constant, "address"</function>`
	
	Exemple :
	`<let name="error"></let>`
	`<function name="ncfunc.cap.write" return="error"> 0, "drive/cu/p0009"</function>`
	`<if>`
	`<condition>error != 0</condition>`
	`<then>`
	`<break />`
	`</then>`
	`</if>`
	
	ou
	
	`<let name="cu_p978" dim="25" ></let>`
	`...`
	`...`
	`<function name="ncfunc.cap.write" return="error" rows="23">cu_p978, "drive/cu/p0978[0, 1]"`
	`</function>`

Nom de la fonction	Signification

Ncfunc PI-Service	Le service d'instance de programme (PI) permet de transmettre des requêtes au NCK.
	Si le service s'est exécuté sans problème, la fonction retourne la valeur 1 dans les variables Return.
	
	Manipulation de la liste d'outils :
	_N_CREATO - Créer un outil
	_N_DELETO - Supprimer un outil
	_N_CREACE - Créer un tranchant d'outil
	_N_DELECE - Supprimer un tranchant d'outil
	
	Activation de décalages d'origine :
	_N_SETUFR - Active le frame utilisateur actuel
	_N_SETUDT - Active les données utilisateur actuelles
	
	Recherche de bloc :
	_N_FINDBL - Activer un bloc
	_N_FINDAB - Annuler une recherche de bloc
	
	Reconfiguration des paramètres machine :
	_N_CONFIG - Les paramètres machine avec le niveau de prise d'effet NEW_CONF, saisis l'un après l'autre par l'opérateur, sont activés simultanément.
	
	Syntaxe :
	`<function name="ncfunc.pi_service" return="return var"> pi name, var1, var2, var3, var4, var5`
	`</function>`
	
	Attributs :
	name - Nom de la fonction
	return - Nom de la variable dans laquelle est enregistré le résultat de l'exécution :
	• Valeur == 1 - Requête exécutée avec succès
	• Wert == 0 - Auftrag fehlerhaft
	
	Valeurs de balise :
	pi name - Nom du service PI (String)
	var1 à var5 - Arguments spécifiques à la PI

Nom de la fonction	Signification

Ncfunc PI-Service Suite	**Arguments :**
	• _N_CREATO
	var1 - Numéro d'outil
	• _N_DELETO
	var1 - Numéro d'outil
	• _N_CREACE
	var1 - Numéro d'outil
	var2 - Numéro de tranchant
	• _N_DELECE
	var1 - Numéro d'outil
	var2 - Numéro de tranchant
	• _N_SETUFR
	Aucun argument
	• _N_SETUDT
	var1 - Plage des données utilisateur à activer
	– 1 - Données de correction d'outil
	– 2 - Frame de base actif
	– 3 - Frame réglable actif
	• _N_FINDBL
	var1 - Mode de recherche
	– 2 - Recherche avec calcul de contour
	– 4 - Recherche du point de fin de bloc
	– 1 - Recherche sans calcul
	• _N_FINDAB
	Aucun argument
	• _N_CONFIG
	Aucun argument
	
	Exemple :
	Création d'un outil - Numéro d'outil 3
	
	`<function name="ncfunc.pi_service">_T"_N_CREATO", 3</function>`
	
	Suppression du tranchant 1 de l'outil 5
	
	`<function name="ncfunc.pi_service">_T"_N_DELECE", 5, 1`
	`</function>`

Nom de la fonction	Signification

ncfunc chan PI-Service	La fonction exécute un service PI spécifique au canal. Le numéro de canal est transmis après le nom du service PI. S'ensuivent alors tous les autres paramètres d'appel.
	
	Paramètres :
	channel - Numéro de canal
	
	Syntaxe :
	`<function name="ncfunc.chan_pi_service" return="error">`
	`_T"_N_SETUFR", channel, ...`
	`</function>`
	
	Exemple :
	`<let name="chan" >1</let>`
	`<function name="ncfunc.chan_pi_service" return="error">`
	`_T"_N_SETUFR", chan</function>`
	
	`<function name="ncfunc.chan_pi_service" return="error">`
	`_T"_N_SETUDT", chan, _T"016", _T"00000", _T"00000"</function>`

Ncfunc display resolution	La fonction retourne l'instruction de conversion définie dans la commande pour les nombres à virgule flottante. Une variable String doit être mise à disposition en tant que variable.
	
	Voir aussi les paramètres machine d'affichage MD203 DISPLAY_RESOLUTION et MD204 DISPLAY_RESOLUTION_INCH
	
	Syntaxe :
	`<function name="ncfunc.displayresolution" return="dislay_res" />`
	
	Exemple :
	`<let name="dislay_res" type="string"></let>`
	...
	`<function name="ncfunc.displayresolution" return="dislay_res" />`
	
	`  <control name = "cdistToGo" xpos = "210" ypos = "156" refvar="nck/Channel/GeometricAxis/progDistToGo[2]" hotlink="true" height="34" fieldtype="readonly" format="$$$dislay_res" time="superfast" color_bk="#ffffff"/>`

Nom de la fonction	Signification

Ncfunc Get drive by axis name	La fonction fournit l'indice d'adressage d'un Motor Module en indiquant le nom d'axe correspondant.
	La fonction fournit -1 comme valeur de retour, si l'affectation ne peut pas être déterminée. Cela peut p. ex. être le cas lorsque l'affectation des axes/entraînements n'a pas été effectuée.
	
	Paramètres :
	str - Nom d'axe
	
	Valeur de retour :
	Indice du Motor Module - Nom correspondant à une variable dans laquelle est écrit l'indice déterminé.
	
	Syntaxe :
	`<function name="NCFUNC.GETDRVBYAX_NAME" return="<index>">`
	`<axis name></function>`
	
	Exemple :
	`<let name="drv_idx" />`
	`<function name="NCFUNC.GETDRVBYAX_NAME" return="drv_idx">_T"X1"`
	`</function>`
	`<control name = "c_do2" xpos = "140" ypos = "114" width = "320"`
	`fieldtype="itemlist" refvar="drive/dc/p10[`**`do$ drv_idx`**`]"`
	`hotlink="true" color_bk="#f1f1f1">`
	`<item value="0">Ready</item>`
	`<item value="1">Quick commissioning</item>`
	`<item value="2">Power unit commissioning</item>`
	`<item value="3">Motor commissioning</item>`
	`<item value="4">Encoder commissioning</item>`
	`<item value="5">Technological application/units`
	`</item>`
	`<item value="15">Data sets</item>`
	`<item value="17">Basic positioning commissioning`
	`</item>`
	`<item value="25">Commissioning the position control</item>`
	`<item value="29">Download</item>`
	`<item value="30">Parameter reset</item>`
	`<item value="95">Safety Integrated commissioning`
	`</item>`
	`</control>`
	`</init>`

Nom de la fonction	Signification

Ncfunc Get drive by axis index	La fonction fournit l'indice d'adressage d'un Motor Module en indiquant l'indice d'axe correspondant.
	La fonction fournit -1 comme valeur de retour, si l'affectation ne peut pas être déterminée. Cela peut p. ex. être le cas lorsque l'affectation des axes/entraînements n'a pas été effectuée.
	
	Paramètres :
	index - Indice d'axe
	
	Valeur de retour :
	Indice du Motor Module - Nom correspondant à une variable dans laquelle est écrit l'indice déterminé.
	
	Syntaxe :
	`<function name="NCFUNC.GETDRVBYAX_IDX" return="<index>">`
	`<axis index></function>`
	
	Exemple :
	`<let name="drv_idx" />`
	`<function name="NCFUNC.GETDRVBYAX_IDX" return="drv_idx">1</function>`
	`<control name = "c_do2" xpos = "140" ypos = "114" width = "320"`
	`fieldtype="itemlist" refvar="drive/dc/p10[`**`do$ drv_idx`**`]"`
	`hotlink="true" color_bk="#f1f1f1">`
	`<item value="0">Ready</item>`
	`<item value="1">Quick commissioning</item>`
	`<item value="2">Power unit commissioning</item>`
	`<item value="3">Motor commissioning</item>`
	`<item value="4">Encoder commissioning</item>`
	`<item value="5">Technological application/units`
	`</item>`
	`<item value="15">Data sets</item>`
	`<item value="17">Basic positioning commissioning`
	`</item>`
	`<item value="25">Commissioning the position control</item>`
	`<item value="29">Download</item>`
	`<item value="30">Parameter reset</item>`
	`<item value="95">Safety Integrated commissioning`
	`</item>`
	`</control>`
	`</init>`

Nom de la fonction	Signification

Ncfunc Get drive by drive name	La fonction fournit l'indice d'adressage d'un Motor Module en indiquant le nom du Motor Module.
	La fonction fournit -1 comme valeur de retour, si l'affectation ne peut pas être déterminée. Cela peut p. ex. être le cas lorsque l'affectation des axes/entraînements n'a pas été effectuée.
	
	Paramètres :
	string - Nom du Motor Module
	
	Valeur de retour :
	Indice du Motor Module - Nom correspondant à une variable dans laquelle est écrit l'indice déterminé.
	
	Syntaxe :
	`<function name="NCFUNC.GETDRVBYDRV_NAME" return="<index>">`
	`<drive name></function>`
	
	Exemple :
	`<let name="drv_idx" />`
	`<function name="NCFUNC.GETDRVBYDRV_NAME"`
	`return="drv_idx">_T"SERVO_3.13:4"</function>`
	`<control name = "c_do2" xpos = "140" ypos = "114" width = "320"`
	`fieldtype="itemlist" refvar="drive/dc/p10[`**`do$ drv_idx`**`]"`
	`hotlink="true" color_bk="#f1f1f1">`
	`<item value="0">Ready</item>`
	`<item value="1">Quick commissioning</item>`
	`<item value="2">Power unit commissioning</item>`
	`<item value="3">Motor commissioning</item>`
	`<item value="4">Encoder commissioning</item>`
	`<item value="5">Technological application/units`
	`</item>`
	`<item value="15">Data sets</item>`
	`<item value="17">Basic positioning commissioning`
	`</item>`
	`<item value="25">Commissioning the position control</item>`
	`<item value="29">Download</item>`
	`<item value="30">Parameter reset</item>`
	`<item value="95">Safety Integrated commissioning`
	`</item>`
	`</control>`
	`</init>`

Nom de la fonction	Signification

Ncfunc Get drive by bud address	La fonction fournit l'indice d'adressage d'un Motor Module en indiquant l'adresse de bus du Motor Module.
	La fonction fournit -1 comme valeur de retour, si l'affectation ne peut pas être déterminée. Cela peut p. ex. être le cas lorsque l'affectation des axes/entraînements n'a pas été effectuée.
	
	Paramètres :
	bus - Numéro de bus
	slave - Numéro d'esclave
	component - Numéro de composant
	
	Valeur de retour :
	Indice du Motor Module - Nom correspondant à une variable dans laquelle est écrit l'indice déterminé.
	
	Syntaxe :
	`<function name="NCFUNC.GETDRVBYBUS_ADDR" return="<index>"><bus>,<slave>,<component></function>`
	
	Exemple :
	`<let name="drv_idx" />`
	`<function name="NCFUNC.GETDRVBYBUS_ADDR" return="drv_idx"> 3,13,4 </function>`
	`<control name = "c_do2" xpos = "140" ypos = "114" width = "320" fieldtype="itemlist" refvar="drive/dc/p10[`**`do$ drv_idx`**`]" hotlink="true" color_bk="#f1f1f1">`
	`<item value="0">Ready</item>`
	`<item value="1">Quick commissioning</item>`
	`<item value="2">Power unit commissioning</item>`
	`<item value="3">Motor commissioning</item>`
	`<item value="4">Encoder commissioning</item>`
	`<item value="5">Technological application/units </item>`
	`<item value="15">Data sets</item>`
	`<item value="17">Basic positioning commissioning </item>`
	`<item value="25">Commissioning the position control</item>`
	`<item value="29">Download</item>`
	`<item value="30">Parameter reset</item>`
	`<item value="95">Safety Integrated commissioning </item>`
	`</control>`
	`</init>`

Nom de la fonction	Signification

Ncfunc Get MD limits	La fonction copie les limites d'un paramètre machine dans la variable spécifiée de type de données **StructMDLimits**. La définition de la structure est comprise dans le fichier struct_def.xml.
	Plus d'informations, voir chapitre "Créer un fichier struct_def.xml (Page 142)"
	
	Paramètres :
	range - Permet d'indiquer une chaîne de caractères qui spécifie la plage du paramètre machine :
	• **displayMD** - Paramètres machine d'affichage
	• **generalMD** - Paramètres machine globaux CN
	• **channelMD** - Paramètres machine spécifiques à un canal CN
	• **axisMD** - Paramètres machine spécifiques à un axe CN
	• **generalSettingMD** - Données de réglage globales CN
	• **channelSettingMD** - Données de réglage spécifiques à un canal CN
	• **axisSettingMD** - Données de réglage spécifiques à un axe CN
	• **channelGUD** - Données utilisateur spécifiques à un canal CN
	• **globalGUD** - Données utilisateur globales CN
	MD-number - Numéro du paramètre machine de la plage
	variable name - Après l'appel de fonction, cette variable contient la valeur de la limite
	range/unit - facultatif (par ex., numéro de canal)
	
	Syntaxe :
	`<function name="NCFUNC.GETMD_LIMITS"><range>, <MD-number, <variabel name>, <range/unit>></function>`
	
	Exemple :
	`<let name="limits" type="StructMDLimits" /> <function name="NCFUNC.GETMD_LIMITS">_T"generalMD", 19220, _T"limits"</function> <op> upper_BAGS = limits.upperLimit; </op>`

Ncfunc bico to int	La fonction convertit une chaîne spécifiée dans le format FCOM en valeur de nombre entier (Integer) (voir SINAMICS).
	
	Syntaxe :
	`<function name="ncfunc.bicotoint" return="integer variable"> bico-string</function>`
	
	Exemple :
	`<let name="s_np0480_0" type="string"></let> <let name="i_p0480_0">0</let>`
	
	`<function name="ncfunc.bicotoint" return="i_p0480_0">s_np0480_0 </function>`

Nom de la fonction	Signification

Ncfunc int to bico	La fonction convertit une valeur de nombre entier (Integer) en chaîne au format FCOM (voir SINAMICS).
	
	Syntaxe :
	`<function name="ncfunc.inttobico"`
	`return="string variable">integer variable</function>`
	
	Exemple :
	`<function name="ncfunc.inttobico"`
	`return="s_p0480_0">"drive/dc/p0480[0, DO2]"</function>`

Ncfunc is bico str valid	La fonction retourne la valeur zéro lorsqu'il s'agit d'une chaîne spécifiée au format FCOM (voir SINAMICS).
	
	Syntaxe :
	`<function name="ncfunc.isbicostrvalid"`
	`return="integer variable">string varaible</function>`
	
	Exemple :
	`…`
	`<let name="s_np0480_0" type="string"></let>`
	`…`
	`…`
	`<control name = "cp0480_0" xpos = "402"`
	`ypos = "76" hotlink="true" refvar="s_np0480_0" >`
	`<property item_data="4001" />`
	`</control>`
	`…`
	`…`
	`<function name="ncfunc.isbicostrvalid"`
	`return="valid">cp0480_0</function>`

Nom de la fonction	Signification

Ncfunc password	La fonction définit ou supprime un niveau de mot de passe de la CN.
	• Définition du mot de passe :
	Le mot de passe doit être spécifié sous forme de paramètre pour le niveau de mot de passe souhaité.
	• Suppression du mot de passe :
	Une chaîne vide supprime le niveau de mot de passe.
	
	Syntaxe :
	`<function name="ncfunc.password">password`
	`</function>`
	
	Exemple :
	`<let name="mot de passe" type="string"></let>`
	
	`<function name="ncfunc.password" > password`
	`</function>`
	`<function name="ncfunc.password" > _T"CUSTOMER"`
	`</function>`
	
	Suppression du mot de passe :
	`<function name="ncfunc.password" > _T"" </function>`

Control form color	La fonction retourne la couleur de texte ou d'arrière-plan de la boîte de dialogue comme chaîne de caractères.
	Plus d'informations, voir chapitre "Code couleur (Page 75)"
	
	Plage :
	• BACKGROUND – Demander la valeur de couleur de l'arrière-plan
	• TEXT – Demander la valeur de couleur du texte (premier plan)
	
	Syntaxe :
	`<function name="control.formcolor"`
	`return="variable">_T"Zone"</function>`
	
	Exemple :
	`<let name="bk_color" type="string"></let>`
	
	`<function name="control.formcolor"`
	`return="bk_color">_T"BACKGROUND"</function>`

Nom de la fonction	Signification

Control local time	La fonction copie l'heure locale dans un tableau avec 7 éléments de tableau.
	Le nom de la variable est attendu en tant que paramètre d'appel.
	Ce qui suit est enregistré dans l'élément de tableau :
	• Indice 0 - Année
	• Indice 1 - Mois
	• Indice 2 - Jour de semaine
	• Indice 3 - Jour
	• Indice 4 - Heure
	• Indice 5 - Minute
	• Indice 6 - Seconde
	
	Syntaxe :
	`<function name ="control.localtime">_T"time_array"`
	`</function>`
	
	Exemple :
	`<!-- index`
	`0 = Year`
	`1 = Month`
	`2 = Day of week`
	`3 = Day`
	`4 = Hour`
	`5 = Minute`
	`6 = Second`
	`-->`
	`<let name="time_array" dim="7" />`
	
	`<function name ="control.localtime">_T"time_array"`
	`</function>`

Control exist	La fonction fournit 1 comme valeur de retour si l'élément de commande spécifié existe.
	
	Syntaxe :
	`<function name="CONTROL.EXIST"`
	`return="<return variable>"><control name></function>`
	
	Exemple :
	`<let name="ret" /> <function name="CONTROL.EXIST"`
	`return="ret">_T"edit"</function>`

Control is modified	La fonction fournit 1 comme valeur de retour si le contenu de l'élément de commande Edit spécifié a été modifié.
	
	Syntaxe :
	`<function name="CONTROL.ISMODIFIED"`
	`return="<return variable>"><control name></function>`
	
	Exemple :
	`<let name="ret" /> <function name="CONTROL.ISMODIFIED"`
	`return="ret">_T"edit"</function>`

Nom de la fonction	Signification

Control is visible	La fonction fournit 1 comme valeur de retour si l'élément de commande spécifié est visible.
	
	Syntaxe :
	`<function name="CONTROL.ISVISIBLE"`
	`return="<return variable>"><control name></function>`
	
	Exemple :
	`<let name="ret" /><function name="CONTROL.ISVISIBLE"`
	`return="ret">_T"edit"</function>`

Control set modified	La fonction modifie le flag Modified de l'élément de commande Edit spécifié.
	
	Paramètres :
	control name
	• Valeur = 1 - marquer le champ comme modifié
	• Valeur = 0 - marquer le champ comme non modifié
	
	Syntaxe :
	`<function name="CONTROL.SETMODIFIED"`
	`return="<return variable>"><control name>,`
	`<Flag></function>`
	
	Exemple :
	`<let name="b" >1</let>`
	`<let name="ret" />`
	`<control name="edit" xpos="10" ypos="80" width="30" refvar="b"`
	`hotlink = "true" />`
	`<function name="CONTROL.SETMODIFIED"`
	`return="ret">_T"edit", 1</function>`

Nom de la fonction	Signification

Control set working area	Si le contenu d'une **vue de défilement** est modifié, p. ex. en supprimant ou en ajoutant des éléments de commande, la zone visible de la vue de défilement doit être recalculée.
	L'appel de cette fonction permet d'effectuer le calcul.
	
	Valeur :
	Nom de la **vue de défilement**
	
	Exemple :
	`...`
	`<control name="area" xpos="8" ypos="72" width="554" height="320" fieldtype="scrollarea" >`
	`  <control name="test" xpos="20" ypos="40"  width="20" fieldtype="readonly" />`
	`</control >`
	
	`<control name="test1" xpos="20" ypos="80"  width="20" fieldtype="readonly" parent="area"/>`
	`<control name="test2" xpos="20" ypos="100"  width="20" fieldtype="readonly" parent="area"/>`
	
	`<function name="CONTROL.SETWORKINGAREA">_T"area"</function>`
	`...`

String to compare	Deux chaînes sont comparées entre elles sur le plan lexicographique.
	La fonction retourne la valeur de retour zéro si les chaînes sont identiques, une valeur inférieure à zéro si la première chaîne est plus petite que la deuxième, ou une valeur supérieure à zéro si la deuxième chaîne est plus petite que la première.
	Paramètres :
	str1 - Chaîne
	str2 - Chaîne de comparaison
	
	Syntaxe :
	`<function name="string.cmp" return ="<int var>" > str1, str2 </function>`
	
	Exemple :
	`<let name="rval">0</let>`
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string">A brown bear hunts a brown dog.</let>`
	
	`<function name="string.cmp" return="rval"> str1, str2 </function>`
	Résultat :
	rval= 0

Nom de la fonction	Signification

String to compare sans tenir compte de la casse	Deux chaînes peuvent être comparées entre elles du point de vue lexicographique sans tenir compte de la casse.
	La fonction retourne la valeur de retour zéro si les chaînes sont identiques, une valeur inférieure à zéro si la première chaîne est plus petite que la deuxième, ou une valeur supérieure à zéro si la deuxième chaîne est plus petite que la première.
	
	Paramètres :
	str1 - Chaîne
	str2 - Chaîne de comparaison
	
	Syntaxe :
	`<function name="string.icmp" return ="<int var>" > str1, str2 </function>`
	
	Exemple :
	`<let name="rval">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string">A brown Bear hunts a brown Dog.</let>`
	`<function name="string.icmp" return="rval"> str1, str2 </function>`
	Résultat :
	rval= 0

String left	La fonction extrait le nombre nCount de caractères à gauche de la chaîne 1 et les copie dans la variable Return.
	
	Paramètres :
	str1 - Chaîne
	nCount - Nombre de caractères
	
	Syntaxe :
	`<function name="string.left" return="<result string>"> str1, nCount </function>`
	
	Exemple :
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let>`
	
	`<function name="string.left" return="str2"> str1, 12 </function>`
	Résultat :
	str2="A brown bear"

Nom de la fonction	Signification

String right	La fonction extrait le nombre nCount de caractères à droite de la chaîne 1 et les copie dans la variable Return.
	
	Paramètres :
	str1 - Chaîne
	nCount - Nombre de caractères
	
	Syntaxe :
	`<function name="string.right" return="<result string>"> str1, nCount </function>`
	
	Exemple :
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let>`
	
	`<function name="string.right" return="str2"> str1, 10 </function>`
	Résultat :
	str2="brown dog."

String middle	La fonction extrait le nombre de caractères spécifié de la chaîne 1 à partir de l'indice iFirst et les copie dans la variable Return.
	
	Paramètres :
	str1 - Chaîne
	iFirst - Indice de départ
	nCount - Nombre de caractères
	
	Syntaxe :
	`<function name="string.middle"`
	`return="<result string>"> str1, iFirst, nCount </function>`
	
	Exemple :
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let>`
	
	`<function name="string.middle" return="str2"> str1, 2, 5 </function>`
	Résultat :
	str2="brown"

Nom de la fonction	Signification

String length	La fonction fournit le nombre de caractères d'une chaîne.
	
	Paramètres :
	str1 - Chaîne
	
	Syntaxe :
	`<function name="string.length" return="<int var>"> str1 </function>`
	
	Exemple :
	`<let name="length">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<function name="string.length" return="length"> str1 </function>`
	Résultat :
	length = 31

Strings to replace	La fonction remplace toutes les chaînes partielles trouvées par la nouvelle chaîne.
	
