SIEMENS

SINUMERIK

SINUMERIK 840D sl
Easy XML

Manuale di programmazione

Valido per:

Software CNC Versione 4.95

07/2021
6FC5398-5EP40-0CA0

Avvertenze di legge

Concetto di segnaletica di avvertimento

Questo manuale contiene delle norme di sicurezza che devono essere rispettate per salvaguardare l'incolumità personale e per evitare danni materiali. Le indicazioni da rispettare per garantire la sicurezza personale sono evidenziate da un simbolo a forma di triangolo mentre quelle per evitare danni materiali non sono precedute dal triangolo. Gli avvisi di pericolo sono rappresentati come segue e segnalano in ordine descrescente i diversi livelli di rischio.

⚠ **PERICOLO**
questo simbolo indica che la mancata osservanza delle opportune misure di sicurezza **provoca** la morte o gravi lesioni fisiche.

⚠ **AVVERTENZA**
il simbolo indica che la mancata osservanza delle relative misure di sicurezza **può causare** la morte o gravi lesioni fisiche.

⚠ **CAUTELA**
indica che la mancata osservanza delle relative misure di sicurezza può causare lesioni fisiche non gravi.

ATTENZIONE
indica che la mancata osservanza delle relative misure di sicurezza può causare danni materiali.

Nel caso in cui ci siano più livelli di rischio l'avviso di pericolo segnala sempre quello più elevato. Se in un avviso di pericolo si richiama l'attenzione con il triangolo sul rischio di lesioni alle persone, può anche essere contemporaneamente segnalato il rischio di possibili danni materiali.

Personale qualificato

Il prodotto/sistema oggetto di questa documentazione può essere adoperato solo da **personale qualificato** per il rispettivo compito assegnato nel rispetto della documentazione relativa al compito, specialmente delle avvertenze di sicurezza e delle precauzioni in essa contenute. Il personale qualificato, in virtù della sua formazione ed esperienza, è in grado di riconoscere i rischi legati all'impiego di questi prodotti/sistemi e di evitare possibili pericoli.

Uso conforme alle prescrizioni di prodotti Siemens

Si prega di tener presente quanto segue:

⚠ **AVVERTENZA**
I prodotti Siemens devono essere utilizzati solo per i casi d'impiego previsti nel catalogo e nella rispettiva documentazione tecnica. Qualora vengano impiegati prodotti o componenti di terzi, questi devono essere consigliati oppure approvati da Siemens. Il funzionamento corretto e sicuro dei prodotti presuppone un trasporto, un magazzinaggio, un'installazione, un montaggio, una messa in servizio, un utilizzo e una manutenzione appropriati e a regola d'arte. Devono essere rispettate le condizioni ambientali consentite. Devono essere osservate le avvertenze contenute nella rispettiva documentazione.

Marchio di prodotto

Tutti i nomi di prodotto contrassegnati con ® sono marchi registrati della Siemens AG. Gli altri nomi di prodotto citati in questo manuale possono essere dei marchi il cui utilizzo da parte di terzi per i propri scopi può violare i diritti dei proprietari.

Esclusione di responsabilità

Abbiamo controllato che il contenuto di questa documentazione corrisponda all'hardware e al software descritti. Non potendo comunque escludere eventuali differenze, non possiamo garantire una concordanza perfetta. Il contenuto di questa documentazione viene tuttavia verificato periodicamente e le eventuali correzioni o modifiche vengono inserite nelle successive edizioni.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANIA

Numero di articolo del documento: 6FC5398-5EP40-0CA0
Ⓟ 07/2021 Con riserva di modifiche

Indice del contenuto

Introduzione 1

1.1 Informazioni su SINUMERIK

Dalle semplici macchine CNC standard alle macchine standardizzate fino ai concetti di macchine Premium modulari – i controlli CNC SINUMERIK forniscono la soluzione più adatta per ogni concetto di macchina. Sia che si tratti di produzioni di pezzi singoli o di serie, pezzi semplici o complessi – SINUMERIK è la soluzione di automazione altamente produttiva e omogenea per tutti i settori di produzione: dalla prototipazione e costruzione di utensili alla produzione di stampi e di grandi serie.

Per ulteriori informazioni consultare il sito web corrispondente SINUMERIK (https://www.siemens.com/sinumerik).