	Paramètres :
	string - Variable String
	find string - Chaîne à remplacer
	new string - Nouvelle chaîne
	
	Syntaxe :
	`<function name="string.replace"> string, find string, new string </function>`
	
	Exemple :
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	
	`<function name="string.replace" > str1, _T"a brown dog" , _T"a big salmon"</function>`
	Résultat :
	str1 = "A brown bear hunts a big salmon!"

Nom de la fonction	Signification

Strings to remove	La fonction supprime toutes les chaînes partielles trouvées.
	
	Paramètres :
	string - Variable String
	remove string - Chaîne partielle à supprimer
	
	Syntaxe :
	`<function name="string.remove"> string, remove string </function>`
	
	Exemple :
	`<let name="index">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	`<function name="string.remove" > str1, _T"a brown dog" </function>`
	Résultat :
	str1 = "A brown bear hunts"

Strings to insert	La fonction insère une chaîne au niveau de l'indice spécifié.
	
	Paramètres :
	string - Variable String
	index - Indice (base zéro)
	insert string - Chaîne à ajouter
	
	Syntaxe :
	`<function name="string.insert"> string, index, insert string`
	`</function>`
	
	Exemple :
	`<let name="str1" type="string">A brown bear hunts. </let>`
	`<let name="str2" type="string">a brown dog</let>`
	
	`<function name="string.insert" > str1, 19, str2`
	`</function>`
	Résultat :
	str1 = "A brown bear hunts a brown dog"

Nom de la fonction	Signification

String delete	La fonction supprime le nombre de caractères défini à partir de la position de départ indiquée.
	
	Paramètres :
	string - Variable String
	start index - Indice de départ (base zéro)
	nCount - Nombre de caractères à supprimer
	
	Syntaxe :
	`<function name="string.delete"> string, start index , nCount`
	`</function>`
	
	Exemple :
	`<let name="str1" type="string">A brown bear hunts. </let>`
	
	`<function name="string.delete" > str1, 2, 5`
	`</function>`
	Résultat :
	str1 = "A bear hunts"

String find	La fonction recherche dans la chaîne transmise la première correspondance avec la chaîne partielle.
	Si la chaîne partielle est trouvée, la fonction fournit l'indice sur le premier caractère (commençant par zéro), sinon -1.
	
	Paramètres :
	string - Variable String
	findstring - Chaîne à rechercher
	startindex – Indice de départ (facultatif)
	
	Syntaxe :
	`<function name="string.find" return="<int val>"> str1, find string`
	`</function>`
	
	Exemple :
	`<let name="index">0</let>`
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	`<function name="string.find" return="index"> str1, _T"brown"`
	`</function>`
	
	Résultat :
	Indice = 2
	
	ou
	`<function name="string.find" return="index"> str1, _T"brown", 1`
	`</function>`

Nom de la fonction	Signification

String reverse find	La fonction recherche dans la chaîne transmise la dernière correspondance avec la chaîne partielle.
	Si la chaîne partielle est trouvée, la fonction fournit l'indice sur le premier caractère (commençant par zéro), sinon -1.
	
	Paramètres :
	string - Variable String
	find string - Chaîne à rechercher
	startindex – Indice de départ (facultatif)
	
	Syntaxe :
	`<function name="string.reversefind"`
	`return="<int val>"> str1, find string </function>`
	
	Exemple :
	`<let name="index">0</let>`
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	`<function name="string.reversefind" return="index"> str1, _T"brown"`
	`</function>`
	
	Résultat :
	Indice = 21
	
	ou
	`<function name="string.reversefind"`
	`return="index"> str1, _T"brown" 10 </function>`
	
	Résultat :
	Indice = 2

String GetAt	La fonction lit un caractère à partir de la position spécifiée.
	Paramètres :
	str - Chaîne
	index - Indice de base zéro sur le caractère à lire
	
	Valeur de retour :
	Il convient d'indiquer le nom d'une variable String dans laquelle le caractère doit être enregistré.
	
	Syntaxe :
	`<function name="string.getat" return="<result string>"><string>,`
	`<index></function>`
	
	Exemple :
	`<let name="resStr" type="string" />`
	`<function name="string.getat"`
	`return="resStr">_T"brown", 2</function>`

Nom de la fonction	Signification

String pixel height	Calcul de la hauteur d'une chaîne de caractères en pixels. Le calcul est effectué en utilisant la police de caractères actuelle de l'élément de commande spécifié ou de la forme. Le nom de l'élément de commande doit être transféré en tant que chaîne de caractères. Si une chaîne vide est indiquée, le calcul se rapporte à la forme.
	
	Paramètres :
	control name
	Chaîne de caractères
	La chaîne de caractères peut être indiquée sous forme de texte ou une variable String doit être spécifiée.
	
	Return :
	Est attendu le nom d'une variable Integer dans laquelle la hauteur est écrite.
	
	Syntaxe :
	`<function name="STRING.PIXELHEIGHT"`
	`return="<return variable>"><control name>,`
	`<varaible or text></function>`
	
	Exemple :
	Calcul de la largeur par rapport à la police de caractères d'une forme
	`<let name="pixelsh" />`
	`...`
	`<function name="STRING.PIXELHEIGHT"`
	`return="pixelsh">_T"", cedit1</function>`
	
	Calcul de la largeur par rapport à la police de caractères d'un élément de commande
	`<function name="STRING. PIXELHEIGHT"`
	`return="pixelsh"> cedit1, cedit1</function>`

Nom de la fonction	Signification

String pixel width	Calcul de la largeur d'une chaîne de caractères en pixels. Le calcul est effectué en utilisant la police de caractères actuelle de l'élément de commande spécifié ou de la forme. Le nom de l'élément de commande doit être transféré en tant que chaîne de caractères. Si une chaîne vide est indiquée, le calcul se rapporte à la forme.
	
	Paramètres :
	control name
	Chaîne de caractères
	La chaîne de caractères peut être indiquée sous forme de texte ou une variable String doit être spécifiée.
	
	Return :
	Est attendu le nom d'une variable Integer dans laquelle la largeur est écrite.
	
	Syntaxe :
	`<function name="STRING.PIXELWIDTH"`
	`return="<return variable>"><control name>,`
	`<varaible or text></function>`
	
	Exemple :
	Calcul de la largeur par rapport à la police de caractères d'une forme
	`<let name="pixels" />`
	`...`
	`<function name="STRING. PIXELWIDTH"`
	`return="pixels">_T"", cedit1</function>`
	
	Calcul de la largeur par rapport à la police de caractères d'un élément de commande
	`<function name="STRING.PIXELWIDTH"`
	`return="pixels"> cedit1, cedit1</function>`

String SetAt	La fonction écrit un caractère à la position spécifiée. L'indice doit être inférieur à la longueur de texte maximale.
	
	Paramètres :
	str - Chaîne
	char - Caractère devant être écrit à la position spécifiée
	index - Indice de base zéro sur la chaîne de caractères de la variable cible
	
	Syntaxe :
	`<function name="string.setat" ><destination string>,`
	`<character string>, <index></function>`
	
	Exemple :
	`<let name="str" type="string" >br_wn</string>`
	`<function name="string.setat">str, ">_T"o", 2`
	`</function>`

Nom de la fonction	Signification

String Split	La fonction divise une chaîne en un certain nombre de chaînes partielles et les copie dans le tableau String indiqué. La séparation a lieu au niveau du séparateur indiqué. Le séparateur n'est pas enregistré dans la chaîne partielle.
	La fonction agrandit le tableau automatiquement lorsque le nombre de séparateurs trouvés est supérieur à la taille de tableau définie.
	
	Paramètres :
	str - Chaîne
	char - Nom correspondant à une variable qui contient le séparateur
	number - Nom correspondant à une variable qui contient le nombre de chaînes partielles générées après l'exécution de la fonction.
	
	Valeur de retour :
	Il convient d'indiquer le nom d'un tableau String contenant les chaînes partielles après l'exécution de la fonction.
	
	Syntaxe :
	`<function name=" string.split"`
	`return="<result string array>"><string>, <char>,`
	`<number></function>`
	
	Exemple :
	`<let name="strlist" type="string" dim="2"/>`
	`<let name="str_num" />`
	`<function name="string.split" return="strlist">`
	`_T"brown;green;blue;red", _T";", str_num </function>`

String trim left	La fonction supprime les espaces de début d'une chaîne.
	
	Paramètres :
	str1 - Variable String
	
	Syntaxe :
	`<function name="string.trimleft" > str1 </function>`
	
	Exemple :
	`<let name="str1" type="string">test trim left</let>`
	`<function name="string.trimleft" > str1 </function>`
	Résultat :
	str1 = "test trim left"

Nom de la fonction	Signification

String trim right	La fonction supprime les espaces de fin d'une chaîne.
	
	Paramètres :
	str1 - Variable String
	
	Syntaxe :
	`<function name="string.trimright" > str1 </function>`
	
	Exemple :
	`<let name="str1" type="string"> test trim right </let>`
	`<function name="string.trimright" > str1`
	`</function>`
	Résultat :
	str1 = "test trim right"

Sinus	La fonction calcule le sinus de la valeur transmise en degrés.
	
	Paramètres :
	double - Angle
	
	Syntaxe :
	`<function name="sin" return="<double val>"> double </function>`
	
	Exemple :
	`<let name= "sin_val" type="double"></let>`
	`<function name="sin" return="sin_val"> 20.0`
	`</function>`

Cosinus	La fonction calcule le cosinus de la valeur transmise en degrés.
	
	Paramètres :
	double - Angle
	
	Syntaxe :
	`<function name="cos" return="<double val>"> double </function>`
	
	Exemple :
	`<let name= "cos_val" type="double"></let>`
	`<function name="cos" return="cos_val"> 20.0`
	`</function>`

Nom de la fonction	Signification

Tangente	La fonction calcule la tangente de la valeur transmise en degrés.
	
	Paramètres :
	double - Angle
	
	Syntaxe :
	`<function name="tan" return="<double val>"> double </function>`
	
	Exemple :
	`<let name= "tan_val" type="double"></let>`
	`<function name="tan" return="tan_val"> 20.0`
	`</function>`

ARCSIN	La fonction calcule l'arc sinus de la valeur transmise en degrés.
	
	Paramètres :
	double - x dans la plage de -PI/2 à +PI/2
	
	Syntaxe :
	`<function name="arcsin" return="<double val>"> double </function>`
	
	Exemple :
	`<let name= "arcsin_val" type="double"></let>`
	`<function name="arcsin" return="arcsin_val"> 20.0`
	`</function>`

ARCOS	La fonction calcule l'arc cosinus de la valeur transmise en degrés.
	
	Paramètres :
	double - x dans la plage de -PI/2 à +PI/2
	
	Syntaxe :
	`<function name="arcos" return="<double val>"> double </function>`
	
	Exemple :
	`<let name= "arccos_val" type="double"></let>`
	`<function name="arccos" return="arccos_val"> 20.0`
	`</function>`

ARCTAN	La fonction calcule l'arc tangente de la valeur transmise en degrés.
	
	Paramètres :
	double - Arc tangente de y/x
	
	Syntaxe :
	`<function name="arctan" return="<double val>"> double </function>`
	
	Exemple :
	`<let name= "arctan_val" type="double"></let>`
	`<function name="arctan" return="arctan_val"> 20.0`
	`</function>`

Nom de la fonction	Signification
Traitement de fichier	

Lecture d'un fichier	La fonction lit le contenu du fichier spécifié dans une variable String.
	Il est éventuellement possible d'indiquer le nombre de caractères à lire comme second paramètre.
	
	Attribut :
	name - Le nom de fichier doit être saisi en minuscules.
	Un chemin relatif qui utilise le répertoire appl ou dvm comme point de départ permet d'accéder à des fichiers dans d'autres répertoires.
	return - Nom des variables locales
	
	Paramètres :
	progname - Nom de fichier
	number of characters - Nombre de caractères à lire, en octets (facultatif)
	
	Syntaxe :
	`<function name="doc.readfromfile"`
	`return="<string var>"> progname, number of characters`
	`</function>`
	
	Exemple :
	`<let name = "my_var" type="string" ></let>`
	
	Système de fichiers CN
	`<function name="doc.readfromfile"`
	`return="my_var"> _T"n:\mpf\test.mpf"`
	`</function>`
	
	Carte CF
	`<function name="doc.readfromfile"`
	`return="my_var"> _T"f:\appl\test.mpf"`
	`</function>`
	
	ou
	
	`<function name="doc.readfromfile"`
	`return="my_var"> _T".\test.mpf"`
	`</function>`

Nom de la fonction	Signification

Ecriture d'un fichier	La fonction écrit le contenu d'une variable String dans le fichier spécifié.
	Le nom de fichier doit être saisi en minuscules.
	Un chemin relatif qui utilise le répertoire appl ou dvm comme point de départ permet d'accéder à des fichiers dans d'autres répertoires.
	
	Paramètres :
	progname - Nom de fichier
	str1 - Chaîne
	
	Syntaxe :
	`<function name="doc.writetofile" > progname, str1 </function>`
	
	Exemple :
	`<let name = "my_var" type="string" > file content </let>`
	
	Système de fichiers CN
	`<function name="doc.writetofile">_T"n:\mpf\test.mpf", my_var </function>`
	
	Carte CF
	`<function name="doc.writetofile">_T"f:\appl\test.mpf", my_var </function>`
	
	ou
	
	`<function name="doc.writetofile">_T".\test.mpf", my_var </function>`

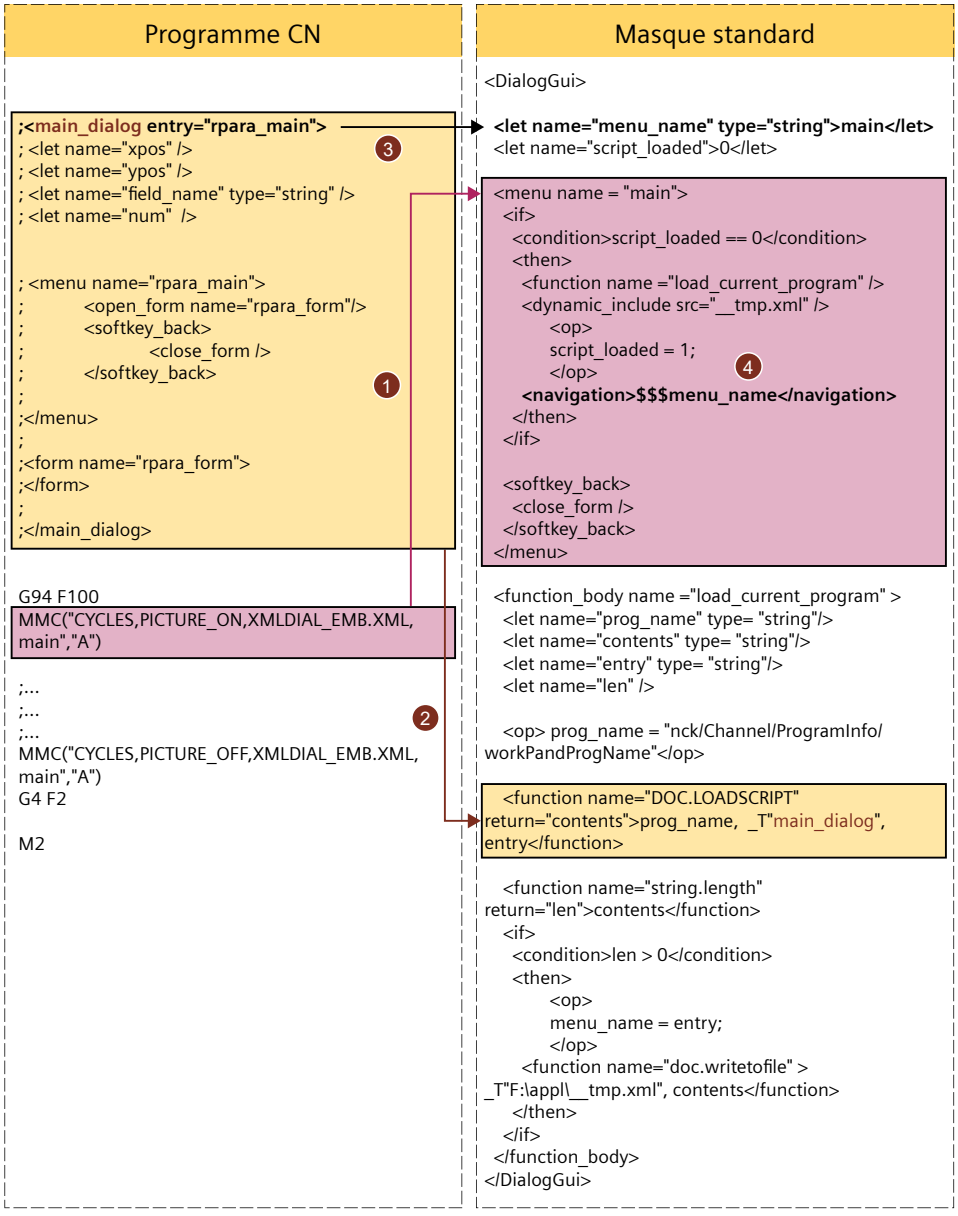
Nom de la fonction	Signification

Suppression d'un fichier	La fonction supprime le fichier spécifié du répertoire.
	Le nom de fichier doit être saisi en minuscules.
	Un chemin relatif qui utilise le répertoire appl ou dvm comme point de départ permet d'accéder à des fichiers dans d'autres répertoires.
	
	Paramètres :
	progname - Nom de fichier
	
	Syntaxe :
	`<function name="doc.remove" > progname </function>`
	
	Exemple :
	Système de fichiers CN
	`<function name="doc.remove">_T"n:\mpf\test.mpf"`
	`</function>`
	
	Carte CF
	`<function name="doc.remove">_T"f:\appl\test.mpf"`
	`</function>`
	
	ou
	
	`<function name="doc.remove">_T".\test.mpf"`
	`</function>`

Nom de la fonction	Signification

Extraction de parties de script	La fonction copie dans la variable locale indiquée une description de la boîte de dialogue insérée dans un programme pièce.
	
	Comme paramètre d'appel, il faut indiquer le nom du programme, celui de la boîte de dialogue et une variable pour l'enregistrement du nom de menu principal. Si le nom de la description de la boîte de dialogue a été trouvé dans le programme pièce, la variable de retour contient cette description. Si le contenu de la variable est enregistré dans un fichier, le script peut être exécuté par un appel indirect.
	
	Le système met à disposition un script qui extrait la description de la boîte de dialogue du programme pièce actif et active le dialogue. Ce script peut être appelé dans une instruction MMC afin d'activer le masque appartenant au programme pièce.
	
	Syntaxe :
	`<function name="doc.loadscript"`
	`return="<name of script variable>">progname, _T"dialog part name",`
	`main menu </function>`
	
	Attribut :
	return - Variable dans laquelle est enregistrée le script extrait
	
	Paramètres :
	progname - Chemin d'accès complet au programme (le nom du chemin d'accès peut être transmis à la fonction en notation DOS)
	main menu - Le nom de menu trouvé est copié dans cette variable
	dialog part name - Nom de balise dans laquelle la description de la boîte de dialogue est insérée
	
	Exemple :
	`<function name="doc.loadscript" return="contents">prog_name,`
	`_T"main_dialog", entry</function>`

Nom de la fonction	Signification
Extraction de parties de script Suite	**Diagramme exemple :**



Nom de la fonction	Signification

Exist	Si le fichier existe, la fonction retourne la valeur 1.
	Le nom de fichier doit être saisi en minuscules.
	Un chemin relatif qui utilise le répertoire appl ou dvm comme point de départ permet d'accéder à des fichiers dans d'autres répertoires.
	
	Paramètres :
	progname - Nom de fichier
	
	Syntaxe :
	`<function name="doc.exist" return="<int_var>" > progname </function>`
	
	Exemple :
	`<let name ="exist">0</let>`
	
	Système de fichiers CN
	`<function name="doc.exist" return="exist">_T"n:\mpf\test.mpf" </function>`
	
	Carte CF
	`<function name="doc.exist" return="exist">_T"f:\appl\test.mpf" </function>`
	
	ou
	
	`<function name="doc.exist" return="exist">_T".\test.mpf" </function>`

Sélection de programme CN	La fonction sélectionne le programme spécifié pour l'exécution. Le programme doit être stocké dans le système de fichiers CN.
	
	Paramètres :
	progname - Nom de fichier
	
	Syntaxe :
	`<function name="ncfunc.select"> progname </function>`
	
	Exemple :
	Système de fichiers CN
	`<function name="ncfunc.select"> _T"n:\mpf\test.mpf" </function>`

Nom de la fonction	Signification

Forçage d'un bit individuel	La fonction sert à manipuler les bits individuels des variables spécifiées.
	Les bits peuvent être mis à 1 ou à 0.
	
	Syntaxe :
	`<function name="ncfunc.bitset" refvar="address" value="set/reset" > bit0, bit1, … bit9 </function>`
	
	Attributs :
	refvar - indique le nom de la variable dans laquelle la combinaison de bits doit être écrite.
	value – Valeur de bit plage de valeurs 0 et 1
	
	Valeurs :
	Les numéros de bits doivent être transmis en tant que valeurs de fonction en commençant par 0.
	Au maximum 10 bits peuvent être modifiés par appel.
	
	Exemple :
	`<function name="ncfunc.bitset" refvar="nck/Channel/Parameter/R[1]" value="1" > 0, 2, 3, 7 </function>`
	
	`<function name="ncfunc.bitset" refvar="nck/Channel/Parameter/R[1]" value="0" > 1, 4 </function>`

Supprimer un élément de commande	La fonction supprime l'élément de commande spécifiée.
	
	Syntaxe :
	`<function name="control.delete"> control name </function>`
	
	Attribut :
	name – Nom de la fonction
	
	Valeur :
	control name – Nom de l'élément de commande
	
	Exemple :
	`<function name="control.delete"> _T"my_editfield" </function>`

Nom de la fonction	Signification

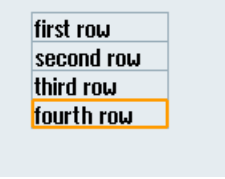
Add Item	La fonction insère un nouvel élément à la fin de la liste.
	Remarque :
	La fonction n'est disponible que pour les types d'élément de commande "listbox" et "graphic-box".
	
	Syntaxe :
	`<function name="control.additem"> control name, item </function>`
	
	Attribut :
	name – Nom de la fonction
	
	Valeurs :
	control name – Control name
	item - Expression à insérer
	itemdata - Valeur entière, définie par l'utilisateur
	
	Exemple :
	`<let name ="itemdata">1</let>`
	…
	…
	…
	`<op> item_string = _T"text1" </op>`
	`<function name="control.additem">_T"listbox1", item_string, itemdata </function>`

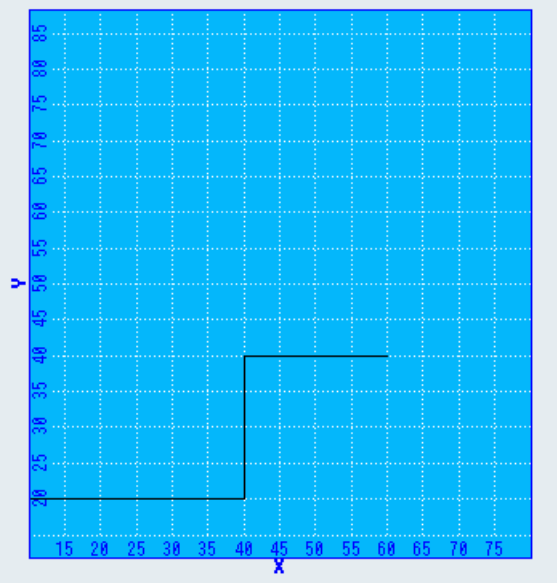
Nom de la fonction	Signification

Insert Item	La fonction insère un nouvel élément à la position indiquée.
	Remarque :
	La fonction n'est disponible que pour le type d'élément de commande "listbox".
	