1.2 Informazioni su questa documentazione

Destinatari

La presente documentazione è rivolta a:

- Programmatori
- Progettisti

Vantaggi

Il Manuale di programmazione consente ai destinatari di progettare interfacce specifiche del cliente e dell'applicazione con elementi di script basati su XML.

Configurazione standard

Nella documentazione presente viene descritta la funzionalità della configurazione standard. L'insieme delle funzionalità descritte nella presente documentazione può discostarsi dalle funzionalità presenti nel sistema fornito. Per le funzionalità del sistema fornito riferirsi esclusivamente alla documentazione per l'ordinazione.

Il sistema può contenere altre funzioni oltre a quelle descritte nella presente documentazione. Ciò non costituisce però obbligo di implementazione di tali funzioni in caso di nuove forniture o di assistenza tecnica.

Per motivi di chiarezza, la presente documentazione non può comprendere tutte le informazioni dettagliate relativa a tutti i tipi di prodotto. La documentazione non può altresì tenere conto di tutti i casi possibili di installazione, funzionamento e manutenzione.

Per le funzionalità aggiuntive o le modifiche apportate dal costruttore della macchina vedere la documentazione pubblicata dal costruttore.

Pagine web di terze parti

Questo documento può contenere collegamenti ipertestuali a pagine web di terze parti. Siemens non si assume la responsabilità per i contenuti di queste pagine web e neppure fa proprie queste pagine ed i relativi contenuti. Siemens non verifica le informazioni di queste pagine web e non è responsabile per i contenuti ed informazioni riportati in esse. Il rischio per il loro uso è a carico dell'utente.

1.3 Documentazione in Internet

1.3.1 Panoramica della documentazione SINUMERIK 840D sl

Una documentazione completa sulle funzioni di SINUMERIK 840D sl a partire dalla versione 4.8 SP4 è riportata in Panoramica della documentazione 840D sl (https://support.industry.siemens.com/cs/ww/en/view/109766213).

I documenti si possono visualizzare oppure scaricare in formato PDF o HTML5.

La documentazione è suddivisa nelle seguenti categorie:

- Utente: Uso
- Utente: Programmazione
- Costruttore/Service: Funzioni
- Costruttore/Service: Hardware
- Costruttore/Service: Progettazione/Messa in servizio
- Costruttore/Service: Safety Integrated
- Costruttore/Service: SINUMERIK Integrate / MindApp
- Informazioni e training
- Costruttore/Service: SINAMICS

1.3.2 Panoramica della documentazione dei componenti operativi SINUMERIK

Una documentazione estesa sui componenti operativi SINUMERIK si trova in Panoramica della documentazione dei componenti operativi SINUMERIK (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW).

I documenti si possono visualizzare oppure scaricare in formato PDF o HTML5.

La documentazione è suddivisa nelle seguenti categorie:

- Pannelli operatore
- Pulsantiere di macchina
- Machine Pushbutton Panel
- Handheld Unit/mini-tastiere operative manuali
- Altri componenti operativi

Una panoramica dei principali documenti, contributi e link sul tema "SINUMERIK" si trova in Panoramica SINUMERIK - Pagina tematica (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW).