	Syntaxe :
	`<function name="control.insertitem"> control name, index, item, itemdata </function>`
	
	Attribut :
	name – Nom de la fonction
	
	Valeurs :
	control name - Control name
	index - Position commençant par zéro
	item - Expression à insérer
	itemdata - Valeur entière, définie par l'utilisateur
	
	Exemple :
	`<let name ="itemdata">1</let>`
	…
	…
	…
	`<op> item_string = _T"text2" </op>`
	`<function name="control.insertitem">_T"listbox1", 1, item_string, itemdata </function>`

Delete Item	La fonction supprime un élément à la position indiquée.
	Remarque :
	La fonction n'est disponible que pour le type d'élément de commande "listbox".
	
	Syntaxe :
	`<function name="control.deleteitem"> control name, index </function>`
	
	Attribut :
	name - Nom de la fonction
	
	Valeurs :
	control name - Control name
	index- Indice commençant par 0
	
	Exemple :
	`<function name="control.deleteitem">_T"listbox1", 1</function>`

Nom de la fonction	Signification

Load Item	La fonction insère une liste d'expressions dans l'élément de commande.
	La fonction n'est disponible que pour les types d'élément de commande "listbox" et "graphic-box".
	
	Syntaxe :
	`<function name="control.loaditem"> control name, list </function>`
	
	Attribut :
	name – Nom de la fonction
	
	Valeurs :
	control name – Control name
	Variable String **list**
	La chaîne de caractères contenue dans la variable doit correspondre à la structure décrite ci-dessous.
	
	Structure de la liste :
	La liste contient un nombre d'expressions à séparer les unes des autres par \\n.
	
	Exemple :
	Dans l'exemple, le contenu est affecté au menu déroulant à l'aide de la fonction control.loaditem.
	Le caractère de commande \\n sépare les unes des autres les expressions appartenant à la ligne.
	`<CONTROL name = "list" xpos = "160" ypos = "32"`
	`width ="100" height="200" fieldtype="listbox"/>`
	`<let name="lb_list" type="string" />`
	`<op>`
	`lb_list = _T"first row\\n";`
	`lb_list = lb_list + _T"second row\\n";`
	`lb_list = lb_list + _T"third row\\n";`
	`lb_list = lb_list +_T"fourth row\\n";`
	`</op>`
	`<function name="control.loaditem">_T"list",`
	`lb_list</function>`
	
	 

Nom de la fonction	Signification
Load Item Suite	**Exemple :** Dans l'exemple, le contenu est affecté à la boîte graphique à l'aide de la fonction control.loaditem. Le caractère de commande \\n sépare les uns des autres les éléments graphiques. `<control name= "c_gbox" xpos = "6" ypos="32" width="328" height="356" fieldtype="graphicbox" /> <let name="plot_list" type="string" /> <op> plot_list = _T"l;10;20;40;20\\n"; plot_list = plot_list + _T"l;40;20;40;40\\n"; plot_list = plot_list + _T"l;40;40;60;40\\n"; plot_list = plot_list +_T"l;80;0;80;100\\n"; </op> <function name="control.loaditem">_T"c_gbox", plot_list</function>` 

Empty	La fonction supprime le contenu des éléments de commandes de menu déroulant et de boîte graphique spécifiées.
	
	Syntaxe :
	`<function name="control.empty"> control name, </function>`
	
	Attribut :
	name – Nom de la fonction
	
	Valeurs :
	control name – Control name
	
	Exemple :
	`<function name="control.empty">_T"listbox1" </function>`

Nom de la fonction	Signification

Get focus	La fonction fournit le nom de l'élément de commande qui est activé pour la saisie.
	
	Syntaxe :
	`<function name="control.getfocus" return="focus_name" />`
	
	Attributs :
	name – Nom de la fonction
	return – Il convient de spécifier une variable String dans laquelle le nom de l'élément de commande est copié.
	
	Exemple :
	`<let name>="focus_field" type="string"></let>`
	`<function name="control.getfocus" return="focus_field"/>`

Set focus	La fonction active l'élément de commande spécifiée pour la saisie.
	Le nom de l'élément de commande doit être transmis sous forme d'expression de texte de la fonction.
	
	Syntaxe :
	`<function name="control.setfocus"> control name`
	`</function>`
	
	Attribut :
	name - Nom de la fonction
	
	Valeur :
	control name – Nom du contrôle
	
	Exemple :
	`<function name="control.setfocus" ">_T"listbox1"`
	`</function>`

Get cursor selection	Pour un menu déroulant, la fonction retourne l'indice de curseur.
	Le nom de l'élément de commande doit être transmis sous forme d'expression de texte de la fonction.
	
	Syntaxe :
	`<function name="control.getcurssel" return="var">control name`
	`</function>`
	
	Exemple :
	`<let name>="index"></let>`
	`<function name="control.getcurssel" return="index">_T"listbox1"`
	`</function>`

Nom de la fonction	Signification

Set cursor selection	Pour un menu déroulant, la fonction place le curseur sur la ligne correspondante.
	Le nom de l'élément de commande doit être transmis sous forme d'expression de texte de la fonction.
	
	Syntaxe :
	`<function name="control.setcurssel" > control name, index </function>`
	
	Exemple :
	`<let name>="index">2</let>`
	`<function name="control.setcurssel" ">_T"listbox1",index </function>`

Get Item	Pour un menu déroulant, la fonction copie le contenu de la ligne sélectionnée dans la variable spécifiée.
	Une variable de référence doit être spécifiée en tant que variable String.
	Le nom de l'élément de commande doit être transmis sous forme d'expression de texte de la fonction.
	
	Syntaxe :
	`<function name="control.getitem" return="var"> control name, index </function>`
	
	Exemple :
	`<let name>="index">2</let>`
	`<let name>="item" type="string"></let>`
	
	`<function name="control.getitem" return="item" ">_T"listbox1",index </function>`

Get Item Data	Pour un menu déroulant, la fonction copie la valeur attribuée de manière spécifique à l'utilisateur dans la variable spécifiée.
	Dans le cas d'un élément de commande Edit, la fonction copie la valeur attribuée de manière spécifique à l'utilisateur (item_data) dans la variable spécifiée.
	Une variable de référence doit être spécifiée en tant que variable Integer.
	Le nom de l'élément de commande doit être transmis sous forme d'expression de texte de la fonction.
	
	Syntaxe :
	`<function name="control.getitemdata" return="var"> control name, index </function>`
	
	Exemple :
	`<let name>="index">2</let>`
	`<let name>="itemdata"></let>`
	
	`<function name="control.getitemdata" return="itemdata" ">_T"listbox1",index</function>`

Nom de la fonction	Signification
Image Box Set	Cette fonction sert à commander la zone visible des éléments de commande **Imagebox** et **Graphicbox**. Comme paramètres d'appel, il convient d'indiquer le nom de l'élément de commande, la commande ainsi que les valeurs correspondantes. **Syntaxe :** `<function name="control.imageboxset">control name, command, command parameter</function>` **Paramètres d'appel :** • **x_offs** La fonction déplace le détail à la position X indiquée. Il convient d'indiquer comme paramètre supplémentaire les coordonnées du bord gauche. • **y_offs** La fonction déplace le détail à la position Y indiquée. Il convient d'indiquer comme paramètre supplémentaire les coordonnées du bord supérieur. • **xy_offs** La fonction déplace le détail sur les positions X/Y indiquées. Il convient d'indiquer comme paramètres supplémentaires les coordonnées du bord gauche et du bord supérieur. • **followCursor** Le détail suit la position définie du curseur. • **zoomplus** L'image est agrandie d'un facteur de 0,12. • **zoomminus** L'image est réduite d'un facteur de 0,12. • **autozoom** L'image est automatiquement mise à l'échelle dans la zone d'affichage. • **Update** L'élément de commande est redessiné. • **SetBkColor** La commande définit la couleur d'arrière-plan spécifiée. Comme autre paramètre, il convient d'indiquer le codage couleur. • **SetCursorRect** Le détail indiqué dans le rectangle est déplacé dans la zone visible. • **SetAnimationState** (valable uniquement pour l'élément de commande Imagebox) – **start** - La commande lance une animation. – **stop** - La commande met fin à une animation.

Nom de la fonction	Signification

Image Box Set Suite	**Exemple :**
	`...`
	`...`
	`<softkey POSITION="1">`
	`<caption>zoom%nin</caption>`
	`<function name="control.imageboxset">_T"c_gbox",`
	`_T"zoomplus"</function>`
	`</softkey>`
	`...`
	`...`
	`<form name = "main_form">`
	`<init>`
	`<caption> graphicbox</caption>`
	`<control name= "c_gbox" xpos = "6" ypos="32"`
	`width="328" height="356" fieldtype="graphicbox" />`
	`<let name="plot_list" type="string" />`
	`<op>`
	`plot_list = _T"1;10;20;40;20\\n";`
	`plot_list = plot_list + _T"1;40;20;40;40\\n";`
	`plot_list = plot_list + _T"1;40;40;60;40\\n";`
	`plot_list = plot_list +_T"1;80;0;80;100\\n";`
	`</op>`
	`<function name="control.loaditem">_T"c_gbox",`
	`plot_list</function>`
	`</init>`
	`</form>`

Image Box Get	Cette fonction sert à consulter les propriétés de l'élément de commande Imagebox. Comme paramètres d'appel, il faut fournir la commande ainsi que les valeurs correspondantes.
	
	Syntaxe :
	`<function name="control.imageboxget">control name, command,`
	`command parameter</function>`
	
	Paramètres d'appel :
	GetViewSize
	Fournit la taille de la zone d'affichage
	Comme autre paramètre, il convient d'indiquer une variable du type de structure SIZE.
	
	Exemple :
	`<let name="view_size" type="size" />`
	`<function name="control.imageboxget">_T"topoview", _T"`**`GetViewSize`**`",`
	`view_size</function>`

Nom de la fonction	Signification

Bitmap Dim	La fonction recopie la dimension d'un bitmap dans une variable du type de structure SIZE. Pour la définition de type, il convient d'intégrer le fichier struct_def.xml dans le projet.
	Plus d'informations, voir chapitre "Créer un fichier struct_def.xml (Page 142)"
	
	Syntaxe :
	`<function name="bitmap.dim" >name, variable type`
	`</function>`
	
	Paramètres :
	name - Chemin d'accès
	variable type - Nom de variable d'une variable de type SIZE
	
	Exemple :
	`<let name="bmp_size" type="size" />`
	
	`<function name="bitmap.dim" >_T"test.bmp", bmp_size`
	`</function>`

Screen Resolution	La fonction recopie la résolution d'écran absolue du système dans une variable du type de structure SIZE. Pour la définition de type, il convient d'intégrer le fichier struct_def.xml dans le projet. **Plus d'informations**, voir chapitre "Créer un fichier struct_def.xml (Page 142)" **Syntaxe :** `<function name="hmi.screen_resolution" >varialble type </function>`
Get IHM Resolution	La fonction recopie la résolution d'écran utilisée par SINUMERIK Operate dans une variable du type de structure SIZE. Pour la définition de type, il convient d'intégrer le fichier struct_def.xml dans le projet. **Plus d'informations**, voir chapitre "Créer un fichier struct_def.xml (Page 142)" **Syntaxe :** `<function name="hmi.get_hmi_resolution" >variable type </function>`
Get Caption Height	La fonction fournit la hauteur de la ligne de titre en pixels. **Syntaxe :** `<function name="hmi.get_caption_height" return="<return var>" />` **Attributs :** **return** - Variable Integer (nombre entier)

Nom de la fonction	Signification

Get Font Families	La fonction fournit une liste des polices installées. Les noms de polices sont répertoriés séparément les uns des autres par un caractère LF. Le nom d'une variable string est transmis comme valeur de retour.
	
	Syntaxe :
	`<function name="HMI.GET_FONT_FAMILIES"`
	`return="< String-Variable>" />`
	
	Exemple :
	`<init>`
	`...`
	`<let name="families" type="string" />`
	`<function name="HMI.GET_FONT_FAMILIES" return="families" />`
	`<control name="lb" xpos="8" ypos="40" width="260" height="140"`
	`fieldtype="listbox">`
	`</control>`
	`<function name="control.loaditem">_T"lb", families</function>`
	`...`
	`<update_controls type="true" />`
	`</init>`

Abs	La fonction retourne la valeur absolue du nombre indiqué.
	
	Syntaxe :
	`<function name="abs" return="var"> value`
	`</function>`

SDEG	La fonction convertit la valeur spécifiée en degrés.
	
	Syntaxe :
	`<function name="sdeg" return="var"> value`
	`</function>`

SRAD	La fonction convertit la valeur spécifiée en radians.
	
	Syntaxe :
	`<function name="srad" return="var"> value`
	`</function>`

SQRT	La fonction calcule la racine carrée de la valeur spécifiée.
	
	Syntaxe :
	`<function name="sqrt" return="var"> value`
	`</function>`

ROUND	La fonction arrondit le nombre transmis au nombre spécifié de chiffres après la virgule. Si aucun nombre de chiffres après la virgule n'est défini, la fonction arrondit en tenant compte du premier chiffre après la virgule.
	
	Syntaxe :
	`<function name="round" return="var"> value, nDecimalPlaces`
	`</function>`

Nom de la fonction	Signification
FLOOR	La fonction retourne la valeur entière la plus grande possible, inférieure ou égale à la valeur transmise. **Syntaxe :** `<function name="floor" return="var"> value </function>`
CEIL	La fonction retourne la valeur entière la plus petite possible, supérieure ou égale à la valeur transmise. **Syntaxe :** `<function name="ceil" return="var"> value </function>`
LOG	La fonction calcule le logarithme de la valeur spécifiée. **Syntaxe :** `<function name="log" return="var"> value </function>`
LOG10	La fonction calcule le logarithme décimal de la valeur spécifiée. **Syntaxe :** `<function name="log10" return="var"> value </function>`
POW	La fonction calcule la valeur "a^b". **Syntaxe :** `<function name="pow" return="var"> a, b </function>`
MIN	La fonction compare les valeurs transmises et retourne la valeur la plus petite. **Syntaxe :** `<function name="min" return="var"> value1, value2 </function>`
MAX	La fonction compare les valeurs transmises et retourne la valeur la plus grande. **Syntaxe :** `<function name="max" return="var"> value1, value2 </function>`
RANDOM	La fonction retourne un nombre pseudo-aléatoire. **Syntaxe :** `<function name="random" return="var" </function>`

Nom de la fonction	Signification
MMC	**Fonction** : L'instruction MMC permet d'afficher dans SINUMERIK Operate, à partir du programme pièce, les masques de dialogue définis par l'utilisateur. L'apparence des masques de dialogue est déterminée par une configuration purement textuelle (fichier XML dans le répertoire constructeur), le logiciel système restant inchangé. Les modifications dans le système de base d'Operate transforment les paramètres comme suit : XML → CYCLES ou POPUPDLG XML_ON → PICTURE_ON XML_OFF → PICTURE_OFF **Syntaxe :** MMC ("groupe fonctionnel, instruction, fichier de script XML, nom de menu, réservé, réservé, durée d'affichage ou variable d'acquittement, réservé", "mode d'acquittement") **Signification** : • MMC Appel interactif de masques de dialogue à partir du programme pièce dans SINUMERIK Operate. • CYCLES ou POPUPDLG Identification pour boîtes de dialogue utilisateur devant être démarrées à partir d'un programme pièce. • PICTURE_ON ou PICTURE_OFF Instruction : Sélection/désélection d'une boîte de dialogue • Fichier XML Nom du fichier de script XML • Menu Nom de la balise de menu qui gère la boîte de dialogue à afficher. • réservé réservé pour SINUMERIK Operate • réservé réservé pour SINUMERIK Operate • Durée Durée d'affichage de la boîte de dialogue pour le mode d'acquittement "N" • réservé réservé pour SINUMERIK Operate • Mode d'acquittement "S" synchrone, acquittement via la touche logicielle "OK" "N" asynchrone, la boîte de dialogue est fermée au bout du temps convenu

Nom de la fonction	Signification

MMC Suite	**Exemple d'appel synchrone :**
	Les modifications dans le système de base d'Operate transforment les paramètres comme suit :
	XML → CYCLES ou POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	Instruction CN
	`MMC("POPUPDLG or CYCLES",PICTURE_ON,mmc_cmd.xml,cmd1,,,,,","S")`
	`Fichier : mmc_cmd.xml`
	
	`<menu name = "cmd1">`
	`<open_form name = "cmd1_form" />`
	`<softkey_ok>`
	`<close_form />`
	`</softkey_ok>`
	`</menu>`
	
	`<form name = "cmd1_form" xpos ="12" ypos="100" width="500" height="240">`
	`<init>`
	`</init>`
	`<paint>`
	`</paint>`
	`</form>`
	
	Exemple d'appel asynchrone (pas d'acquittement attendu) :
	Les modifications dans le système de base d'Operate transforment les paramètres comme suit :
	XML → CYCLES ou POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	Instruction CN
	`MMC("POPUPDLG or CYCLES",PICTURE_ON,mmc_cmd.xml,cmd1,,,10,,","N")`
	
	`Fichier : mmc_cmd.xml <menu name = "cmd1">`
	
	`<open_form name = "cmd1_form" />  <softkey_ok>  <close_form />`
	`</softkey_ok>`
	`</menu>`
	
	`<form name = "cmd1_form" xpos ="12" ypos="100" width="500" height="240">`
	`<init>`
	`</init>`
	`<paint>`
	`</paint>`
	`</form>`

Nom de la fonction	Signification

MMC Suite	**Exemple d'extraction de parties de script d'un programme pièce :**
	Les modifications dans le système de base d'Operate transforment les paramètres comme suit :
	XML → CYCLES ou POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	Instruction CN
	`MMC("POPUPDLG, PICTURE_ON, XMLDIAL_EMB.XML,main","S")`
	`Fichier : xmldial_emb.xml`
	
	`<DialogGui>`
	` <let name="menu_name" type="string">main</let>`
	` <let name="script_loaded">0</let>`
	` <menu name = "main">`
	` <if>`
	` <condition>script_loaded == 0</condition>`
	`  <then>`
	` <function name ="load_current_program" />`
	` <dynamic_include src="__tmp.xml" />`
	`  <op>`
	`  script_loaded = 1;`
	`  </op>`
	` <navigation>$$$menu_name</navigation>`
	` </then>`
	` </if>`
	` <softkey_back>`
	` <close_form />`
	` </softkey_back>`
	` </menu>`
	` <function_body name ="load_current_program" >`
	` <let name="prog_name" type= "string"/>`
	` <let name="contents" type= "string"/>`
	` <let name="entry" type= "string"/>`
	` <let name="len" />`
	` <op> prog_name = "nck/Channel/ProgramInfo/workPandProgName"</op>`
	` <function name="DOC.LOADSCRIPT"`
	`return="contents">prog_name, _T"main_dialog", entry`
	`</function>`
	` <function name="string.length" return="len">contents</function>`
	` <if>`
	` <condition>len > 0</condition>`
	` <then>`
	`  <op>`
	`  menu_name = entry;`
	`  </op>`
	` <function name="doc.writetofile" >_T"__tmp.xml", contents`
	`</function>`
	` </then>`
	` </if>`
	` </function_body>`
	`</DialogGui>`

Nom de la fonction	Signification
MMC Suite	Exemple de programme ;<main_dialog entry="rpara_main"> ; <let name="xpos" /> ; <let name="ypos" /> ; <let name="field_name" type="string" /> ; <let name="num" /> ; <menu name="rpara_main"> ; <open_form name="rpara_form"/> ; <softkey_back> ; <close_form /> ; </softkey_back> ;</menu> ; ;<form name="rpara_form"> ; <init> ; <caption>test mask</caption> ; <let name="count" >0</let> ; <op> ; xpos = 120; ; ypos = 34; ; ; "nck/Channel/Parameter/R[10]" = 10; ; </op> ; <!-- load the number of controls --> ; <op> ; num = "nck/Channel/Parameter/R[10]"; ; </op> ; ; <while> ; <condition>count < num</condition> ; <print name="field_name" text="edit%d">count</print> ; ; <control name = "$field_name" xpos = "$xpos" ypos = "$ypos" refvar="nck/Channel/Parameter/R[$count]" hotlink="true" /> ; <op> ; ypos = ypos +24; ; count = count +1; ; </op> ; </while> ; ; </init> ; ; <paint> ; <op> ; xpos = 8; ; ypos = 36; ; count = 0; ; </op> ; <while> ; <condition>count < num</condition> ; <print name="field_name" text="R-Parameter%d">count </print> ; ; <text xpos = "$xpos" ypos = "$ypos" >$$$field_name</text> ; <op>

Nom de la fonction	Signification
	; ypos = ypos +24; ; count = count +1; ; </op> ; </while> ; ; </paint> ; ;</form> ; ;</main_dialog> … G94 F100 MMC("POPUPDLG, PICTURE_ON, XMLDIAL_EMB.XML,main","A") … … … MMC("POPUPDLG, PICTURE_OFF, XMLDIAL_EMB.XML,main","A") G4 F2 M2
ISNUMERIC	La fonction vérifie le contenu d'une chaîne de caractères pour voir si elle peut être considérée comme un nombre. Les espaces et les tabulations sont ignorées. Le signe et la virgule décimale sont également des caractères valides. La fonction fournit 1 comme valeur de retour si les conditions sont remplies. **Paramètres :** **string** - Chaîne de caractères **Syntaxe :** <function name="isnumeric" return="result"><string> </function>
ROOT	La fonction extrait la nième racine d'un nombre. **Paramètres :** **value** - Nombre **n** - Exposant de la racine **Syntaxe :** <function name="ROOT" return="result"><value>, <n></function>

3.12 Commande multitactile

3.12.1 Fonction Multitouch

Vue d'ensemble

La mise en œuvre d'écrans Multitouch nécessite l'adaptation de la commande à la fonctionnalité étendue des écrans. Par exemple, la disposition fixe des touches logicielles disparaît et est remplacée par une disposition libre des touches, ou bien elle la complète. Du fait de la multiplicité des possibilités de disposition des touches, il n'est plus pertinent de n'utiliser, pour la programmation, qu'un seul contrôle dont l'aspect ne correspond qu'au design du logiciel de commande. Les commandes à bascule, qui ne reconnaissent que deux états, peuvent par ex. être remplacées par des commutateurs représentant chaque état par une icône et réagissant à une pression ou un balayage du doigt.

Les graphiques affichés peuvent être mis en capacité de réagir à un simple mouvement de doigt. Pour ce faire, la commande **imagebox** est étendue en conséquence.

Remarque

Les graphiques créés avec la balise Paint ne réagissent pas aux gestes.

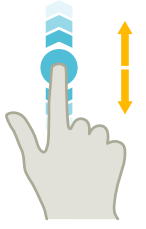
En dehors des commandes fournies par l'analyseur syntaxique, les gestes du doigt peuvent être traités dans le script, notamment pour permettre de proposer de nouvelles stratégies de navigation dans les boîtes de dialogue. Les gestes du doigt peuvent être utilisés pour agrandir/réduire le contenu des boîtes de dialogue.