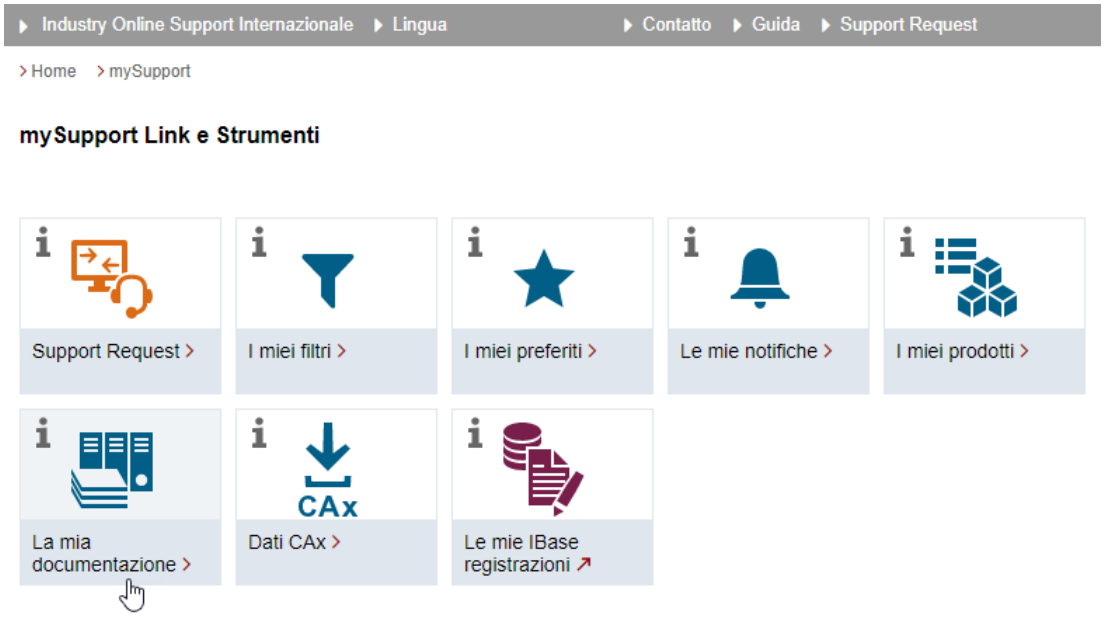
1.4 Feedback relativo alla documentazione tecnica

Per domande, suggerimenti o correzioni relativi alla documentazione tecnica pubblicata nel Siemens Industry Online Support servirsi del link "Inviare feedback" alla fine di ogni contributo.

1.5 Documentazione mySupport

Con il sistema basato su web "Documentazione mySupport" si possono trovare le informazioni per organizzare e personalizzare la vostra documentazione in base ai contenuti Siemens e per adattarla alla propria documentazione di macchina.

Avviare l'applicazione "La mia documentazione" facendo clic sul riquadro della pagina iniziale mySupport (https://support.industry.siemens.com/cs/ww/it/my):



Il manuale configurato può essere esportato in formato RTF, PDF o XML.

Nota

I contenuti Siemens che supportano l'applicazione "Documentazione mySupport" si riconoscono dalla presenza del link "Configurazione".

1.6 Service e Support

Product Support

Ulteriori informazioni sul prodotto sono disponibili in Internet al seguente indirizzo:

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

A questo indirizzo sono disponibili le seguenti informazioni:

- Informazioni aggiornate sul prodotto (comunicazioni sui prodotti)
- FAQ (domande frequenti)
- Manuali
- Download
- La Newsletter con le ultime novità dei prodotti
- Forum internazionale per lo scambio di informazioni ed esperienze tra utenti ed esperti
- La banca dati dei nostri partner di riferimento locali (→ "Contatti")
- Informazioni su assistenza in loco, riparazioni, pezzi di ricambio e molto altro ancora (→ "Servizi")

Technical Support

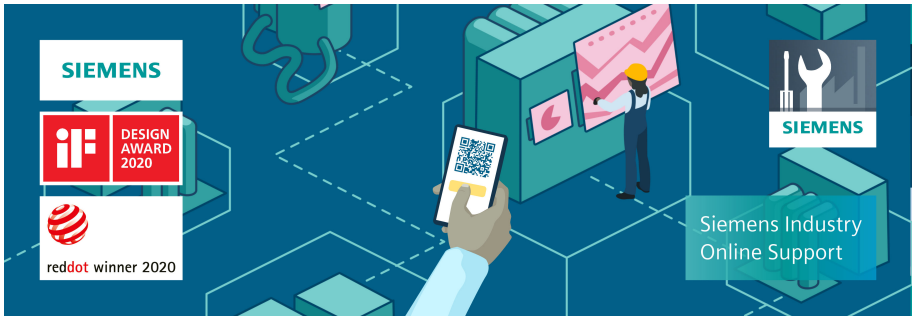
I numeri telefonici della consulenza tecnica specifica dei vari Paesi si trovano in Internet al seguente indirizzo (https://support.industry.siemens.com/cs/ww/it/sc/4868) nella sezione "Contatti".

Per sottoporre una domanda di carattere tecnico, utilizzare il modulo online nella sezione "Support Request".

Training

Al seguente indirizzo (https://www.siemens.com/sitrain) si possono trovare informazioni relative a SITRAIN.
SITRAIN è un programma di formazione per prodotti, sistemi e soluzioni della tecnica di automazione e di azionamento Siemens.