L'analyseur syntaxique Easy XML prend en charge les gestes du doigt suivants :

Gestes

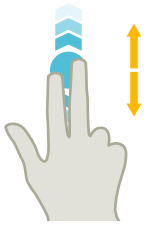
Toucher (Tap)

- Sélectionner une fenêtre
- Sélectionner un objet (par ex. bloc CN)
- Activer un champ de saisie
  - Saisir ou écraser une valeur
  - Toucher une nouvelle fois pour modifier la valeur

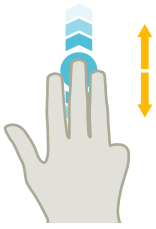


Balayer verticalement avec 1 doigt (Flick)

- Faire défiler une liste (par ex. programmes, outils, origines)
- Faire défiler un fichier (par ex. programme CN)



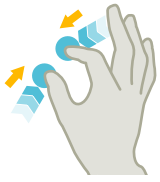
Balayer verticalement avec 2 doigts (Flick)
- Faire défiler une liste page par page (par ex. NPV)
- Faire défiler un fichier page par page (par ex. programmes CN)



Balayer verticalement avec 3 doigts (Flick)
- Faire défiler jusqu'au début ou la fin d'une liste
- Faire défiler jusqu'au début ou la fin d'un fichier

Balayer horizontalement avec 1 doigt (Flick)
- Faire défiler une liste comportant de nombreuses colonnes

Étirer (Spread)
- Agrandir des contenus graphiques (par ex. simulation, vue moulage)



Pincer (Pinch)
- Réduire des contenus graphiques (par ex. simulation, vue moulage)

Déplacer avec 1 doigt (Pan)
- Déplacer des contenus graphiques (par ex. simulation, vue moulage)
- Faire glisser les contenus d'une liste

Pour gérer les gestes du doigt, la balise **gesture_event** et la variable **$gestureinfo** sont utilisées. Elles permettent des actions de programme pour les gestes du doigt décodés.

3.12.2 Programmation des gestes du doigt

Balise gesture_event

Remarque

Cette balise n'est exécutée que si le tableau de commande prend en charge la commande Multitouch.

Si le logiciel de commande reconnaît un geste du doigt, l'analyseur syntaxique fournit les informations sur les gestes dans les variables **$gestureinfo** et exécute la balise **gesture_event**. La variable a le type de données **GestureInfoStruct** et contient les attributs suivants :

Attribut	Valeur	Signification
type	3	Geste Déplacer
	4	Geste Réduire
	5	Geste Toucher
flag	1	Facteur d'échelle modifié
	2	Angle de rotation modifié
state	1	Démarré
	2	Mis à jour
	3	Terminé
item_data		Identification de la commande active
	-1	Le geste n'est affecté à aucune commande
	!= -1	Le geste a été exécuté dans une commande à laquelle cette valeur a été affectée lors de sa création
point		Position du geste
	point.x	Position X du geste
	point.y	Position Y du geste
delta		Écart entre le début du geste et l'événement actuel
	<écart d'échelle>	En cas de modification du facteur d'échelle
	<écart angulaire>	En cas de modification de l'angle de rotation

Les attributs sont prédéfinis dans le fichier **struct_def.xml**.

Pour plus d'informations, voir chapitre "Créer un fichier struct_def.xml (Page 142)"

3.12.3 Commandes gestuelles pour graphiques

Variable de commande imagebox

Pour la commande gestuelle des graphiques, il existe les trois extensions suivantes pour la variable de commande **Imagebox** :

Attribut	Signification/Comportement
rotationangle	La valeur d'attribut indique l'angle selon lequel le graphique doit être représenté.
setrotationmode	La valeur d'attribut **true** permet de traiter le geste Pincer
setzoommode	La valeur d'attribut **true** permet d'agrandir/de réduire et de déplacer un graphique dans la zone d'affichage. Par défaut, cet attribut a la valeur **false**.

Syntaxe

```
<property rotationangle="Winkel" />
<property setrotationmode="true/false" />
<property setzoommode="true/false" />
```

Exemple de rotation de bitmap

Tourner un bitmap de 30,5 degrés dans le sens horaire :

```
<let name="image_box_pict_name" type="string" >pic1.bmp</let>
…
…
…
 <control name="image_view" xpos="100" ypos="123" width="300"
height="200" fieldtype="imagebox" refvar="image_box_pict_name"
hotlink="true" >
 <property item_data="1000" />
 <property rotationangle="30.5" />
 <property setrotationmode ="true" />
 </control>
```

Exemple d'agrandissement de bitmap

Autoriser l'agrandissement d'un bitmap :

```
<control name="image_view" xpos="100" ypos="123" width="300"
height="200" fieldtype="imagebox" refvar="image_box_pict_name"
hotlink="true" >
 <property item_data="1000" />
 <property setzoommode="true" />
 </control>
```

Fonction de commande imageboxset

De plus, les propriétés sont définies via la fonction **control.imageboxset**.

Les commandes suivantes sont disponibles :

Commande	**Signification/Comportement**
SetRotationAngle	Le graphique est pivoté selon l'angle indiqué dans **lparam1**.
SetRotationMode	La valeur de **lparam1** définit la valeur du pincement par rapport à la rotation. Valeur égale à 1 - le geste est traité par la commande
SetZoomMode	Si la valeur de **lparam1** est égale à 1, le pincement est traité pour l'agrandissement/la réduction ainsi que le déplacement du graphique.

3.12.4 Traitement des gestes

Balise message

Si le facteur d'échelle est supérieur à 1,0, l'image est déplacée dans la zone d'affichage. Pour un facteur de 1,0, ce geste peut être utilisé, par ex., pour faire défiler une liste d'images. Dans ce cas, l'analyseur syntaxique envoie un message sous une forme qui peut être exploitée dans la balise **message**.

Le paramètre de message **$message_par1** contient la valeur Item_Data de la commande. Le paramètre de message **$message_par2** signale le sens du mouvement du geste.

Le **$message_par2** suivant peut prendre les valeurs suivantes :

- -1 défilement vers la gauche
- 1 défilement vers la droite
- -2 défilement vers le haut
- 2 défilement vers le bas

Exemple

```
…
…
<let name="image_box_pict_name" type="string" ></let>
<let name="pictindex" />
<let name="image_box_list" type="string" dim="4">
 "pic1.bmp",
 "pic2.bmp",
 "pic3.bmp",
 "pic4.bmp",
</let>

…
…
<control name="image_view" xpos="100" ypos="123" width="300"
height="200" fieldtype="imagebox" refvar="image_box_pict_name"
hotlink="true" >
 <property item_data="1000" />
 <property setzoommode="false" />
</control>
…
…
…
<!--
 -1 défilement vers la gauche
1 défilement vers la droite
-2 défilement vers le haut
2 défilement vers le bas
 -->
<message>
 <if condition="$message_par1 == 1000">
```

```
  <then>

   <switch>
    <condition>$message_par2</condition>
    <case value="1">
     <op>
      pictindex = pictindex+1;
     </op>
     <if condition="pictindex > 3">
      <then>
       <op>
        pictindex = 0;
       </op>
      </then>
     </if>
    </case>
    <case value="-1">
     <op>
      pictindex = pictindex-1;
     </op>
     <if condition="pictindex < 0">
      <then>
       <op>
        pictindex = 3;
       </op>
      </then>
     </if>
    </case>
   </switch>
    <op>
     image_box_pict_name = image_box_list[$pictindex];
    </op>
   </then>
  </if>
 </message>
…
…
```

3.13 Configuration personnalisée des boutons

Les boutons peuvent être intégrés dans un formulaire en tant que touches d'action ou touches de sélection.

3.13.1 Bouton-poussoir

Balise property

Le bouton-poussoir est un élément de commande qui peut être utilisé en tant que touche ou commutateur (touche avec fonction de verrouillage). Le bouton peut être configuré, selon les définitions d'attributs, selon le style "Softkey Look & Feel" ou selon un style propre à l'utilisateur. Les attributs correspondants doivent être définis avec la balise **property**.

La modification des couleurs de premier plan et d'arrière-plan est possible avec les attributs **color_fg**, **color_bk**, **color_fg_pressed** et **color_bk_pressed**.

Les états **enfoncé/verrouillé** ou **relâché** sont représentés par les valeurs Un ou Zéro. Pour mettre en œuvre la modification d'état d'une touche, l'indication d'une fonction de traitement est requise. L'indication de la fonction de traitement s'effectue avec l'attribut **function** dans la balise Control.

L'état d'un commutateur peut être déterminé par la lecture des variables Control ou d'une variable de référence affectée.

Pour une commande tactile, la notification du geste de doigt "enfoncé" et "relâché" n'est émise qu'une fois le geste du doigt "relâché" terminé.

Syntaxe

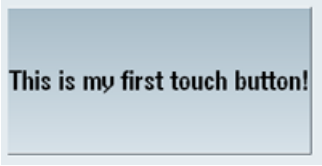
```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" >
 <caption></caption>
</control>
```

Ou avec fonction de traitement

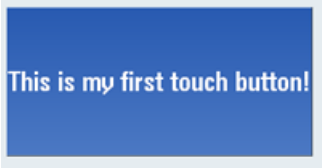
```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" function="button_handler" hotlink="true" >
 <caption></caption>
</control>
…
…
…
 <function_body name="button_handler">
…
…
…
 </function_body>
```

Ou

```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" refvar="button_state" hotlink="true" >
 <caption></caption>

 <property checkable="true" />
</control>
```



désactivé



activé, verrouillé

Exemple

```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" refvar="button_state" hotlink="true"
color_fg="#ff00ff" color_bk="#000eee">
 <property checkable="true" />
  <caption></caption>
 <property picture="sk_circ_grind_cg.png" alignment="left"
TextAlignedToPicture="true" scaled="true"/>
</control>
```

3.13.2 Fonctions du bouton-poussoir

3.13.2.1 Sous-balises pour le bouton-poussoir

Balise caption

Avec la balise **caption**, le programmeur définit le texte du bouton à afficher pour l'état non enfoncé. Par défaut, le texte est centré. Le sens du texte peut être modifié avec l'attribut **alignment**. Les valeurs d'attribut suivantes peuvent être indiquées :

Valeur d'attribut	Alignement
left	justifié à gauche
right	justifié à droite
top	en haut
bottom	en bas
center	centré

Si un autre texte doit être affiché pour l'état enfoncé, une deuxième instruction caption doit être programmée avec l'attribut **pressed** et la valeur **true**.

Syntaxe

Affichage centré

```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" >
 <caption>Button text</caption>
</control>
```

Affichage aligné à gauche

```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" >
 <caption alignment= "left">Button text</caption>
</control>
```

Affichage enfoncé

```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" >
 <caption pressed="true">Button text pressed</caption>

</control>
```

3.13.2.2 Propriétés du bouton-poussoir

Des propriétés supplémentaires sont affectées à la commande avec la balise **property**.

Définir la propriété de la touche

L'attribut **checkable** permet d'utiliser le bouton en tant que commutateur. Pour cela, la valeur de l'attribut doit être définie sur **true**.

Syntaxe

```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" >
 <property checkable="true/false" />
</control>
```

Verrouiller la touche

L'attribut **disabled** commande l'opérabilité du bouton. Si la valeur est **true**, les manipulations ne sont pas traitées.

Les changements d'état appelés par une variable de référence affectée se traduisent par la mise à jour du bouton.

Syntaxe

```
<property disabled="true/false" />
```

Exemple

```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" >
 <caption alignment="left">Button text</caption>
 <property disabled="true " />
</control>
```

Affecter des icônes

Le design prédéfini peut être adapté en spécifiant vos propres icônes ou couleurs de boutons. Pour les états "non enfoncé", "enfoncé" et "verrouillé", il est possible d'affecter une icône correspondante au bouton. En l'absence d'affectation pour les états "enfoncé" ou "verrouillé", la commande indique l'icône pour l'état "non enfoncé".

D'autres attributs commandent le sens, l'échelle et la transparence de chaque icône.

Attribut	Signification/Comportement
Picture	Icône au premier plan Nom de l'icône pour l'état "non enfoncé"
PicturePressed	Icône au premier plan Nom de l'icône pour l'état "enfoncé"
PictureDisabled	Icône au premier plan Nom de l'icône pour l'état "verrouillé"

Attribut	Signification/Comportement
BackgroundPicture	Icône à l'arrière-plan Nom de l'icône pour l'état "non enfoncé"
BackgroundPicturePressed	Icône à l'arrière-plan Nom de l'icône pour l'état "enfoncé"
BackgroundPictureDisabled	Icône à l'arrière-plan Nom de l'icône pour l'état "verrouillé"

Attribut alignment

Dans la balise peuvent être indiqués d'autres attributs dont les valeurs se rapportent à l'**alignement** des affectations d'icônes présentées :

Valeur d'attribut	Alignement
left	justifié à gauche
right	justifié à droite
top	en haut
bottom	en bas
center	centré
stretch	Icône en arrière-plan : Si la valeur **stretch** est utilisée, l'analyseur syntaxique met l'icône à l'échelle de la zone carrée du bouton. Le contour souhaité pour le bouton peut être créé en faisant apparaître la couleur transparente avec l'attribut **transparent**.

3.13.2.3 Variables de commande pour le bouton-poussoir

Il est possible de modifier ultérieurement les propriétés du bouton en affectant de nouvelles valeurs aux variables d'attributs ci-dessous. L'affectation intervient via une instruction d'opération par le biais de l'indication du nom de la commande, suivi du nom de la variable de commande. Les deux noms doivent être séparés par un point.

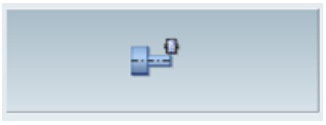
Syntaxe

<Control-Name>.<Control-Variable>

Variable de commande	Signification/Comportement
backgroundpicture	Type de variable : valeur du type string Nom de l'icône en arrière-plan
backgroundpicturedisabled	Type de variable : valeur du type string Nom de l'icône en arrière-plan pour l'état verrouillé
backgroundpicturepressed	Type de variable : valeur du type string Nom de l'icône en arrière-plan pour l'état enfoncé
picture	Type de variable : valeur du type string Nom de l'icône au premier plan
picturedisabled	Type de variable : valeur du type string Nom de l'icône au premier plan pour l'état verrouillé
picturepressed	Type de variable : valeur du type string Nom de l'icône au premier plan pour l'état enfoncé
picture_textalignedtopicture	Type de variable : Bool Valeur : 1 = true 0 = false Le texte du bouton est aligné sur l'icône
picturedisabled_textalignedtopicture	Type de variable : Bool Valeur : 1 = true 0 = false Le texte du bouton est aligné sur l'icône "état verrouillé"
picturepressed_textalignedtopicture	Type de variable : Bool Valeur : 1 = true 0 = false Le texte du bouton est aligné sur l'icône "état enfoncé"

Variable de commande	Signification/Comportement
backgroundpicture_keepaspectratio	Type de variable : Bool Valeur : 1 = true 0 = false Le rapport largeur/hauteur de l'icône en arrière-plan est conservé ou ignoré
backgroundpicturedisabled_keepaspectratio	Type de variable : Bool Valeur : 1 = true 0 = false Le rapport largeur/hauteur de l'icône en arrière-plan "état verrouillé" est conservé ou ignoré
backgroundpicturepressed_keepaspectratio	Type de variable : Bool Valeur : 1 = true 0 = false Le rapport largeur/hauteur de l'icône en arrière-plan "état enfoncé" est conservé ou ignoré
picture_keepaspectratio	Type de variable : Bool Valeur : 1 = true 0 = false Le rapport largeur/hauteur de l'icône est conservé ou ignoré
picturedisabled_keepaspectratio	Type de variable : Bool Valeur : 1 = true 0 = false Le rapport largeur/hauteur de l'icône "état verrouillé" est conservé ou ignoré
picturepressed_keepaspectratio	Type de variable : Bool Valeur : 1 = true 0 = false Le rapport largeur/hauteur de l'icône "état enfoncé" est conservé ou ignoré
backgroundpicture_scaled	Type de variable : Bool Valeur : 1 = true 0 = false L'icône en arrière-plan est adaptée aux dimensions du bouton

Variable de commande	Signification/Comportement
backgroundpicturedisabled_scaled	Type de variable : Bool Valeur : 1 = true 0 = false L'icône en arrière-plan "état verrouillé" est adaptée aux dimensions du bouton
backgroundpicturepressed_scaled	Type de variable : Bool Valeur : 1 = true 0 = false L'icône en arrière-plan "état enfoncé" est adaptée aux dimensions du bouton
picture_scaled	Type de variable : Bool Valeur : 1 = true 0 = false L'icône est adaptée aux dimensions du bouton
picturedisabled_scaled	Type de variable : Bool Valeur : 1 = true 0 = false L'icône "état verrouillé" est adaptée aux dimensions du bouton
picturepressed_scaled	Type de variable : Bool L'icône "état enfoncé" est adaptée aux dimensions du bouton
backpicture_alignment	Type de variable : valeur du type string Voir l'attribut alignment
backgroundpicturedisabled_alignment	
backgroundpicturepressed_alignment	
picture_alignment	
picturedisabled_alignment	
picturepressed_alignment	
caption	Type de variable : valeur du type string Texte du bouton "état non enfoncé"
captionpressed	Type de variable : valeur du type string Texte du bouton "état enfoncé"
caption_alignment	Type de variable : valeur du type string Voir l'attribut alignment
captionpressed_alignment	
captiondisabled_alignment	
disabled	Type de variable : Bool Valeur : 1 = true - bouton-poussoir verrouillé 0 = false - bouton-poussoir actionnable



Picture

PicturePressed

PictureDisabled

BackgroundPicture + Picture

Attribut TextAlignedToPicture

Si la valeur de l'attribut est **true**, le texte des icônes est orienté en conséquence.

```
<property picture="sk_circ_grind_cg.png" alignment="left"
TextAlignedToPicture="true" />
```

Attribut scaled

Si la valeur de l'attribut est **true**, les dimensions de l'image sont adaptées aux dimensions du bouton de manière à ce que 50 % au maximum de la hauteur ou de la largeur du bouton du graphique soient disponibles.

```
<property picture="sk_circ_grind_cg.png" alignment="left"
TextAlignedToPicture="true" scaled="true"/>
```

Attribut stretch (seulement actif pour les icônes en arrière-plan)

Si la valeur de l'attribut est **true**, les dimensions de l'image sont adaptées aux dimensions du bouton.

```
<property backgroundpicture="f:\appl\pbutton.png"
alignment="stretch" />
```

```
<property backgroundpicture="f:\appl\pbutton.png"
alignment="stretch" transparent="#ffffff"/>
```

```
<caption alignment="left" >This is my first touch button!</caption>
 <property picture="sk_circ_grind_cg.png" alignment="left"
TextAlignedToPicture="true" scaled="true"/>
 <property backgroundpicture="pbutton.png" alignment="stretch"
transparent="#ffffff"/>
```

3.13.3 Commutateur on/off

Une commande de commutation est un élément graphique indiquant un état sur deux par une icône. Cette commande est actionnable par voie tactile ou à l'aide de la souris.

L'état sélectionné peut être enregistrée dans une variable de référence ou le changement d'état peut être déterminé dans une des fonctions affectées à la commande. L'affectation s'effectue avec l'attribut **function** dans la balise control.

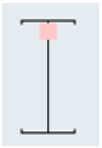
L'attribut **alignment** permet l'alignement vertical ou horizontal du bouton.

Cette commande n'a pas de design par défaut. Si aucune icône n'est affectée à la commande, seuls le cadre et la position du commutateur (représentée par un point) s'affichent.

Syntaxe

```
<control name="name" xpos = "x position" ypos="y position"
width="width" height="height" fieldtype="switch" alignment="hr/vr">
 <property left_position="value" />
 <property right_position="value" />
</control>
```

Attribut alignment



Commutateur vertical : Valeur **vr**

Commutateur horizontal : Valeur **hr**

3.13.4 Fonctions du commutateur

3.13.4.1 Propriétés du commutateur

Balise property

Les positions de commutateur suivantes peuvent être affectées à la commande avec la balise **property**.

Attribut	Signification/Comportement
left_position	La valeur de l'attribut est affectée à la variable control lorsque le commutateur est déplacé vers la gauche.
right_position	La valeur de l'attribut est affectée à la variable control lorsque le commutateur est déplacé vers la droite.
upper_position	La valeur de l'attribut est affectée à la variable control lorsque le commutateur est déplacé vers le haut.
lower_position	La valeur de l'attribut est affectée à la variable control lorsque le commutateur est déplacé vers le bas.
disabled	Cet attribut commande l'opérabilité du commutateur. Si la valeur est **true**, aucune manipulation n'est traitée. Les changements d'état appelés par une variable de référence affectée se traduisent par la mise à jour de l'état du commutateur.

Le design final du commutateur doit être défini en spécifiant d'autres attributs. Ces attributs doivent être définis avec les attributs **left_position**, **right_position**, **upper_position** ou **lower_position**.

Le contour souhaité pour le commutateur peut être créé en faisant apparaître une couleur comme transparente avec l'attribut **transparent**.

Attribut	Signification/Comportement
picture	Nom de l'icône au premier plan pour l'état "non enfoncé"
pictureDisabled	Nom de l'icône au premier plan pour l'état "verrouillé"

Attribut	Signification/Comportement
transparent	La valeur d'attribut contient la valeur de la couleur transparente. Cette valeur de couleur est utilisée pour toutes les icônes affectées au commutateur.
caption	La valeur d'attribut contient le texte correspondant à la position du commutateur. Ce texte est représenté sur l'élément et limité par les dimensions du commutateur.

3.13.4.2 Variables control pour le commutateur

Il est possible de modifier ultérieurement les propriétés du commutateur en affectant de nouvelles valeurs aux variables control ci-dessous. L'affectation intervient via une instruction d'opération par le biais de l'indication du nom de la commande, suivi du nom de la variable de commande. Les deux noms doivent être séparés par un point.

Syntaxe

<Control-Name>.<Control-Variable>

Variable de commande	Signification/Comportement
Left_position	Type de variable : Int La valeur de l'attribut est affectée à la variable control lorsque le commutateur est déplacé vers la gauche.
Right_position	Type de variable : Int La valeur de l'attribut est affectée à la variable control lorsque le commutateur est déplacé vers la droite.
Lower_position	Type de variable : Int La valeur de l'attribut est affectée à la variable control lorsque le commutateur est déplacé vers le bas.
Upper_position	Type de variable : Int La valeur de l'attribut est affectée à la variable control lorsque le commutateur est déplacé vers le haut.
Picture_left_position	Type de variable : valeur du type string Nom de l'icône pour la position à gauche
Picture_right_position	Type de variable : valeur du type string Nom de l'icône pour la position à droite
Picture_upper_position	Type de variable : valeur du type string Nom de l'icône pour la position en haut
Picture_lower_position	Type de variable : valeur du type string Nom de l'icône pour la position en bas
Picturedisabled_left_position	Type de variable : valeur du type string Nom de l'icône pour la position à gauche "état verrouillé"
Picturedisabled_right_position	Type de variable : Nom de l'icône pour la position à droite "état verrouillé"
Picturedisabled_upper_position	Type de variable : valeur du type string Nom de l'icône pour la position en haut "état verrouillé"
Picturedisabled_lower_position	Type de variable : valeur du type string Nom de l'icône pour la position en bas "état verrouillé"

Variable de commande	Signification/Comportement
Caption_left_position	Type de variable : valeur du type string Texte en position à gauche
Caption_right_position	Type de variable : valeur du type string Texte en position à droite
Caption_upper_position	Type de variable : valeur du type string Texte en position en haut
Caption_lower_position	Type de variable : valeur du type string Texte en position en bas
disabled	Type de variable : Bool Valeur : 1 = true - commutateur verrouillé 0 = false - commutateur actionnable

Affichage horizontal

```
<control name="name" xpos="x position" ypos="y position"
width="width" height="height" fieldtype="switch" alignment="hr">
 <property left_position="value" picture="name" />
 <property right_position="value" picture="name" />
</control>
```

Affichage vertical

```
<control name="name" xpos="x position" ypos="y position"
width="width" height="height" fieldtype="switch" alignment ="vr">
 <property upper_position="value" picture="name" />
 <property lower_position="value" picture="name" />
</control>
```

Ou affecter fonction de traitement

```
<control name="name" xpos="x position" ypos="y position"
width="width" height="height" fieldtype="switch"
function="switch_handler" hotlink="true" alignment ="vr">
</control>
```

Exemple

icon_left.bmp

icon_right.bmp

```
<let name="switch_state" />
```

```
<control name="main_switch" xpos="100" ypos="53" width="40"
height="30" fieldtype="switch" refvar="switch_state" hotlink="true"
alignment="hr">
 <property left_position="10" picture="icon_left.bmp" />
 <property right_position="11" picture="icon_right.bmp" />
</control>
```

Affecter propriétés via la variable control

```
<control name="main_switch" xpos = "450" ypos="340" width="20"
height="46" fieldtype="switch" refvar="switch_statebf"
hotlink="true" alignment="vr">
 <property upper_position="1" />
 <property lower_position="0" />
</control>
…
…
…
<op>
  main_switch.transparent = #ffffff;
  main_switch.picture_upper_position = _T"switch_up.png";
  main_switch.picture_lower_position = _T"switch_bottom.png";
  main_switch.disable= 1;
</op>
```

3.13.5 Bouton radio

Les boutons radio permettent de sélectionner une ou plusieurs options. Un bouton doit être créé pour chaque option. Tous les boutons radio correspondant à un formulaire, un encadré ou une zone de défilement interagissent les uns avec les autres, de manière à ne pouvoir sélectionner qu'un bouton à la fois. Les états des boutons sont transmis aux variables control.

Syntaxe

```
<control name="name" xpos="x position" ypos="y position"
width="width" height="height" fieldtype="radiobutton"  >
 <caption>button text</caption>
</control>
```

3.13.6 Case à cocher

Les cases à cocher permettent de sélectionner plusieurs options. Un bouton doit être créé pour chaque option. L'état de la case à cocher est transmis aux variables control.

Syntaxe

```
<control name="name" xpos="x position" ypos="y position"
width="width" height="height" fieldtype="checkbox"  >
 <caption>button text</caption>
</control>
```

3.13.7 Encadré

Un encadré contient visuellement un groupe de commandes avec un cadre et un intitulé. Il permet de regrouper des commandes logiquement interdépendantes qui ne doivent interagir que dans le cadre de ce groupe. Les coordonnées des commandes proposées se rapportent au coin supérieur gauche sous la ligne de titre.

Toutes les commandes appartenant au groupe sont proposées en tant que sous-commandes dans la balise control.

Lors de l'attribution du nom de commande et de la valeur **Item_Data** proposés, il convient de veiller à ce que ces éléments soient univoques parmi toutes les commandes du formulaire.

L'intitulé de l'encadré est défini avec la balise **caption**.

Syntaxe

```
<control name="name" xpos="x position" ypos="y position"
width="width" height="height" fieldtype="groupbox">
 <caption>caption text</caption>
  <control>…</control>
…
…
…
</control>
```

3.13.8 Zone de défilement

Une zone de défilement est utilisée pour afficher les commandes au sein d'une zone donnée. Les coordonnées des commandes proposées se rapportent au coin supérieur gauche de la zone de défilement. Si des commandes apparaissent en dehors de la zone visible, il est possible de déplacer celle-ci. Toutes les commandes appartenant à la zone sont proposées en tant que sous-commandes dans la balise control.

Lors de l'attribution du nom de commande et de la valeur **Item_Data** proposés, il convient de veiller à ce que ces éléments soient univoques parmi toutes les commandes du formulaire.

Syntaxe

```
<control name="name" xpos="x position" ypos="y position"
width="width" height="height" fieldtype="scrollarea">
  <control>…</control>
…
…
…
</control>
```

Commande tactile

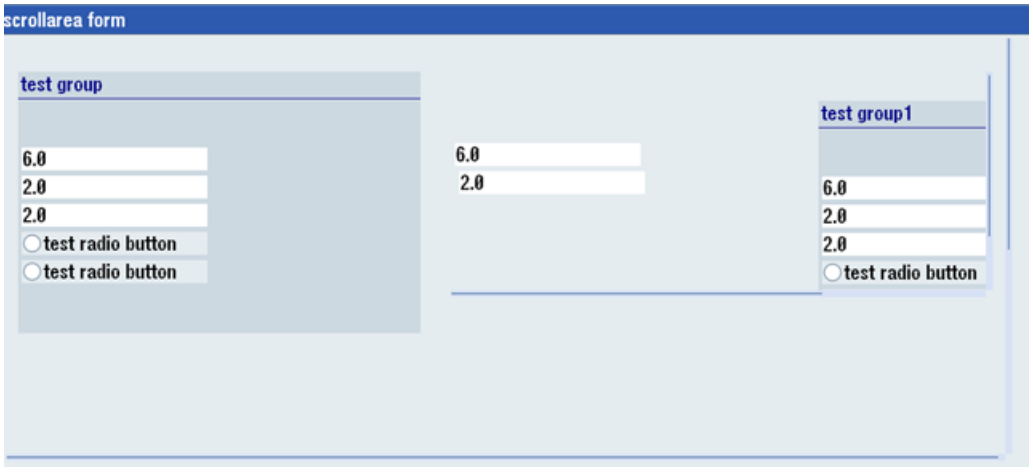
Si la commande en surbrillance n'est pas totalement visible, la zone d'affichage se décale jusqu'à ce que la commande soit complètement visible.

Geste de tabulation

Ce geste est transmis au script quand le geste ne peut être affecté à aucune commande proposée.

Exemple

Zones de défilement imbriquées



```
<let name="CB1" >1</let>
<let name="RB1" />

<form name="scrollarea_form">
 <init>
  <caption>scrollarea form</caption>
   <data_access type="true" />
    <control name= "c_scroll" xpos = "2" ypos="24" width="520"
height="300" fieldtype="scrollarea" >
    <control name= "c_group" xpos = "6" ypos="24" width="208"
height="186" fieldtype="groupbox" >
    <caption>test group</caption>
    <control name = "c18" xpos = "2" ypos = "54" refvar="nck/Channel/
Parameter/R[1]" hotlink="true" />
    <control name = "c19" xpos = "2" ypos = "74" refvar="nck/Channel/
Parameter/R[2]" hotlink="true" />
    <control name = "c20" xpos = "2" ypos = "94" refvar="nck/Channel/
Parameter/R[3]" hotlink="true" />
    <control name="radiobg" xpos = "2" ypos="114"
fieldtype="radiobutton" >
    <caption>test radio button</caption>
    <property backgroundpicture="f:\appl\cycle_my1.png" />
   </control>
   <control name="radiobg1" xpos = "2" ypos="134"
fieldtype="radiobutton" >
    <caption>test radio button</caption>
    <property backgroundpicture="f:\appl\cycle_my1.png" />
   </control>
  </control>
 <control name= "c_scroll_emb" xpos = "230" ypos="24" width="280"
height="160" fieldtype="scrollarea" >

 <control name = "c18emb" xpos = "2" ypos = "54" refvar="nck/Channel/
Parameter/R[1]" hotlink="true" />
 <control name = "c19emb" xpos = "4" ypos = "74" refvar="nck/Channel/
Parameter/R[2]" hotlink="true" />
 <control name = "c20emb" xpos = "3" ypos = "194" refvar="nck/Channel/
Parameter/R[3]" hotlink="true" />
 <control name= "c_group1" xpos = "190" ypos="24" width="208"
height="186" fieldtype="groupbox" >
 <caption>test group1</caption>
 <control name = "c18gemb" xpos = "2" ypos = "54" refvar="nck/Channel/
Parameter/R[1]" hotlink="true" />
 <control name = "c19gemb" xpos = "2" ypos = "74" refvar="nck/Channel/
Parameter/R[2]" hotlink="true" />
 <control name = "c20gemb" xpos = "2" ypos = "94" refvar="nck/Channel/
Parameter/R[3]" hotlink="true" />
 <control name="radiobggemb" xpos = "2" ypos="114"
fieldtype="radiobutton" >
  <caption>test radio button</caption>
  <property backgroundpicture="f:\appl\cycle_my1.png" />
</control>
```

```
<control name="radiobg1emb" xpos = "2" ypos="134"
fieldtype="radiobutton" >
 <caption>test radio button</caption>
  <property backgroundpicture="f:\appl\cycle_my1.png" />
 </control>
 </control>
 </control>
<control name = "c22" xpos = "2" ypos = "400" refvar="nck/Channel/
Parameter/R[12]" hotlink="true" />
 <control name="ChB1" xpos="208" ypos="430" width="50" height="30"
fieldtype="checkbox" refvar="PLC/M100.7" hotlink = "true"
color_bk="#00ffff">
  <caption>Check1</caption>
 </control>
  <control name="ChB2" xpos="208" ypos="460" width="58" height="30"
fieldtype="checkbox" refvar="PLC/M100.6" hotlink = "true"
color_bk="#00ffff">
  <caption>Check2</caption>
 </control>
  <control name="Radio1" xpos="208" ypos="490" width="40"
height="30" fieldtype="radiobutton" refvar="PLC/M100.0" hotlink =
"true" color_bk="#00ffff">
   <caption>Radio1</caption>
  </control>
   <control name="Radio2" xpos="208" ypos="520" width="45"
height="30" fieldtype="radiobutton" refvar="PLC/M100.1" hotlink =
"true" color_bk="#00ffff">
    <caption>Radio2</caption>
  </control>
   <control name="Radio3" xpos="208" ypos="550" width="50"
height="30" fieldtype="radiobutton" refvar="PLC/M100.2" hotlink =
"true" >
    <caption>Radio3</caption>
   </control>
    <control name="VChB1" xpos="8" ypos="430" width="50" height="30"
refvar="PLC/MB100" hotlink = "true" color_bk="#00ffff"/>
    <control name="VChB2" xpos="8" ypos="465" width="53" height="30"
refvar="PLC/MB100" hotlink = "true" color_bk="#00ffff"/>
   </control>
   <control name="ChB3" xpos="208" ypos="340" width="60" height="30"
fieldtype="checkbox" refvar="CB1" >
    <caption>Check3</caption>
   </control>
   </init>
  <timer>
  </timer>
</form>
```

3.14 Application Sidescreen

3.14.1 Easy XML dans Sidescreen

Le langage de script Easy XML peut aussi être utilisé pour créer des boîtes de dialogue dans la zone Sidescreen. Les composants Sidescreen **Page** et **Widget** sont disponibles pour afficher les boîtes de dialogue. L'intégration se fait par le biais du fichier **slsidescreen.ini**. Les composants Sidescreen sont automatiquement lancés au démarrage du logiciel de commande et quittés seulement lors de l'arrêt. En d'autres termes, l'analyseur syntaxique charge les scripts affectés une seule fois lors de la première activation des composants Sidescreen.

Les dimensions de la zone Sidescreen sont définies par les paramètres de représentation figurant dans le fichier **slsidescreen_<Résolution d'écran>.ini**. Pour permettre une configuration indépendante, la largeur utile est de 276 pixels pour les scripts Easy XML. La hauteur utile d'une boîte de dialogue Easy XML dépend du composant Sidescreen utilisé. 768 pixels sont disponibles pour une page. Pour un widget, la hauteur est définie par la gestion de Sidescreen et peut être interrogée par la fonction **GETHMIResolution**.

L'analyseur de script attend un menu pour activer une forme ou pour effectuer une ramification vers d'autres formes. Le menu principal doit contenir le nom **main**. Dans Sidescreen, les balises de clé logicielle sont prises en charge de sorte que la navigation vers d'autres formes doit s'effectuer via un élément de commande de la forme.

Le nombre de pages Sidescreen ou de Widgets Sidescreen n'est pas limité par l'analyseur syntaxique.

3.14.2 Intégration des boîtes de dialogue Sidescreen

Les pages ou les widgets sont intégrés par le biais du fichier **slsidescreen.ini**.

Les pages doivent être créées à la section **[Sidescreen]**. Chaque page doit être indiquée avec le mot clé **PAGE** suivi du numéro continu de page et des attributs nécessaires. L'attribut **name** définit le nom d'objet de la page. L'attribut **implementation** indique la bibliothèque d'implémentation et le nom de classe. Ces deux attributs doivent être indiqués séparés par une virgule. Les pages du type **SlSideScreenPage** peuvent intégrer des éléments Sidescreen. Ceci s'effectue par les entrées **ELEMENT**. La règle de création d'une ligne d'élément **ELEMENTxxx** correspond à la règle de création d'une page. Si la page ne contient que des boîtes de dialogue configurées avec Easy XML, il faut indiquer **slagmforms.SlEESideScreenPage** comme valeur de l'attribut **implementation**.

Les widgets peuvent être ajoutés à un élément Sidescreen à la section **[Element_<name>]**.

La règle de création d'une ligne de widget **WIDGETxxx** correspond à la règle de création d'une page.

Pour activer un widget Sidescreen Easy XML, il faut indiquer **slagmforms.SlEESideScreenWidget** comme valeur de l'attribut **implementation**.

Syntaxe

```
ELEMENT002= name:=<name>, implementation:=SlSideScreenElement

...
...

[Element_<name>]
WIDGET001= name:=<Widget Name>, implementation:=
slagmforms.SlEESideScreenWidget
```

Le descripteur de widget est utilisé par défaut pour former le nom de module "main".

Exemple

```
PAGE004= name:=sidescreen_proginfluence,
implementation:=slagmforms.SlEESideScreenPage
```

Nom de module "main" : sidescreen_proginfluence.xml

Autres propriétés

Il est possible d'affecter d'autres caractéristiques aux pages ou aux widgets en saisissant une section avec le nom **PAGE_<pagename>**, **ELEMENT_<elmentname>** ou **WIDGET_<widgetname>** dans le fichier **slsidescreen.ini**.

Les caractéristiques suivantes peuvent être affectées à une page, un élément ou un widget :

Attribut	Signification/Comportement
TextId	Descripteur du texte indépendant de la langue
TextFile	Nom de fichier texte

Attribut	Signification/Comportement
TextContext	Contexte de texte EASY_XML
Icône	Nom de l'icône

Property	Signification/Comportement
maskPath	L'attribut property définit le nom de module "main" à utiliser
maskName	L'attribut property définit le nom de menu "main" à utiliser
focusable	L'attribut property définit si l'élément peut être activé

Exemple

```
[Page_sidescreen_proginfluence]
PROPERTY001= name:=focusable, type:=bool, value:="true"
Icon=sidescreen_proginfluence.png

[PAGE_sidescreen_proginfluence]
Icon= sidescreen_proginfluence.png
PROPERTY001= name:=focusable, type:=bool, value:="true"
PROPERTY002= name:=maskPath, type:=QString, value:="
sidescreen_proginfluence.xml"
PROPERTY003= name:=maskName, type:=QString, value:="main"
```

3.14.3 Gestion des langues et des textes

Pour la gestion des textes indépendants de la langue, il est possible d'affecter un fichier texte à chaque composant. Le nom du fichier texte se compose du nom du composant, de la version de langue et de l'extension ".ts".

Il faut utiliser **EASY_XML** comme contexte de texte.

Syntaxe

<name>_<language>.ts

Exemple

```
PAGE004= name:=sidescreen_proginfluence,
implementation:=slagmforms.SlEESideScreenPage
```

Nom de fichier texte : sidescreen_proginfluence**_eng.ts**

Si aucun fichier texte n'est indiqué, l'analyseur syntaxique recherche le descripteur de texte dans le fichier texte standard **oem_xml_screens_xxx.ts**.

3.14.4 Composants Sidescreen

3.14.4.1 Élément Sidescreen

Si une page est liée à l'implémentation standard, des éléments peuvent lui être affectés. À cet effet, vous devez créer une section **[Page_<page_name>]** et énumérer tous les éléments appartenant à la page.

Exemple

```
[Sidescreen]
PAGE001= name:=<page_name>, implementation:=SlSideScreenPage

[Page_<page_name>]

ELEMENT<numéro>= name:=<Element name>,
implementation:=SlSideScreenElement
```

Chaque élément nécessite une section **[Element_<element name>]** dans laquelle les caractéristiques et les widgets correspondants sont énumérés.

```
[Element_<element_name>]
WIDGET001= name:=<widget_name>, implementation:= <Implémentation>
```

3.14.4.2 Widget Sidescreen

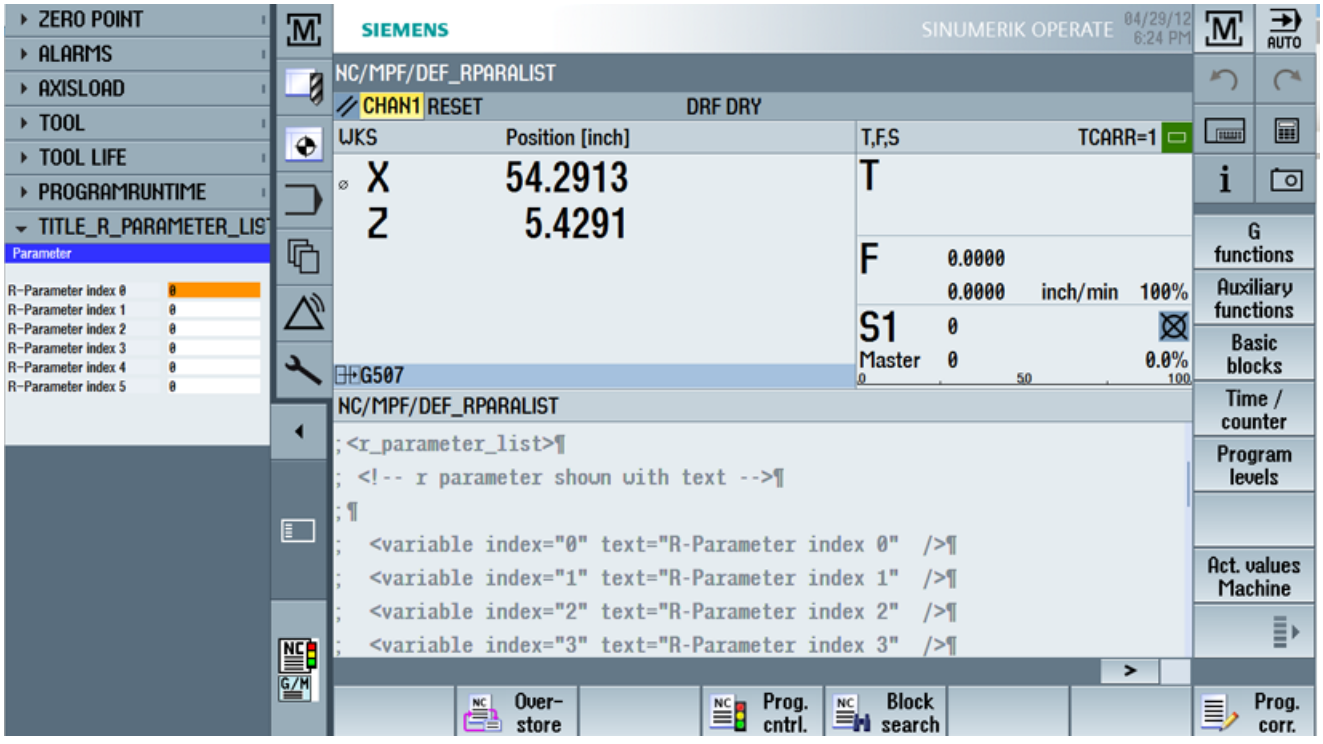
La gestion Sidescreen affecte à un widget une zone dans laquelle les formes configurées peuvent être représentées. Par défaut, un widget n'est pas activé, ce qui empêche d'éditer les champs. Ce comportement peut être modifié en saisissant l'attribut **focusable**. En ouvrant le widget, l'analyseur syntaxique ouvre la forme appartenant au menu principal. Dans la forme, il est possible d'accéder à d'autres menus via une instruction de navigation.

Exemple

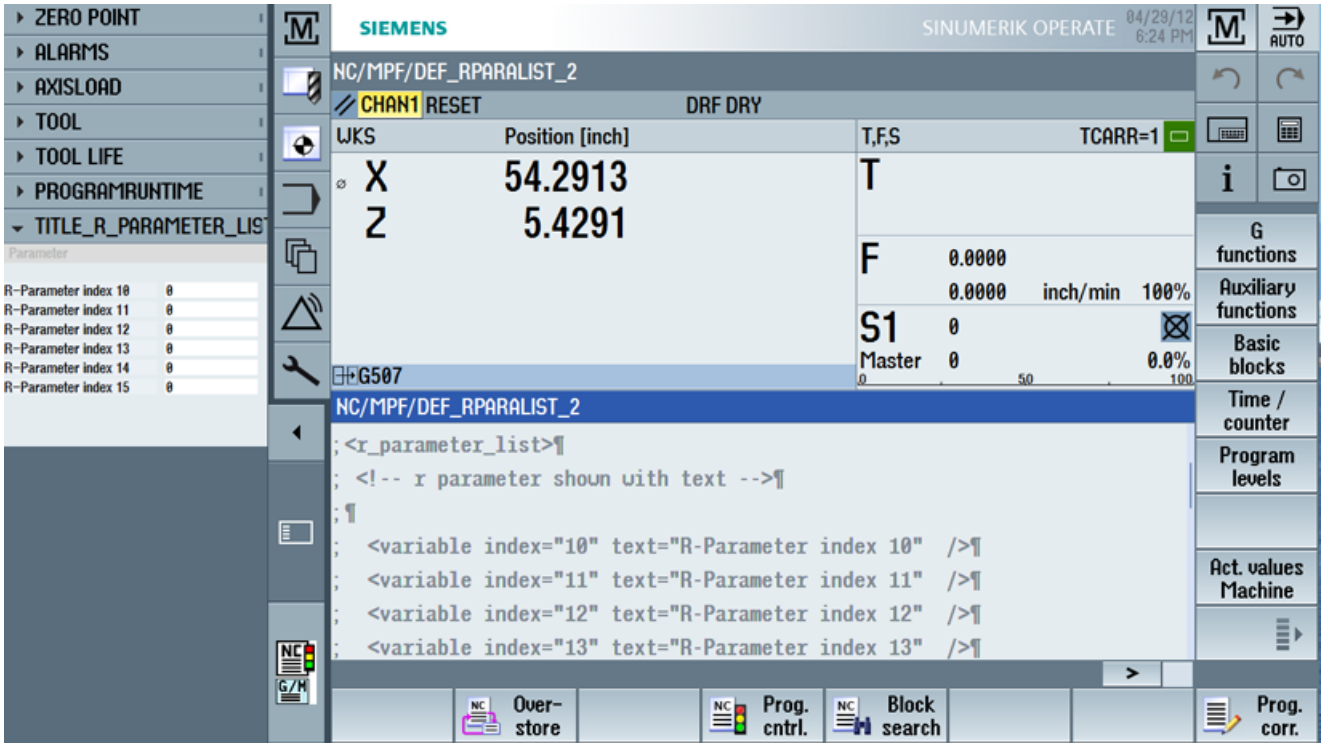
Le script Easy XML recherche la description du contenu de données du widget dans le programme pièce actif. Concrètement, la description d'une liste avec des paramètres R à afficher est attendue. Un texte descriptif peut être affecté à chaque paramètre.

Exemple de Toolbox

Custom Screen Sample\
side_screen_examples\sidescreenwidget\displayRparameter\rparameter_widget.xml



Si un autre programme pièce est sélectionné, le widget est remanié automatiquement.


ZERO POINT
ALARMS
AXISLOAD
TOOL
TOOL LIFE
PROGRAMRUNTIME
TITLE_R_PARAMETER_LIST
Parameter
R-Parameter index 10 0
R-Parameter index 11 0
R-Parameter index 12 0
R-Parameter index 13 0
R-Parameter index 14 0
R-Parameter index 15 0
SIEMENS
SINUMERIK OPERATE
04/29/12
6:24 PM
NC/MPF/DEF_RPARALIST_2
CHAN1 RESET
DRF DRY
WKS
Position [inch]
X 54.2913
Z 5.4291
T,F,S
TCARR=1
T
F 0.0000
0.0000 inch/min 100%
S1 0
Master 0 0.0%
G507
NC/MPF/DEF_RPARALIST_2
;<r_parameter_list>¶
; <!-- r parameter shoun uith text -->¶
;¶
; <variable index="10" text="R-Parameter index 10" />¶
; <variable index="11" text="R-Parameter index 11" />¶
; <variable index="12" text="R-Parameter index 12" />¶
; <variable index="13" text="R-Parameter index 13" />¶
AUTO
G functions
Auxiliary functions
Basic blocks
Time / counter
Program levels
Act. values Machine
Over-store
Prog. cntrl.
Block search
Prog. corr.