Supporto Siemens in mobilità



Con la premiata app "Siemens Industry Online Support" si può accedere, sempre e ovunque, ad oltre 300.000 documenti sui prodotti Siemens Industry. L'app fornisce supporto, tra l'altro, nei seguenti campi d'impiego:

- Soluzione dei problemi legati alla realizzazione di un progetto
- Rimozione della causa di anomalie
- Estensione o riprogettazione di un impianto

Inoltre si può accedere al Technical Forum e ad altri contributi forniti dai nostri esperti:

- FAQ
- Esempi applicativi
- Manuali
- Certificati
- Comunicazioni sui prodotti e molto altro

L'app "Siemens Industry Online Support" è disponibile per Apple iOS e Android.

Codice Data Matrix sulla targhetta identificativa

Il codice Data Matrix riportato sulla targhetta identificativa contiene i dati specifici dell'apparecchio. Questo codice può essere letto con qualsiasi smartphone; tramite l'app "Industry Online Support" si possono pertanto visualizzare informazioni tecniche dell'apparecchio in questione.

1.7 Informazioni importanti sul prodotto

Impiego di OpenSSL

Questo prodotto può contenere il software seguente:

- Software sviluppato dal progetto OpenSSL e destinato all'uso all'interno del toolkit di OpenSSL.
- Software crittografico creato da Eric Young.
- Software sviluppato da Eric Young.

Ulteriori informazioni sono disponibili in Internet:

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

Conformità al regolamento generale sulla protezione dei dati

Siemens rispetta i principi fondamentali della protezione dei dati, in particolare il principio della minimizzazione dei dati (privacy by design).

Per questo prodotto significa che:

Questo prodotto non elabora e non memorizza dati personali, ma soltanto dati tecnici funzionali (ad es. timbro di data e ora). Nel caso in cui l'utente combini questi dati con altri dati (ad es. tabelle dei turni) o memorizzi dei dati personali sullo stesso supporto dati (ad es. disco rigido), creando quindi un riferimento personale, l'utente è obbligato ad assicurare in proprio il rispetto delle disposizioni di legge sulla protezione dei dati.

Avvertenze di sicurezza di base 2

2.1 Avvertenze di sicurezza generali

⚠ AVVERTENZA

Folgorazione e pericolo di morte dovuti a ulteriori fonti di energia

Il contatto accidentale con parti sotto tensione può causare la morte o gravi lesioni.

- Gli interventi su apparecchiature elettriche devono essere effettuati solo da personale qualificato.
- Per tutti gli interventi rispettare le regole di sicurezza specifiche del Paese.

Come regola generale, al fine di garantire la sicurezza si devono eseguire le operazioni seguenti:

1. Preparare la procedura di disinserzione. Informare tutte le persone interessate dalla procedura.
2. Mettere fuori tensione il sistema di azionamento e assicurarlo contro la reinserzione.
3. Attendere che sia trascorso il tempo di scarica indicato sulle targhette di avviso.
4. Verificare l'assenza di tensione reciproca su tutti i collegamenti di potenza e rispetto alla connessione del conduttore di terra.
5. Verificare che i circuiti di tensione ausiliaria presenti siano privi di tensione.
6. Accertarsi che i motori non possano muoversi.
7. Identificare tutte le altre fonti di energia pericolose, come ad es. aria compressa, forza idraulica o acqua. Mettere le fonti di energia in uno stato sicuro.
8. Accertarsi che il sistema di azionamento corretto sia completamente bloccato.

Una volta conclusi gli interventi necessari, ripristinare lo stato di pronto al funzionamento ripetendo le stesse operazioni nella sequenza inversa.

⚠ AVVERTENZA

Scossa elettrica in caso di collegamento di un'alimentazione di corrente inadatta

Il collegamento di un'alimentazione di corrente inadatta può mettere sotto tensione pericolosa parti con cui si può entrare in contatto. Il contatto con una tensione pericolosa può provocare lesioni gravi o la morte.

- Per tutti i connettori e i morsetti dei gruppi elettronici utilizzare solo alimentatori che forniscono tensioni di uscita SELV (Safety Extra Low Voltage) o PELV (Protective Extra Low Voltage).