3.14.4.3 Page Sidescreen

Une page démarrée via l'implémentation **slagmforms.SIEESideScreenPage** met l'ensemble de la zone Sidescreen d'une forme à disposition. Elle ne possède pas de ligne d'en-tête, ce qui empêche de configurer un intitulé via l'instruction Caption. Par défaut, une page n'est pas activée, ce qui empêche d'éditer les champs. Ce comportement peut être modifié en saisissant l'attribut **focusable**. En ouvrant la page, l'analyseur syntaxique ouvre la forme appartenant au menu principal. Dans la forme, il est possible d'accéder à d'autres menus via une instruction de navigation.

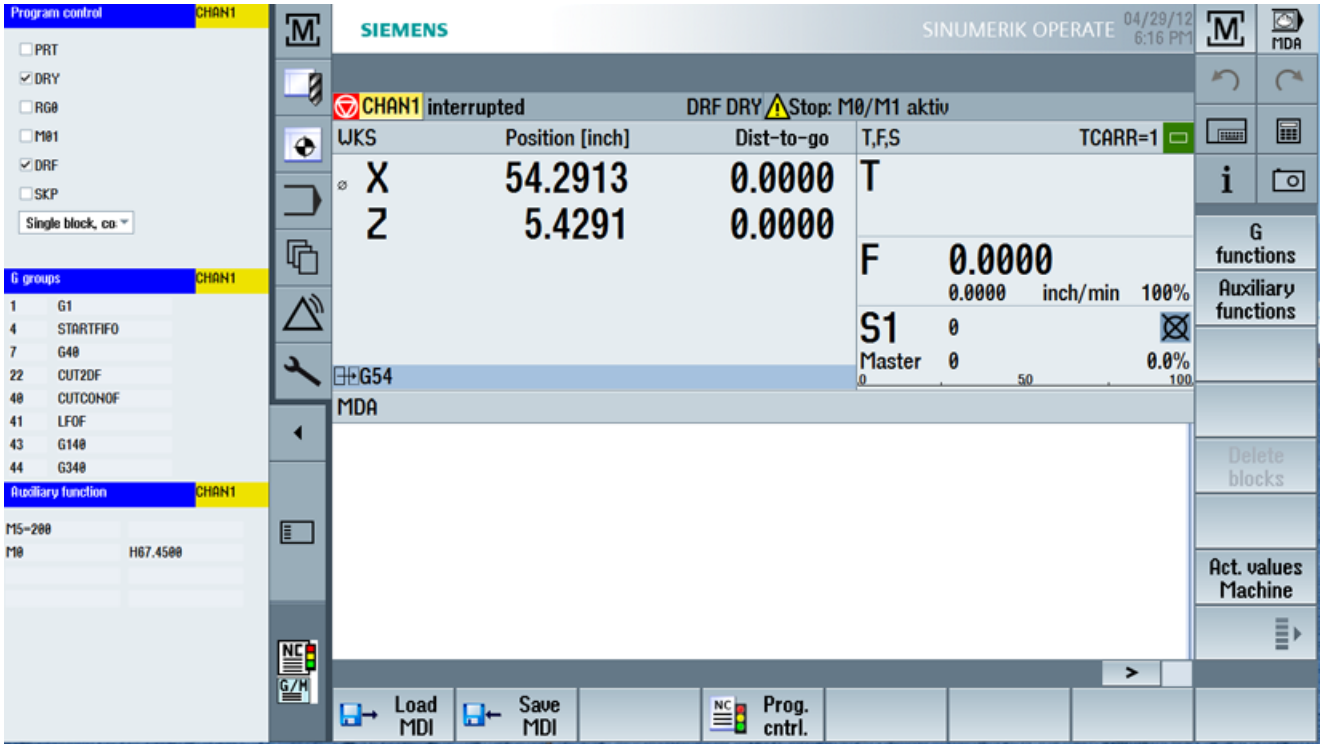
Exemple

La page Sidescreen est affichée comme page qui affiche exclusivement une forme Easy XML.

Dans cet exemple, des informations supplémentaires sont uniquement affichées dans la zone Sidescreen.

Exemple de Toolbox

Custom Screen Sample\side_screen_examples\sidecreenpage\sidescreen_proginfluence.xml




```
[Sidescreen]
PROPERTY001= name:=minimizable, type:=bool, value:="true"
PROPERTY002= name:=buttonBarVisible, type:=bool, value:="true"
PAGE001= name:=WidgetsPage, implementation:=SlSideScreenPage
```

```
PAGE004= name:=sidescreen_proginfluence,
implementation:=slagmforms.SlEESideScreenPage

[Page_sidescreen_proginfluence]
PROPERTY001= name:=focusable, type:=bool, value:="true"
Icon=sidescreen_proginfluence.png
```

Fichier texte : sidescreen_proginfluence_deu.ts

3.15 Application Display Manager

3.15.1 Easy XML dans le Display Manager

La fonction Easy XML peut être incluse dans la configuration Display Manager sur la base de la boîte de dialogue **SlSideScreenPage**. Cette boîte de dialogue peut afficher une ou plusieurs pages Sidescreen en fonction de la place disponible. Plusieurs pages sont affichées les unes à côté 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.

Toutes les applications sont automatiquement lancées au démarrage système par le Display Manager et fermées lors de l'arrêt du système. Cela signifie que les modifications du script ne prennent effet qu'après un redémarrage de l'IHM.

3.15.2 Intégration d'une Customer Area dans le menu du Display Manager

Pour appeler l'analyseur Easy XML dans la plage Customer, une Area doit être liée à la boîte de dialogue Easy XML. Ceci est indiqué par défaut avec la définition d'Area **AREA111** dans le fichier **systemconfiguration.ini**.

Syntaxe

```
AREA0111= name:=CustomXML, dialog:=SlEECustomDialog,
panel:=SlHdStdHeaderPanel
```

Cette plage est invisible à la livraison. L'activation se fait via le fichier **slamconfig.ini** dans lequel le mémento de visibilité **Visible** doit être changé de **false** à **true**.

```
[CustomXML]
TextId=SL_AM_CUSTOM
SidescreenTextId=SL_SIDESCREEN_CUSTOM
TextFile=slam
TextContext=SlAmAreaMenu
SoftkeyPosition=12
Visible=false -> true
```

Exemples

L'intégration de la plage Customer dans un menu du Display Manager se réalise via les fichiers INI du Display Manager en fonction de la résolution.

Dans le fichier correspondant, un point de menu doit être créé sous le commutateur **Operate buttons**, qui contient la référence à l'Area **CustomXML** :

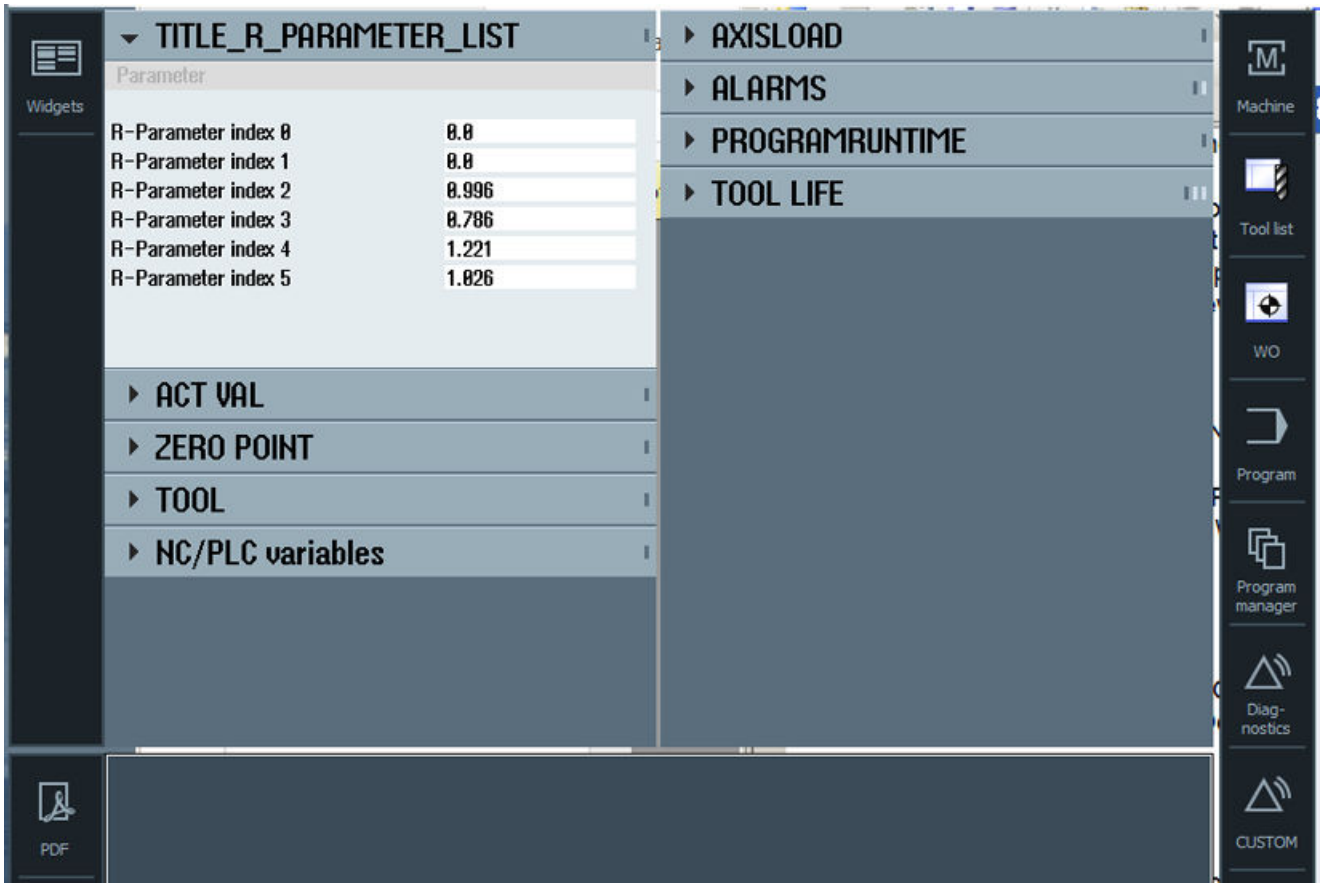
```
MENUITEM107= name:=mieasyXML, menuItemStyle:=misMenu,
onClicked:="hidePopup(); sendCmd(OPERATE, CustomXML);
showApp(defaultFrame, OPERATE)", image:=<bild.png>,
textID:= <textid>
```

Cet élément de menu est à son tour affecté à un menu dans Display Manager :

```
MENU020= name:=mOperate3, menuStyle:=msMenu,
defaultFrame:=fOperate3, items:="miMachine, miToolList,
miWorkOffset, miProgramEdit, miProgramManager, miDiagnosis,
mieasyXML, miStretch, miMaximizeOperate3"
```

3.15.3 Widget Display Manager

Les widgets dans Display Manager sont créés et gérés de la même manière que les widgets Sidescreen. La description des "Widgets Sidescreen" se déroule toutefois dans les fichiers de configuration du Display Managers, par exemple, dans les fichiers INI **sldmwidgets*.ini**. La référence aux fichiers est établie via **systemconfiguration.ini**.



Un exemple avec le fichier **rparameter_widget.xml** est donné au chapitre "Widget Sidescreen (Page 239)".

Exemple

systemconfiguration.ini

```
DLG109= name:=SlWidgetsApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false, cmdline:="-sidescreen1
sldmwidgets1.ini -sidescreen2 sldmwidgets2.ini -spacing 3"
```

Le nom de la boîte de dialogue est affecté à un frame :

```
FRAME020= name:=fTop, x:=66, y:=66, width:=760, height:=505,
app:=SlWidgetsApp
```

Les frames définis dans un Display :

```
DISPLAY001= name:=d3, frames:="fHeader, fOperate3, fMenuOperate3,
fTop, fMenuTop, fBottom, fMenuBottom"
```

sldmwidgets1.ini

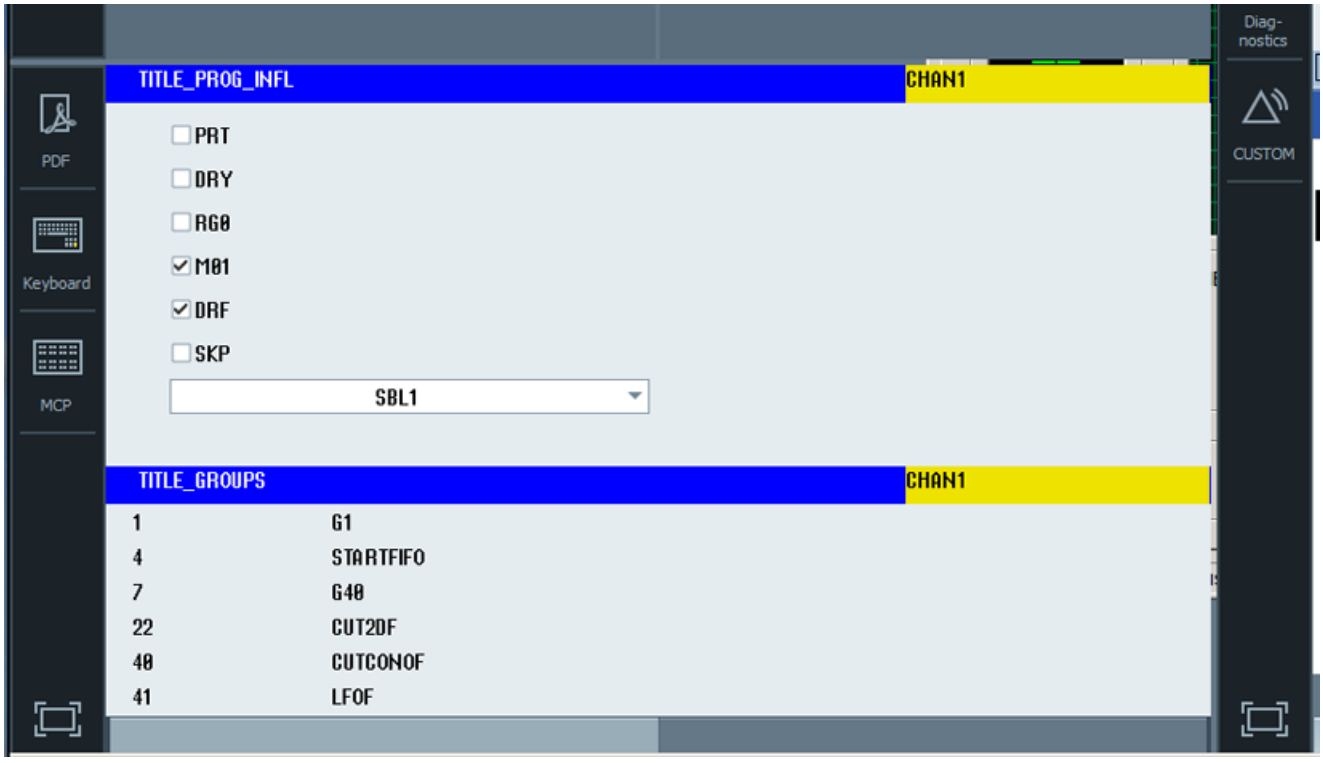
La structure et la signification des commutateurs et des éléments sont décrites au chapitre "Application Sidescreen (Page 235)".

```
[Sidescreen]
PAGE001= name:=WidgetsPage, implementation:=SlSideScreenPage
[Page_WidgetsPage]
ELEMENT001= name:=EE, implementation:=SlSideScreenElement
ELEMENT002= name:=AxesPosition, implementation:=SlSideScreenElement
ELEMENT003= name:=WorkOffset, implementation:=SlSideScreenElement
ELEMENT004= name:=ActTool, implementation:=SlSideScreenElement
ELEMENT005= name:=VariableView, implementation:=SlSideScreenElement

[Element_EE]
TextId=TITLE_R_PARAMETER_LIST
TextFile=rparameter_widget
TextContext=EASY_XML
WIDGET001= name:=rparameter_widget,
implementation:=slagmforms.SlEESideScreenWidget
```

Le script à exécuter (sans extension de fichier) et l'implémentation **slagmforms.SlEESideScreenWidget** sont affectés à **WIDGET001**.

3.15.4 Page Sidescreen du Display Manager

Les pages Sidescreen dans le Display Manager sont créées et gérées de la même manière que les pages Sidescreen. La description des "Pages Sidescreen" se déroule toutefois dans les fichiers INI **sldmxxxpage*.ini**. La référence à ces fichiers est établie via le fichier **systemconfiguration.ini**.



Un exemple avec le fichier **sidescreen_proginfluence.xml** est donné au chapitre "Page Sidescreen (Page 241)".

Exemple

sldmmcppage.ini

```
DLG110= name:=SlMcpApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false, cmdline:="-sidescreen1
sldmmcppage.ini"
```

Le nom de la boîte de dialogue est affecté à un frame :

```
FRAME028= name:=fBelowOperate, x:=896, y:=838, width:=1024,
height:=242, app:=SlKeyboardApp, runnableApps:="SlKeyboardApp,
SlMcpApp"
```

Les frames définis dans un Display :

```
DISPLAY002= name:=d4, frames:="fHeader, fOperate4, fMenuOperate4,
fBelowOperate, fMenuBelowOperate, fTop, fMenuTop, fBottom,
fMenuBottom"
MENUITEM124= name:=miMcp, menuItemStyle:=misMenu,
onClicked:="hidePopup(); showApp(defaultFrame, SlMcpApp)",
image:=dm_mcp.png, textID:=SL_DM_MCP
```

sldmmcppage.ini

```
[Sidescreen]
PAGE001= name:=McpPage,
implementation:=slsidescreenmcppage.SlSideScreenMcpPage
PAGE002= name:=sidescreen_proginfluence,
implementation:=slagmforms.SlEESideScreenPage
```

Le script à exécuter (sans extension de fichier) et l'implémentation **slagmforms.SlEESideScreenPage** sont affectés à **PAGE002**.

3.16 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 " aucune fonction
HK5	ALARM	Sélection du groupe fonctionnel "Diagnostic", boîte de dialogue "Liste des alarmes"
HK6	CUSTOM	Sélection du groupe fonctionnel "Custom"

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 demande précédente. 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 :

```
; configuration of OP hardkeys (KEY1-KEY9) and
; Touches matérielles AP (KEY50-KEY254)
[keyconfiguration]
; MACHINE key (hardkey block)
KEY1.0 = area:=AreaMachine, dialog:=SlMachine
```

```
; PROGRAM key (hardkey block)
KEY2.0 = area:=AreaProgramEdit
; OFFSET key (hardkey block)
KEY3.0 = area:=AreaParameter
; PROGRAM MANAGER key (hardkey block)
KEY4.0 = area:=AreaProgramManager
; ALARM key (hardkey block)
KEY5.0 = area:=AreaDiagnosis, dialog:=SlDgDialog,
cmdline:="-slGfwHmiScreen SlDgAeAlarmsScreen"
; CUSTOM (hardkey block)
KEY6.0 = area:=Custom
; MACHINE key (optional, Shift + F10)
KEY7.0 = area:=AreaMachine, dialog:=SlMachine,
cmdline:="-MKey"
; MENU USER (hardkey block, Ctrl + F10)
KEY10.0 = area:=MenuUser
; MENU FUNCTION (hardkey block, Ctrl + Shift + F10)
KEY11.0 = area:=MenuFunction
KEY50.0 = area:=CustomXML, dialog:=SlEECustomDialog
```

Les descripteurs de zone et de boîte de dialogue figurent dans le fichier systemconfiguration.ini sous [Répertoire système siemens]/cfg.

Si seule l'indication de dialogue**SlEECustomDialog** est utilisée, l'analyseur syntaxique attend le fichier **xmldial.xml** dans le répertoire d'application ainsi qu'une balise de menu avec le nom **main**. D'autres noms de fichier ou noms de menu principal peuvent être indiqués par l'insertion de la clé **cmdline**. Le paramètre **-mainModule** détermine la description XML à charger. Dans ce script, l'analyseur syntaxique attend le menu principal. Le paramètre **-entry** détermine le nom du menu principal. Si ce paramètre est manquant, l'analyseur syntaxique utilise le nom **main** pour démarrer la fonction. Le paramètre **-conf** avec la valeur **slagmdialog.hmi** doit toujours être présent.

Exemple :

```
KEY50.0 = area:=CustomXML, dialog:=SlEECustomDialog,
cmdline:="-conf slagmdialog.hmi -mainModule restore.xml -entry cmc2"
KEY51.0 = area:=CustomXML, dialog:=SlEECustomDialog,
cmdline:="-conf slagmdialog.hmi -mainModule activate.xml
-entry cmc1"
```

Interface utilisateur

Les entrées suivantes dans le fichier systemconfiguration.ini permettent l'utilisation d'Easy XML pour la fonction décrite ci-dessus.

```
AREA0111= name:=CustomXML, dialog:=SlEECustomDialog,
panel:=SlHdStdHeaderPanel
DLG056= name:=SlEECustomDialog,
implementation:=slagmdialog.SlEECustomDialog, process:=SlHmiHost1,
preload:=false, terminate:=true, cmdline:="-conf slagmdialog.hmi"
```

3.17 Affectation des touches logicielles réservées aux boîtes de dialogue utilisateur

Dans tous les groupes fonctionnels standard, une touche logicielle est réservée à l'extension de la fonctionnalité existante. Pour l'activation et la liaison d'une touche logicielle OEM à un projet utilisateur, les fichiers correspondants sont enregistrés dans le système dans le répertoire suivant :

/siemens/sinumerik/hmi/template/cfg

Ces fichiers contiennent des descriptions XML spécifiques à un groupe fonctionnel, qui sont interprétées au moment du démarrage du logiciel de commande et intégrées à l'arborescence de menus du groupe fonctionnel.

Pour l'intégration d'une application propre, le fichier pertinent pour le groupe fonctionnel correspondant doit être copié du répertoire de modèles dans le répertoire suivant, puis être modifié :

/oem/sinumerik/hmi/template/cfg

3.17.1 Définition du texte des touches logicielles indépendamment de la langue

La balise TEXTFILE contient le nom du fichier texte devant être attribué au domaine OEM. Le nom de fichier doit être indiqué sans extension de langue ni de fichier.

Syntaxe

```
<TEXTFILE>oemtextfile</TEXTFILE>
```

Exemple

oem_xml_screens_deu.ts

```
<TEXTFILE>oem_xml_screens</TEXTFILE>
```

Le texte des touches logicielles à afficher est géré dans la balise **property** de la balise **softkey**. La valeur OEM doit être remplacée par l'ID de texte souhaité. La propriété **translationContext** renvoie à une zone dans le fichier texte qui est attribuée au texte. Dans le cas d'easyXML, le contexte EASY_XML doit être indiqué.

Syntaxe

```
<PROPERTY name="textID" type="QString">OEM</PROPERTY>
<PROPERTY name="translationContext" type="QString">OEMContext</
PROPERTY>
```

Exemple

```
<PROPERTY name="textID" type="QString">MY_TEXT1</PROPERTY>
<PROPERTY name="translationContext" type="QString">EASY_XML</
PROPERTY>
```

3.17.2 Attribution d'une zone Easy XML

La cible de navigation doit toujours être définie dans la balise **softkey**. Pour cela, dans la balise **area**, il convient de saisir la cible de navigation **CustomXML** dans l'attribut **name**. Les attributs **dialog** et **screen** doivent être supprimés, car ces paramètres sont définis automatiquement.