⚠ AVVERTENZA

Folgorazione in caso di apparecchiature danneggiate

Ogni manipolazione impropria può danneggiare le apparecchiature. In caso di apparecchiature danneggiate possono essere presenti tensioni elevate sull'involucro o su componenti aperti, il cui contatto può causare lesioni gravi o la morte.

- Durante il trasporto, l'immagazzinaggio e l'esercizio rispettare i valori limite specificati nei dati tecnici.
- Non utilizzare apparecchiature danneggiate.

⚠ AVVERTENZA

Folgorazione in caso di schermi dei cavi non installati

La diafonia capacitiva o può generare tensioni di contatto letali in caso di schermi dei cavi non installati.

- Collegare almeno su un lato al potenziale di terra della custodia le maglie di schermatura e i fili non utilizzati dei cavi.

⚠ AVVERTENZA

Folgorazione in caso di messa a terra mancante

Se la connessione del conduttore di protezione di apparecchi della classe di protezione I manca o è eseguita in modo errato, possono essere presenti tensioni elevate su componenti aperti, il cui contatto può causare lesioni gravi o la morte.

- Mettere a terra l'apparecchio conformemente alle norme.

ATTENZIONE

Danni all'apparecchio dovuti a utensili di serraggio inadeguati

Utensili o metodi di serraggio inadeguati possono danneggiare le viti dell'apparecchio.

- Utilizzare avvitatori che si adattano perfettamente alla testa della vite.
- Serrare le viti con la coppia specificata nella documentazione tecnica.
- Utilizzare una chiave dinamometrica o un cacciavite meccanico di precisione con sensore torsiometrico dinamico e limitazione del numero di giri.

⚠ AVVERTENZA

Propagazione di incendio negli apparecchi da incasso

In caso di incendio, gli involucri degli apparecchi da incasso non possono impedire la fuoriuscita di fiamme e fumo. Ne possono derivare gravi danni alle persone o alle cose.

- Installare gli apparecchi da incasso in un armadio metallico idoneo oppure adottare un altro provvedimento analogo per proteggere le persone dal fumo e dal fuoco in caso di incendio.
- Accertarsi che il fumo possa essere evacuato solo lungo percorsi controllati.

⚠ AVVERTENZA

Movimento inaspettato delle macchine dovuto ad apparecchiature radio o a telefoni cellulari

L'utilizzo di apparecchiature radio o di telefoni cellulari nelle immediate vicinanze dei componenti può causare malfunzionamenti degli apparecchi. I malfunzionamenti possono influire sulla sicurezza funzionale delle macchine e costituiscono pertanto un pericolo per le persone o per le cose.

- Spegnere le apparecchiature radio o i telefoni cellulari se ci si trova a meno di 20 cm circa dai componenti.
- Utilizzare la "SIEMENS Industry Online Support App" solo con l'apparecchio spento.

⚠ AVVERTENZA

Incendio dovuto a spazi di ventilazione insufficienti

Se gli spazi liberi di ventilazione sono insufficienti, può verificarsi un surriscaldamento dei componenti con conseguente pericolo di incendio e sviluppo di fumo. Ne possono conseguire la morte o gravi lesioni. Inoltre le apparecchiature e i sistemi possono avere un tasso di guasti maggiore e una durata di vita inferiore.

- Rispettare le distanze minime per gli spazi liberi di ventilazione del rispettivo componente.

ATTENZIONE

Surriscaldamento in caso di posizione di montaggio non consentita

Una posizione di montaggio non consentita può causare il surriscaldamento dell'apparecchio e quindi il suo danneggiamento.

- Fare funzionare l'apparecchio solo nelle posizioni di montaggio consentite.

⚠ AVVERTENZA

Movimenti imprevisti delle macchine dovuti a funzioni di sicurezza inattive

Funzioni di sicurezza inattive o non adattate possono causare movimenti imprevisti delle macchine, con pericolo di gravi lesioni o di morte.

- Prima della messa in servizio leggere attentamente le informazioni nella relativa documentazione del prodotto.
- Per le funzioni rilevanti per la sicurezza eseguire un controllo di sicurezza del sistema completo, inclusi tutti i componenti rilevanti.
- Accertarsi con un'opportuna parametrizzazione che le funzioni di sicurezza applicate siano attivate e adatte al compito di azionamento e di automazione specifico.
- Eseguire un test funzionale.
- Utilizzare l'impianto in modo produttivo solo dopo aver verificato l'esecuzione corretta delle funzioni rilevanti per la sicurezza.