Syntaxe

```
<NAVIGATION target="area">
<AREA name="OEMArea" dialog="OEMDialog" screen="OEMScreen"/>
</NAVIGATION>
```

Exemple

```
<NAVIGATION target="area">
<AREA name="CustomXML" />
</NAVIGATION>
```

Si la cible de navigation **area** est utilisée, l'analyseur syntaxique attend le fichier xmldial.xml comme module de démarrage et le menu **main** comme point d'entrée dans la gestion des menus. Afin de pouvoir proposer plusieurs applications en parallèle, la fonction **switchToDialog** doit être utilisée. Pour cela, il convient de remplacer la balise NAVIGATION par la balise FUNCTION. Le domaine **CustomXML**, la boîte de dialogue **SlEECustomDialog** ainsi que le nom du module de démarrage et du menu principal doivent être transférés à cette fonction comme arguments d'appel.

La balise **switchToArea** permet de retourner dans le groupe fonctionnel standard depuis le domaine easyXML.

Syntaxe

```
<switchToArea name="<Area>"/>
...
<FUNCTION args="-area CustomXML -dialog SlEECustomDialog -mainModule
<Filename> -entry <Menuname>" name="switchToDialog"/>
```

Exemple de projet

sldiagnose_oem.xml

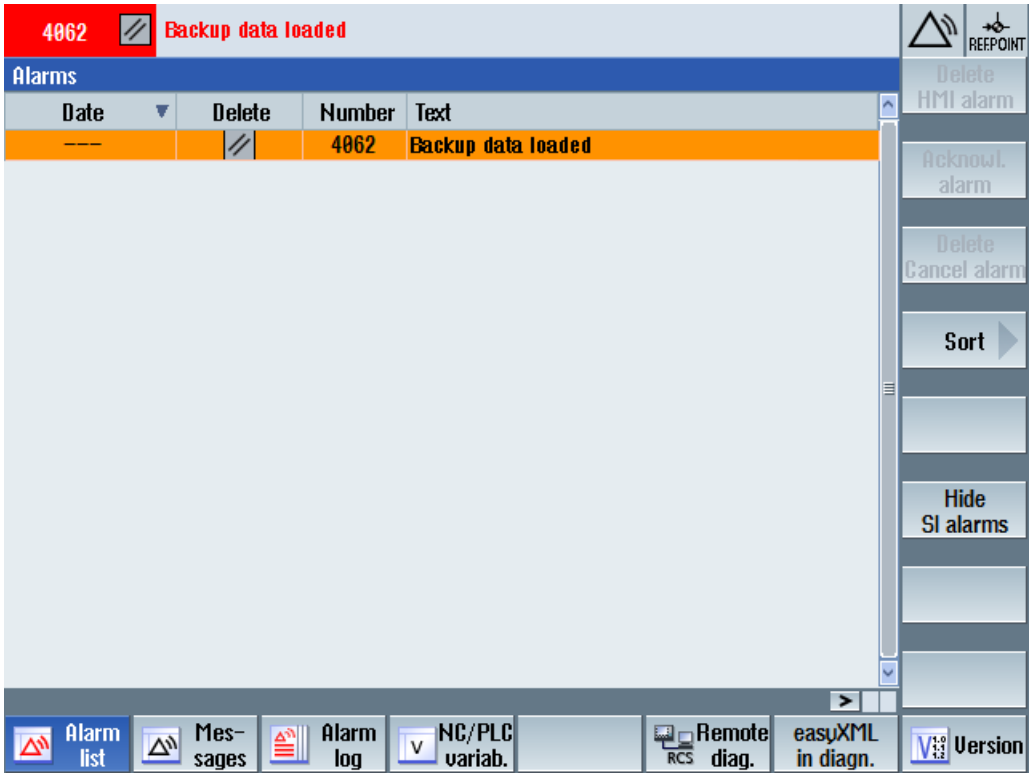
```
<!DOCTYPE DIALOG>
<DIALOG>
<TEXTFILE>oem_xml_screens</TEXTFILE>
<!-- ========================================================================
-->
<!-- OEM softkey number 7 on first horizontal main menu in area diagnosis --
>
<!-- ========================================================================
-->
<MENU name="DgGlobalHu">
 <ETCLEVEL id="0">
 <SOFTKEYGROUP name="GroupEtc">
 <SOFTKEY position="7">
  <PROPERTY name="textID" type="QString">DG_SK</PROPERTY>
  <PROPERTY name="translationContext" type="QString">EASY_XML</PROPERTY>
  <NAVIGATION target="area">
   <AREA name="CustomXML" />
  </NAVIGATION>
 </SOFTKEY>
</SOFTKEYGROUP>
</ETCLEVEL>
</MENU>
</DIALOG>
```

ou

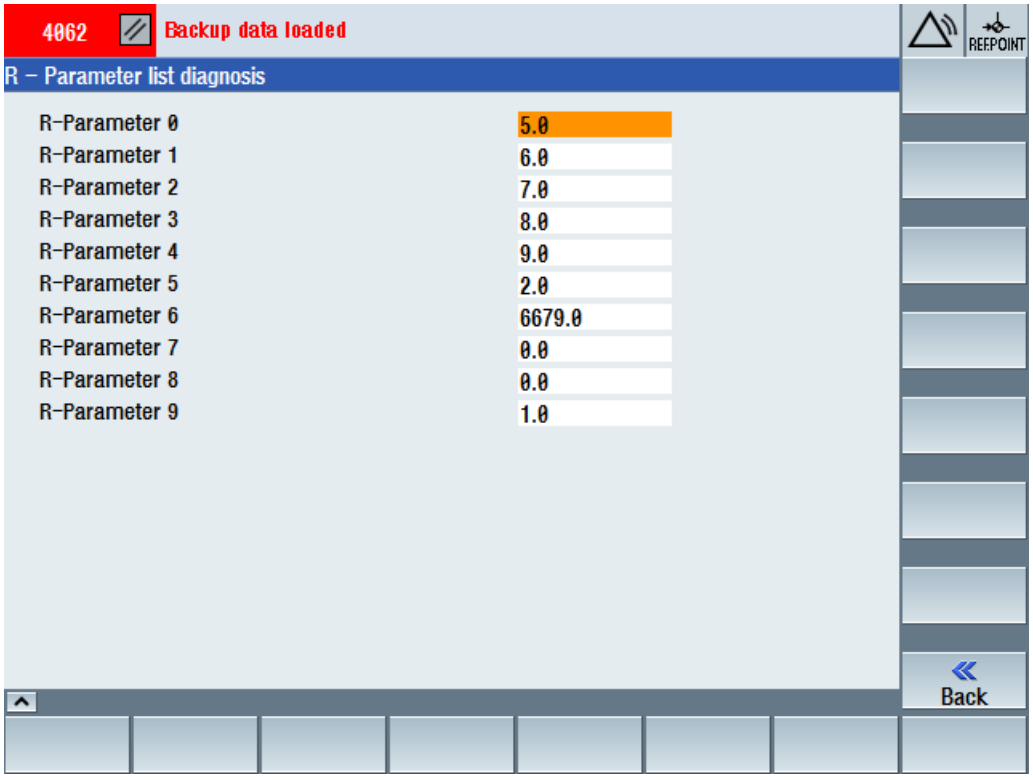
```
<!DOCTYPE DIALOG>
<DIALOG
<!-- ========================================================================
-->
<!-- OEM softkey number 7 on first horizontal main menu in area diagnosis --
>
<!-- ========================================================================
-->
<TEXTFILE>oem_xml_screens</TEXTFILE>
<MENU name="DgGlobalHu">
 <ETCLEVEL id="0">
 <SOFTKEYGROUP name="GroupEtc">
 <SOFTKEY position="7">
  <PROPERTY name="textID" type="QString">DG_SK</PROPERTY>
  <PROPERTY name="translationContext" type="QString">EASY_XML</PROPERTY>
  <FUNCTION args="-area CustomXML -dialog SlEECustomDialog -mainModule
dg.xml -entry main" name="switchToDialog"/>
 </SOFTKEY>
</SOFTKEYGROUP>
</ETCLEVEL>
</MENU>
</DIALOG>
```

dg.xml

```
<DialogGui>
<menu name = "main">
 <open_form name = "R_PARAMETER_LIST" />
 <softkey_back>
  <switchToArea name="AreaDiagnosis" dialog="SlDgDialog"/>
 </softkey_back>
</menu>
<!-- This form shows a R - parameter list -->
<form name = "R_PARAMETER_LIST">
 <init>
 <caption>R - Parameter list diagnosis</caption>
 <data_access type="true" />
  <control name = "c1" xpos = "322" ypos = "34" refvar="nck/Channel/
Parameter/R[0]" hotlink="true" />
  <control name = "c2" xpos = "322" ypos = "54" refvar="nck/Channel/
Parameter/R[1]" hotlink="true" />
  <control name = "c3" xpos = "322" ypos = "74" refvar="nck/Channel/
Parameter/R[2]" hotlink="true" />
  <control name = "c4" xpos = "322" ypos = "94" refvar="nck/Channel/
Parameter/R[3]" hotlink="true" />
  <control name = "c5" xpos = "322" ypos = "114" refvar="nck/Channel/
Parameter/R[4]" hotlink="true" />
  <control name = "c6" xpos = "322" ypos = "134" refvar="nck/Channel/
Parameter/R[5]" hotlink="true" />
  <control name = "c7" xpos = "322" ypos = "154" refvar="nck/Channel/
Parameter/R[6]" hotlink="true" />
  <control name = "c8" xpos = "322" ypos = "174" refvar="nck/Channel/
Parameter/R[7]" hotlink="true" />
  <control name = "c9" xpos = "322" ypos = "194" refvar="nck/Channel/
Parameter/R[8]" hotlink="true" />
  <control name = "c10" xpos = "322" ypos = "214" refvar="nck/Channel/
Parameter/R[9]" hotlink="true" />
 </init>
 <paint>
  <text xpos = "23" ypos = "34">R-Parameter 0</text>
  <text xpos = "23" ypos = "54">R-Parameter 1</text>
  <text xpos = "23" ypos = "74">R-Parameter 2</text>
  <text xpos = "23" ypos = "94">R-Parameter 3</text>
  <text xpos = "23" ypos = "114">R-Parameter 4</text>
  <text xpos = "23" ypos = "134">R-Parameter 5</text>
  <text xpos = "23" ypos = "154">R-Parameter 6</text>
  <text xpos = "23" ypos = "174">R-Parameter 7</text>
  <text xpos = "23" ypos = "194">R-Parameter 8</text>
  <text xpos = "23" ypos = "214">R-Parameter 9</text>
 </paint>
</form>
</DialogGui>
```


4062
Backup data loaded
REFPOINT
Alarms
Date
Delete
Number
Text

4062
Backup data loaded
Delete HMI alarm
Acknowl. alarm
Delete Cancel alarm
Sort
Hide SI alarms
Alarm list
Mes-sages
Alarm log
NC/PLC variab.
Remote diag.
easyXML in diagn.
Version


4062
Backup data loaded
REFPOINT
R – Parameter list diagnosis
R-Parameter 0 5.0
R-Parameter 1 6.0
R-Parameter 2 7.0
R-Parameter 3 8.0
R-Parameter 4 9.0
R-Parameter 5 2.0
R-Parameter 6 6679.0
R-Parameter 7 0.0
R-Parameter 8 0.0
R-Parameter 9 1.0
Back

3.17.3 Descriptions de balise

Balise softkey

En dehors de la balise **softkey**, l'état de la touche logicielle peut être défini avec l'attribut **state** en utilisant l'attribut POSITION. Le numéro de la touche logicielle pour laquelle l'état doit être défini doit être indiqué comme valeur d'attribut.

Exemple

```
<menu name = "menu_sktest">

 <state type="$$$define_sk_type" POSITION="2"/>

<softkey POSITION="2" type="user_controlled" picture="$$$bmp_name">
 <caption>Toggle%nSK</caption>
 <if>
 <condition>sk_type == 0 </condition>
 <then>
  <op> sk_type = 1 </op>
  <op> define_sk_type = _T"PRESSED" </op>
  <op>bmp_name = _T"f:\appl\red_led_on.bmp"</op>
  <state type="$$$define_sk_type" />
 </then>
 <else>
  <op> define_sk_type = _T"NOTPRESSED" </op>
  <op>bmp_name = _T"f:\appl\red_led_off.bmp"</op>
  <op> sk_type = 0 </op>
  <state type="$$$define_sk_type" />
 </else>
 </if>
  <print text="%s">sk_type_name</print>
  <navigation>menu_sktest</navigation>
</softkey>
...
...
```

Balise typedef

Remarque

La valeur par défaut des éléments dans les exemples doit être supprimée.

Une variable est déclarée à l'intérieur de la définition du type au moyen de la balise **element**. Les attributs de la balise correspondent aux attributs de l'instruction **let**. La valeur par défaut des éléments peut être configurée lors de la création des variables.

Exemple

RECT:

```
<typedef name="StructRect" type="struct" >
<element name="left" type="int" />
<element name="top" type="int" />
<element name="right" type="int" />
<element name="bottom" type="int" />
</typedef>
```

POINT :

```
<typedef name="StructPoint" type="struct" >
<element name="x" type="int" />
<element name="y" type="int" />
</typedef>
```

SIZE:

```
<typedef name="StructSize" type="struct" >
<element name="width" type="int" />
<element name="height" type="int" />
</typedef>
```

Balise let

L'attribut **dim** sert à indiquer le nombre d'éléments de tableau. Pour un tableau à deux dimensions, la deuxième dimension est spécifiée après la première, séparée par une virgule. L'accès à un élément de tableau s'effectue via l'indice de tableau spécifié après le nom de variable entre crochets. Les dimensions du tableau peuvent être définies directement ou par le contenu d'une variable.

Exemple

```
 <let name="d1" >3</let>

<let name="list_string" dim="3" type="string" />
```

ou

```
<let name="list_string" dim="$d1" type="string" />
...
...
<op>
  list_string[2] = _T"test";
</op>
```

ou

```
<let name="d1" >3</let>
<let name="d2" >6</let>

<let name="list_string3" dim="3, $d2" type="string" />

<op>
  list_string3[2, 5] = _T"test";
</op>
```

3.18 Création de textes indépendants de la langue

Tous les textes utilisés dans les masques de dialogue peuvent être gérés indépendamment de la langue, en utilisant dans les boîtes de dialogue des descripteurs de texte qui sont remplacés lors de l'exécution par un texte dans la langue sélectionnée. Ce texte doit être enregistré avec le descripteur de texte pour chaque langue proposée dans un fichier texte au format UTF8.

Par défaut, le fichier oem_xml_screens_<code langue>.ts est attribué comme fichier texte à la fonction Easy XML. Le descripteur EASY_XML doit être indiqué comme contexte de texte.

Pour caractériser un descripteur de texte, il convient de faire précéder le descripteur de texte dans le script de deux symboles de dollar.

Structure

oem_xml_screens_deu.ts

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
    <name>EASY_XML</name>
    <message>
    <source>MY_TEXT1</source>
    <translation>text1 from textfile</translation>
    </message>
    <message>
    <source>MY_TEXT2</source>
    <translation>text2 from textfile</translation>
    </message>
</context>
</TS>
```

Exemple

```
<text xpos ="120" ypos="158">$$MY_TEXT1</text>
```

3.19 Fichiers texte spécifiques au projet

3.19.1 Création de fichiers texte spécifiques au projet

Pour la gestion de projets distincts, il est possible d'utiliser des fichiers texte séparés pour un projet, pour chaque forme et pour chaque menu. La déclaration a lieu avec l'attribut TEXTFILE dans les balises correspondantes.

Le nom du fichier texte doit être transféré comme valeur d'attribut sans descripteur de langue ni extension de fichier. Il est possible d'accéder aux textes tant que la forme ou le menu n'est pas fermé. Si un fichier texte est chargé avec la balise **DialogGui**, il est possible d'accéder au contenu tant que le projet est ouvert. Si un contexte de texte est utilisé plusieurs fois, la recherche du descripteur de texte démarre dans le fichier chargé en dernier. Un contexte de texte peut être utilisé par fichier texte.

Exemple

main_form_screens_deu.ts

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
    <name>EASY_XML</name>
    <message>
    <source>MAIN_FORM_TITLE</source>
    <translation>user text file title</translation>
    </message>
</context>
</TS>
```

Activation

```
<DialogGui textfile="main_form_screens">
...
...
</DialogGui>
```

ou

```
<form name="main_form" textfile="main_form_screens">
...
...
</form>
```

ou

```
<menu name="main" textfile="main_form_screens">
...
...
</menu>
```

Exemple de boîte de dialogue

main_form_screens_deu.ts

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
    <name>EASY_XML</name>
    <message>
    <source>MAIN_FORM_TITLE</source>
    <translation>user text file title</translation>
    </message>
    <message>
    <source>MAIN_FORM_TEXT</source>
    <translation>text from text file</translation>
    </message>
</context>
</TS>
```

main2_sktext_screens_deu.ts

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
    <name>EASY_XML</name>
    <message>
    <source>SK1</source>
    <translation>Softkey1</translation>
    </message>
</context>
</TS>
```

form2_screens_deu.ts

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
    <name>EASY_XML</name>
    <message>
    <source>FORM2_TITLE</source>
    <translation>Form2 Title</translation>
    </message>
    <message>
    <source>FORM2_TEXT</source>
    <translation>Form2 text from text file</translation>
    </message>
</context>
</TS>
```

Script de boîte de dialogue

xmldial.xml

```
<DialogGui textfile="main_form_screens">
<menu name = "main">
    <open_form name = "main_form" />
    <softkey POSITION="1">
    <caption>Menu 2</caption>
    <navigation>menu_2</navigation>
    </softkey>
</menu>
<menu name = "menu_2" textfile="main2_sktext_screens">
    <open_form name = "form2" />
    <softkey POSITION="1">
    <caption>$$SK1</caption>
    </softkey>
    <softkey_back>
    <navigation>main</navigation>
    </softkey_back>
</menu>
<form name="main_form" >
 <init>
    <data_access type = "true" />
    <caption>$$MAIN_FORM_TITLE</caption>
 </init>
 <paint>
    <text xpos="5" ypos="30">$$MAIN_FORM_TEXT</text>
 </paint>
</form>
<form name="form2" textfile="form2_screens">
 <init>
    <data_access type = "true" />
    <caption>$$FORM2_TITLE</caption>
 </init>
 <paint>
    <text xpos="5" ypos="30">$$FORM2_TEXT</text>
 </paint>
</form>
</DialogGui>
```

3.19.2 Saisie d'un contexte de texte

Au sein d'un fichier texte, il est possible de regrouper des textes par des noms de domaines définis par l'utilisateur. À l'intérieur de la balise **context**, le descripteur de domaine est défini avec la balise **name**.

Syntaxe

```
<context>
 <name>name</name>
</context>
```

Exemple

```
<context>
 <name>my context 1</name>
...
...
</context>
<context>
 <name>my context 2</name>
...
...
</context>
```

Si l'exécution a lieu avec un contexte propre, le nom du contexte doit être indiqué avec le nom du fichier texte pour un projet, une forme ou un menu via l'attribut TEXTCONTEXT. Il est possible d'accéder aux textes enregistrés à l'intérieur du contexte tant qu'aucun nouveau contexte n'est défini. En d'autres termes, un contexte défini dans le projet est remplacé par le contexte défini dans une forme ou un menu.

Exemple de projet

form2_screens_deu.ts

```
?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
 <name>EASY_XML</name>
  <message>
   <source>FORM2_TITLE</source>
   <translation>Form2 Title</translation>
  </message>
  <message>
   <source>FORM2_TEXT</source>
   <translation>Form2 text from text file</translation>
  </message>
</context>
<context>
 <name>MY_FORM_CONTEXT</name>
  <message>
   <source>FORM3_TITLE</source>
   <translation>Title from the text context MY_FORM_CONTEXT</translation>
  </message>
  <message>
   <source>FORM3_TEXT</source>
   <translation>Form3 text from text file</translation>
  </message>
</context>
<context>
 <name>MY_SOFTKEY_CONTEXT</name>
  <message>
   <source>MENU3_SK1</source>
   <translation>SK%n1</translation>
  </message>
</context>
</TS>
```

using_textcontext.xml

```
...
<menu name = "menu3" textfile="form2_screens"
textcontext="MY_SOFTKEY_CONTEXT">
 <open_form name = "menu3_form" />
 <softkey POSITION="1">
  <caption>$$SK1</caption>
 </softkey>
 <softkey_back>
  <navigation>main</navigation>
 </softkey_back>
</menu>
<form name="menu3_form" textfile="form2_screens"
textcontext="MY_FORM_CONTEXT">
 <init>
  <data_access type = "true" />
  <caption>$$FORM3_TITLE</caption>
 </init>

 <paint>
  <text xpos="5" ypos="30">$$FORM3_TEXT</text>
 </paint>
</form>
...
...
```

3.19.3 Indiquer la liste de fichiers texte

Pour utiliser plusieurs fichiers texte dans un projet, les noms des fichiers requis peuvent être transférés vers l'analyseur syntaxique dans une liste à l'aide de l'attribut **textfilelist**.

La notation des noms de fichier texte s'effectue ligne par ligne et doit se terminer par un saut de ligne. Le nom du fichier sans extension de langue ni extension de fichier est attendu comme nom de fichier texte.

Exemple

Le fichier `form2_screens_deu.ts` doit être exécuté dans la liste avec `form2_screens`.

textfilelist.ini

```
form2_screens
...
...
```

Activation

```
<DialogGui textfilelist="txtfilelist.ini">
...
...
</DialogGui>
```

3.20 Restaurer la session

Lors de la fermeture d'un projet Easy XML, toutes les variables locales sont débloquées et les instructions de script sont déchargées. Si l'utilisateur souhaite conserver les données de variables après l'arrêt de la commande, ces variables doivent comporter l'attribut **permanent**. Les données qui doivent être conservées jusqu'à l'arrêt de la commande doivent être caractérisées par l'attribut **poweroff**.

Si l'application Easy XML est de nouveau sélectionnée, il est possible de commander dans le menu principal quel menu ou quelle forme devra être activé(e) une fois l'application à nouveau sélectionnée. Il incombe au programmeur de s'assurer que toutes les données requises à cet effet sont disponibles.

Si l'attribut **restoresession** est défini dans la balise **DialogGUI**, l'analyseur syntaxique organise la nouvelle sélection du menu ouvert en dernier et de la forme ouverte en dernier. Le programmeur est responsable de la mise à disposition des données requises.

Création de boîtes de dialogue de mise en service 4

4.1 Vue d'ensemble des fonctions

Usage

La fonction "Easy Extend" permet de mettre en service, activer, désactiver ou tester aisément des appareils supplémentaires. La commande indique les appareils disponibles ainsi que leur état dans une liste. Le système peut gérer au maximum 64 appareils.

L'activation ou la désactivation d'un appareil s'effectue par maniement de touche logicielle.

La fonction "Easy Extend" est disponible dans le groupe fonctionnel "Paramètres".

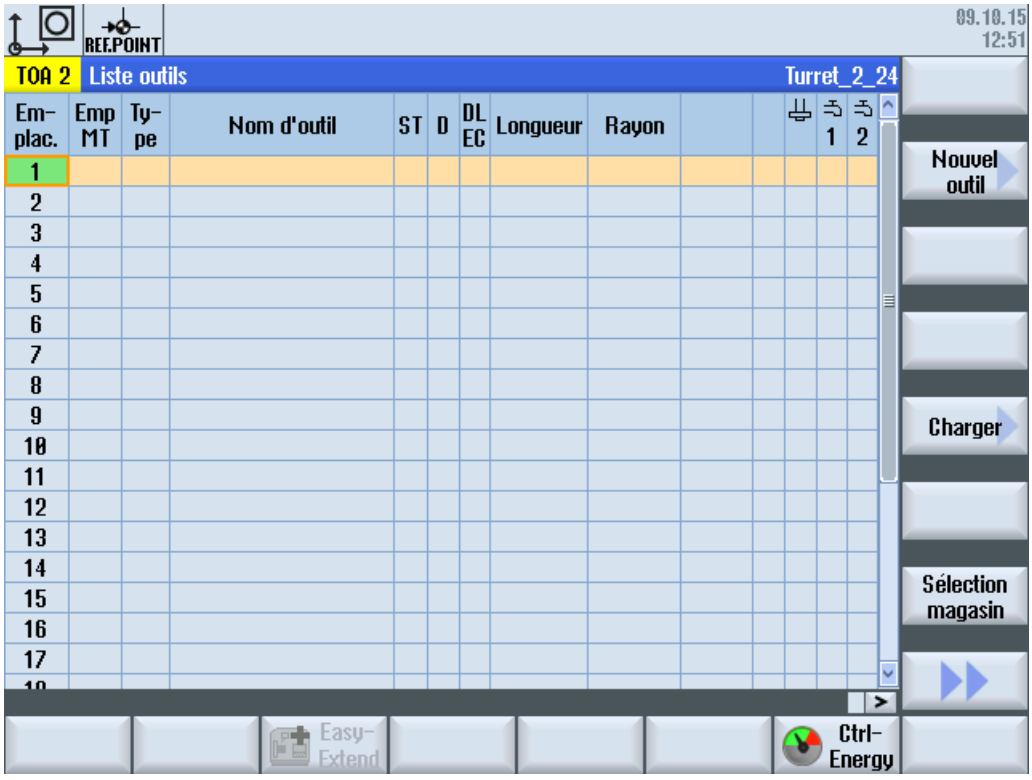


Figure 4-1 Vue racine "Paramètres"

Configuration

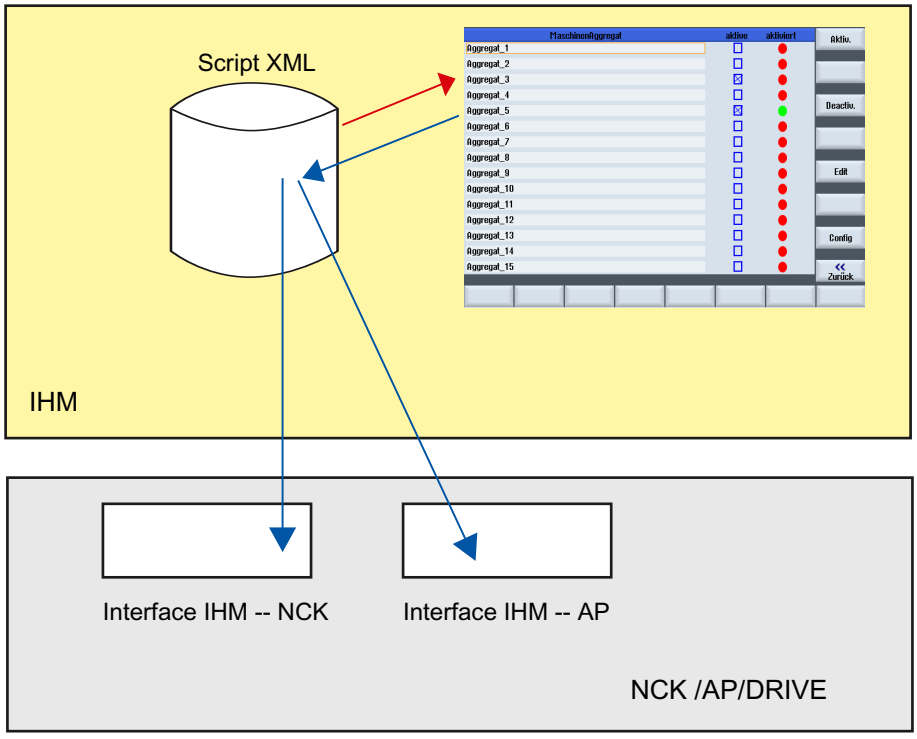


Figure 4-2 Fonctionnement de "Easy Extend"

Pour pouvoir utiliser la fonction "Easy Extend", les fonctions suivantes doivent être configurées par le constructeur de machines :

- **Interface AP ↔ IHM**
  Les gestion des appareils en option est réalisée via la liaison entre l'interface utilisateur et l'AP.
- **Traitement de scripts**
  Le constructeur de machine enregistre dans un script d'instructions les séquences à effectuer pour installer, activer, désactiver et tester un appareil.
- **Boîte de dialogue des paramètres (option)**
  La boîte de dialogue des paramètres sert à afficher les informations d'appareil enregistrées dans le fichier de script.

Archivage des fichiers

L'archivage des fichiers qui font partie de "Easy Extend" s'effectue sur la carte CompactFlash du système, qui se trouve dans le répertoire "dvm" (constructeur de machines).

Fichier	Nom	Répertoire cible
Fichier texte	oem_aggregate_xxx.ts	`\oem\sinumerik\hmi\lng`
Fichier de script	agm.xml	`\oem\sinumerik\hmi\dvm`
Fichier d'archivage	quelconque	`\oem\sinumerik\hmi\dvm\archives`
Programme utilisateur AP	quelconque	PLC

4.2 Configuration dans le programme utilisateur AP

Chargement de configurations

Les configurations créées sont transmises avec le fichier texte et le fichier de script dans le répertoire constructeur de la commande. En outre, le programme utilisateur AP adéquat doit être chargé.

Programmation des appareils

La communication entre l'élément de commande et l'AP s'effectue dans le programme utilisateur AP via un bloc de données défini par le programmeur pour lequel 128 mots sont réservés pour la gestion des appareils. La description du descripteur de bloc de donnée figure au chapitre "PLC_INTERFACE (Page 283)".

Le bloc de données doit être créé et chargé en tant que bloc utilisateur. Dans le script "agm.xml", le bloc correspondant doit être signalé par la balise **plc_interface** de la fonction EasyExtend.

Remarque

Compatibilité

Il est recommandé de définir le bloc de données DB9905 somme interface AP pour que les scripts soient également compatibles avec la SINUMERIK 828D.

Exemple :

```
<plc_interface name = "plc/db1000.dbb0" />
```

Pour créer le bloc, il est possible d'utiliser les formats de données et les descripteurs symboliques du tableau ci-après :

Format de données / descripteur symbolique	Description
DBX0.0 Enable_1 BOOL Bit OFF OFF IHM → AP	L'appareil est mis en service
DBX0.1 Activate_1 BOOL Bit OFF OFF IHM → AP	L'appareil doit être activé
DBX0.2 Deactivate_1 BOOL Bit OFF OFF IHM → AP	L'appareil doit être désactivé
DBB1 Res_1 BYTE sans signe 0 0	Réservé pour utilisation future
DBX2.0 IsActive_1 BOOL Bit OFF OFF AP → IHM	L'appareil est activé
DBX2.1 Error_1 BOOL Bit OFF OFF AP → IHM	L'appareil est défectueux
DBB3 DeviceId_1 BYTE sans signe 0 0	Numéro d'appareil univoque

L'attribution des mots d'AP s'effectue en commençant par l'appareil 1 :

Bloc de données	Désignation de l'appareil
DB9905.DBB0	Appareil 1
DB9905.DBB4	Appareil 2
...	...

Bloc de données	Désignation de l'appareil
DB9905.DBB192	Appareil 49
DB9905.DBB196	Appareil 50

Pour chaque appareil, quatre octets avec la signification suivante sont utilisés :

Octet	Bit	Description	
0	0	== 1	L'appareil est mis en service (réponse IHM)
	1	== 1	L'appareil doit être activé (demande IHM)
	2	== 1	L'appareil doit être désactivé (demande IHM)
	3-7	réservé	
1	0-7	réservé	
2	0	== 1	L'appareil est actif (réponse AP)
	1	== 1	L'appareil est défectueux
	2-7	réservé	
3	0-7	identification univoque de l'appareil	

Archivage des fichiers

L'archivage des fichiers s'effectue sur la carte CompactFlash du système dans le répertoire "oem" (MANUFACTURER) et "oem_i" (INDIVIDUAL).

Remarque

Le nom du fichier doit seulement contenir des minuscules.

Fichier	Nom	Répertoire cible
Fichier texte	oem_aggregate_xxx.ts	/oem/sinumerik/hmi/lng/ /oem_i/sinumerik/hmi/lng/
Fichier de script	agm.xml	/oem/sinumerik/hmi/dvm /oem_i/sinumerik/hmi/dvm
Fichier d'archivage	quelconque	/oem/sinumerik/hmi/dvm/archives /oem_i/sinumerik/hmi/dvm/archives
Programme utilisateur AP	quelconque	PLC

Procédure générale

Le constructeur de machine doit exécuter les étapes suivantes pour la mise à disposition des données requises :

1. Création d'un programme utilisateur AP, qui permet de commander les appareils par l'AP.
2. Mise en service de la "machine standard" avec sauvegarde consécutive des données dans une archive de MES de série.
3. Montage des appareils, mise en service et lecture consécutive des données sous forme d'archive différentielle de MES de série.

Remarque

Modification de la configuration de la machine

Si la modification de paramètres machine de l'entraînement est requise, ceux-ci doivent d'abord être adaptés dans la commande. Cette procédure doit être répétée pour tous les appareils et constellations.

Ajout d'axes

Si la machine est complétée par des axes machine, il convient de respecter un ordre déterminé lors du rajout des objets entraînement (DO), compte tenu que l'archive de mise en service de série contient la constellation de la machine de référence du constructeur de machine et ne peut pas être utilisée pour un ordre modifié.

Il est recommandé de sélectionner les réglages suivants pour les "composants de commande" :

- Données CN
- Données AP
- Paramètres d'entraînement
  - Format ACX (binaire)

4.3 Représentation sur l'interface utilisateur

Boîtes de dialogue sur l'interface utilisateur

Les boîtes de dialogue suivantes sont disponibles pour la fonction "Easy Extend" :

- La commande offre une **boîte de dialogue configurable** dans laquelle les appareils disponibles sont affichés.
- Si la première mise en service n'a pas encore été effectuée, la commande ouvre la **boîte de dialogue de mise en service**.

Si une procédure de mise en service est programmée pour l'appareil (instruction XML : "START_UP") et que celui-ci n'est pas encore en service, la commande démarre la procédure de mise en service.

Une sauvegarde complète des données est d'abord effectuée, avant que les archives MES de série enregistrées dans le fichier de script ne soient lues.

En cas d'erreur, le responsable de la mise en service peut décider d'annuler celle-ci ou de supprimer manuellement les erreurs éventuelles dans les configurations machine.

- La fonction "Interruption" permet un arrêt prématuré de la mise en service. La commande recopie ensuite les fichiers de mise en service préalablement sauvegardés.

Si la mise en service terminée avec succès requiert l'arrêt de la machine, l'instruction XML "POWER_OFF" permet de programmer l'émission d'une signalisation correspondante sur la commande.

4.4 Création de textes localisés

Structure du fichier texte

Les fichiers xml avec les textes localisés doivent être créés au format UTF8 :

Exemples

oem_aggregate_eng.ts

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
    <name>EASY_EXTEND</name>
    <message>
    <source>DEVICE_ONE</source>
    <translation>Device one</translation>
    </message>
    <message>
    <source>DEVICE_TWO</source>
    <translation>Device two</translation>
    </message>
</context>
</TS>
```

oem_aggregate_deu.ts

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
    <name>EASY_EXTEND</name>
    <message>
    <source>DEVICE_ONE</source>
    <translation>Erstes Gerät</translation>
    </message>
    <source>DEVICE_TWO</source>
    <translation>Zweites Gerät</translation>
    </message>
</context>
</TS>
```

4.5 Exemple d'utilisation pour une partie puissance

Activation de l'objet entraînement

L'objet entraînement à activer a déjà été mis en service et désactivé à nouveau par le constructeur de machine pour pouvoir commercialiser éventuellement l'axe ou les axes en tant qu'option.

Les étapes suivantes doivent être effectuées pour activer l'axe :

- Activer l'objet entraînement via p0105.
- Débloquer le 2e axe dans les paramètres machine du canal.
- Sauvegarder les paramètres machine de l'entraînement via p0971.
- Attendre la fin de l'écriture des données.
- Déclencher le redémarrage du NCK et des entraînements.

Programmation :

```
<DEVICE>
  <list_id>1</list_id>
  <name> "Activer l'entraînement" </name>

  <SET_ACTIVE>
    <data name = "drive/dc/p105[DO5]">1</data>
    <data name = "$MC_AXCONF_MACHAX_USED[4]">5</data>
    <data name = "drive/dc/p971[DO5]">1</data>
    <while>
      <condition> "drive/dc/p971[DO5]" !=0 </condition>
    </while>
    <control_reset resetnc ="true" resetdrive = "true"/>
  </SET_ACTIVE>

  <SET_INACTIVE>
    <data name = "drive/dc/p105[DO5]">0</data>
    <data name = "$MC_AXCONF_MACHAX_USED[4]">0</data>
    <data name = "drive/dc/p971[DO5]">1</data>
    <while>
      <condition> "drive/dc/p971[DO5]" !=0 </condition>
    </while>
    <control_reset resetnc ="true" resetdrive = "true"/>
  </SET_INACTIVE>

</DEVICE>
```

Activation de l'appareil commandé par l'AP

L'appareil est adressé via l'octet de sortie 10 et informe l'AP qu'il est prêt à fonctionner via l'octet d'entrée 9.

L'octet de sortie est forcé sur la valeur du code défini pour l'activation. La boucle WHILE attend ensuite que l'appareil soit prêt à fonctionner.

Programmation :

```
<SET_ACTIVE>
  <DATA name = "plc/qb10"> 8 </DATA>
  <while>
    <condition> "plc/ib9" !=1 </condition>
  </while>
</SET_ACTIVE>
```

4.6 Langue de script

Remarque

Tous les éléments de script décrits dans la fonction Création de boîtes de dialogue utilisateur (Page 27) constituent la base de la fonction "Easy Extend". D'autres éléments de script ont été définis pour la gestion d'aggrégats supplémentaires.

Sections de programme du script

Le script est divisé en secteurs suivants :

- Descripteur "Easy Extend"
- Cadre permettant de définir les actions pouvant être exécutées pour un appareil
- Descripteur pour l'appareil
- Descripteur pour la mise en service de l'appareil
- Descripteur pour l'activation de l'appareil
- Descripteur pour la désactivation de l'appareil
- Descripteur pour le test de l'appareil
- Descripteur pour la boîte de dialogue des paramètres

Les différentes balises sont décrites dans les chapitres suivants.

Description

Descripteur <tag>	Signification
AGM	Descripteur pour la fonction "Assistant MES"
DEVICE	Descripteur pour la description de l'appareil **Attributs :** • **option_bit** Un numéro de bit fixe est affecté à l'appareil pour la gestion des options.
NAME	Le descripteur définit le nom de l'appareil affiché dans la boîte de dialogue. Si une référence de texte est utilisée, la boîte de dialogue indique le texte enregistré pour le descripteur.
START_UP	Le descripteur contient la description des séquences requises pour la mise en service de l'appareil.

Descripteur <tag>	Signification
SET_ACTIVE	Le descripteur contient la description des séquences requises pour l'activation de l'appareil. **Attributs :** • **timeout** L'attribut permet de spécifier un timeout en secondes. Si le script n'est pas encore terminé au bout de ce temps, le système interrompt le traitement.
SET_INACTIVE	Le descripteur contient la description des séquences requises pour la désactivation de l'appareil. **Attributs :** • **timeout** L'attribut permet de spécifier un timeout en secondes. Si le script n'est pas encore terminé au bout de ce temps, le système interrompt le traitement.
TEST	Le descripteur contient les instructions permettant de tester la capacité de fonctionnement d'un appareil. **Attributs :** • **timeout** L'attribut permet de spécifier un timeout en secondes. Si le script n'est pas encore terminé au bout de ce temps, le système interrompt le traitement.
UID	Descripteur numérique univoque pour l'identification de l'appareil dans l'interface AP ↔ IHM.
VERSION	Descripteur pour une version

Acquittement négatif de l'exécution d'une fonction

Avec la variable "$actionresult" mise à disposition automatiquement, le système permet de communiquer un résultat négatif d'exécution à l'analyseur syntaxique XML. Si la valeur est mise à zéro, l'analyseur syntaxique interrompt le traitement de la fonction.

Exemple

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>                              Descripteur pour "Assistant MES"
<DEVICE>
  <NAME> Device 1 </NAME>          Descripteur pour l'appareil
  <START_UP>                       Descripteur pour la mise en service de
  …                                l'appareil
  </START_UP>
  <SET_ACTIVE>                     Descripteur pour l'activation de
  …                                l'appareil
  </SET_ACTIVE>
  <SET_INACTIVE>                   Descripteur pour la désactivation de
  …                                l'appareil
  </SET_INACTIVE>
```

```
    <TEST>                          Descripteur pour le test de l'appareil
    …
    </TEST>
  </DEVICE>
  …
  </AGM>
```

4.6.1 CONTROL_RESET

Description

Ce descripteur permet de redémarrer un ou plusieurs constituants de commande. L'exécution du script ne se poursuit que si la commande a repris le mode cyclique.

Programmation

Descripteur :	**CONTROL_RESET**	
Syntaxe :	<CONTROL_RESET resetnc="TRUE" />	
Attributs :	resetnc="true"	Le constituant de la CN est redémarré.
	resetdrive="true"	Les constituants de l'entraînement sont redémarrés.

4.6.2 FILE

Description

Le descripteur permet la lecture ou la création d'archives standard.

- Lecture d'une archive :
  Pour lire une archive, il convient de spécifier son nom de fichier.
- Créer une archive :
  Si l'attribut create= "true" est spécifié, la fonction crée une archive standard (*.arc) sous le nom spécifié et enregistre le fichier dans le répertoire .../dvm/archives.

Programmation

Descripteur :	**FILE**	
Syntaxe :	<file name ="<Nom d'archive>" />	
	<file name ="<Nom d'archive>" create="true" class="<Classes de données>" group="<Plage>" />	
Attributs :	name	Descripteur pour le nom de fichier

create	Une archive de mise en service est créée sous le nom spécifié dans le répertoire …/dvm/archives/ .
group	Spécifie les groupes de données qui doivent être contenus dans l'archive. Si plusieurs groupes de données doivent être sauvegardés, les groupes doivent être spécifiés séparés par un espace. Les groupes de données suivants peuvent être contenus dans l'archive : • NC • PLC • HMI • DRIVES

Exemple

```
<!-- Créer une archive de classe de données -->
<file name="user.arc" create="true"
group="nc plc hmi" />

<!—Importer une archive dans la commande->
<file name="user.arc" />
```

4.6.3 OPTION_MD

Description

Le descripteur permet de redéfinir le paramètre machine d'une option. Dans l'état à la livraison, le système utilise le PM14510 $MN_USER_DATA_INT[0] jusqu'à $MN_USER_DATA_INT[3].

Si le programme utilisateur de l'AP gère les options, les mots de données adéquats doivent être mis à disposition dans un bloc de données ou dans les données globales utilisateur (GUD).

Le paramètre est organisé bit par bit. En commençant par le bit 0, une affectation fixe des bits est effectuée dans l'ordre de listage des appareils, c'est-à-dire que le bit 0 est affecté à l'appareil 1, le bit 1 est affecté à l'appareil 2, etc. Si plus de 16 appareils doivent être gérés, l'affectation des descripteurs d'adresse des groupes d'appareils 1 à 3 a lieu au moyen de l'indice de plage.

Remarque

Conversion de la plage de valeurs

La plage de valeurs du PM14510 $MN_USER_DATA_INT[i] s'étend de -32768 à +32767. Pour pouvoir débloquer les appareils bit par bit via la boîte de dialogue des paramètres machine, une conversion de la combinaison de bits en notation décimale est requise.

Programmation

Descripteur :	**OPTION_MD**	
Syntaxe :	Plage 0 :	
	<option_md name = "Descripteur d'adresse du paramètre" />	
	OU :	
	<option_md name = "Descripteur d'adresse du paramètre" /> index= "0"/>	
	Plage 1 à 3 :	
	<option_md name = "Descripteur d'adresse du paramètre" index= "Indice de plage"/>	
Attributs :	name	Descripteur pour l'adresse, par ex. $MN_USER_DATA_INT[0]
	index	Descripteur pour l'indice de plage : 0 (par défaut) : Appareil 1 à 16 1: Appareil 17 à 32 2: Appareil 33 à 48 3: Appareil 49 à 64

4.6.4 PLC_INTERFACE

Description

Le descripteur permet de redéfinir l'interface AP ↔ IHM. 128 mots adressables sont attendus par le système.

Remarque

Le descripteur n'est pas prédéfini et doit être défini par l'utilisateur.

Programmation

Descripteur :	**PLC_INTERFACE**	
Syntaxe :	<plc_interface name = "Descripteur d'adresse du paramètre" />	
Attributs :	name	Descripteur pour l'adresse, par ex. "plc/mb170"

Exemple : `plc/mb170`

4.6.5 POWER_OFF

Description

Descripteur pour un message demandant à l'opérateur de mettre hors tension la machine. Le texte du message est enregistré de manière permanente dans le système.

Programmation

Descripteur :	**POWER_OFF**
Syntaxe :	<power_off />
Attributs :	--

4.6.6 WAITING

Description

Après une réinitialisation de la CN ou de l'entraînement, le redémarrage des composants concernés est attendu.

Programmation

Descripteur :	**WAITING**	
Syntaxe :	<WAITING WAITINGFORNC ="TRUE" />	
Attributs :	waitingfornc="true"	En attente de redémarrage de la CN.
	waitingfordrive="true"	En attente de redémarrage de l'entraînement.

4.6.7 Descripteur XML pour la boîte de dialogue

Boîte de dialogue de paramétrage

Pour définir ou générer des paramètres supplémentaires pendant l'exécution, une boîte de dialogue peut être configurée pour chaque appareil. Celle-ci s'affiche en activant la touche logicielle "Paramètres supplémentaires".

Pour la création de la boîte de dialogue, tous les éléments de script décrits au chapitre Création de boîtes de dialogue utilisateur (Page 27) sont disponibles.

Exemple

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>
<DEVICE>
  <NAME> Device 1 </NAME>
  <START_UP>
  …
  </START_UP>
  <SET_ACTIVE>
  …
  </SET_ACTIVE>
…
  <FORM name="<dialog name>">          Descripteur pour une boîte de
                                       dialogue utilisateur
    <INIT>
      <CONTROL name = "edit1" .../>    Descripteur pour un champ de saisie
    </INIT>
    <PAINT>                            Descripteur pour un affichage de
      <TEXT>hello world !</TEXT>       texte ou d'image
    </PAINT>
  </FORM>

</DEVICE>
…
</AGM>
```

4.6.8 SOFTKEY_OK, SOFTKEY_CANCEL

Description

Le descripteur SOFTKEY_OK a priorité sur le comportement standard lors de la fermeture d'une boîte de dialogue au moyen de la touche logicielle "OK". Le descripteur SOFTKEY_CANCEL a priorité sur le comportement standard lors de la fermeture d'une boîte de dialogue au moyen de la touche logicielle "CANCEL".

Les fonctions suivantes peuvent être exécutées à l'intérieur de ce descripteur :

- Manipulations de données
- Traitement conditionnel
- Exécution de boucles

Programmation

Descripteur :	**SOFTKEY_OK**
Syntaxe :	<SOFTKEY_OK> ... </SOFTKEY_OK>
Descripteur :	**SOFTKEY_CANCEL**
Syntaxe :	<SOFTKEY_CANCEL> ... </SOFTKEY_CANCEL>

Index

E

F

G

O

P

R

S

T

V

X