Nota

Avvertenze di sicurezza importanti relative alle funzioni Safety Integrated

Se si desidera utilizzare le funzioni Safety Integrated, rispettare le avvertenze di sicurezza contenute nei manuali Safety Integrated.

⚠ AVVERTENZA

Malfunzionamenti della macchina dovuti a parametrizzazione errata o modificata

La parametrizzazione errata o modificata può provocare malfunzionamenti delle macchine e di conseguenza il rischio di morte o gravi lesioni.

- Proteggere la parametrizzazione da ogni accesso non autorizzato.
- Gestire eventuali malfunzionamenti con provvedimenti adeguati, ad es. ARRESTO DI EMERGENZA oppure OFF DI EMERGENZA.

2.2 Danni alle apparecchiature causati da campi elettrici o scariche elettrostatiche

I componenti esposti a pericolo elettrostatico (ESD, Electrostatic Sensitive Device) sono componenti singoli, circuiti integrati, unità o dispositivi che possono essere danneggiati da campi o scariche elettrostatiche.

ATTENZIONE

Danni alle apparecchiature causati da campi elettrici o scariche elettrostatiche

I campi elettrici o le scariche elettrostatiche possono danneggiare singoli componenti, circuiti integrati, unità o dispositivi e quindi causare danni funzionali.

- Per l'imballaggio, l'immagazzinaggio, il trasporto e la spedizione dei componenti, delle unità o dei dispositivi utilizzare solo l'imballaggio originale o altri materiali adatti come ad es. gommapiuma conduttiva o pellicola di alluminio.
- Prima di toccare i componenti, le unità o i dispositivi occorre adottare uno dei seguenti provvedimenti di messa a terra:
  - Indossare un bracciale ESD
  - Indossare scarpe ESD o fascette ESD per la messa a terra nelle aree ESD con pavimento conduttivo
- Appoggiare i componenti elettronici, le unità o gli apparecchi solo su supporti conduttivi (tavoli con rivestimento ESD, materiale espanso ESD conduttivo, sacchetti per imballaggio ESD, contenitori di trasporto ESD).

2.3 Garanzia e responsabilità per gli esempi applicativi

Gli esempi applicativi non sono vincolanti e non hanno alcuna pretesa di completezza per quanto riguarda configurazione ed equipaggiamento o altre eventualità. Essi non rappresentano soluzioni specifiche dei clienti, ma intendono solo proporre un aiuto per la risoluzione di compiti tipici.

L'utente stesso è responsabile del corretto funzionamento dei prodotti descritti. Gli esempi applicativi non esonerano dall'obbligo di cautela nell'impiego, nell'installazione, nell'esercizio e nella manutenzione.

2.4 Avvertenze di sicurezza

Siemens commercializza prodotti e soluzioni dotati di funzioni di Industrial Security che contribuiscono al funzionamento sicuro di impianti, soluzioni, macchine e reti.

Al fine di proteggere impianti, sistemi, macchine e reti da minacce cibernetiche, è necessario implementare - e mantenere continuamente – un concetto di Industrial Security globale ed all'avanguardia. I prodotti e le soluzioni Siemens costituiscono soltanto una componente di questo concetto.

È responsabilità dei clienti prevenire accessi non autorizzati ai propri impianti, sistemi, macchine e reti. Tali sistemi, macchine e componenti dovrebbero essere connessi unicamente a una rete aziendale o a Internet se e nella misura in cui detta connessione sia necessaria e solo quando siano attive appropriate misure di sicurezza (ad es. impiego di firewall e segmentazione della rete).

Per ulteriori informazioni relative a misure di Industrial Security implementabili potete visitare il sito
https://www.siemens.com/industrialsecurity (https://www.siemens.com/industrialsecurity).

I prodotti e le soluzioni Siemens vengono costantemente perfezionati per incrementarne la sicurezza. Siemens raccomanda espressamente che gli aggiornamenti dei prodotti siano effettuati non appena disponibili e che siano utilizzate le versioni più aggiornate. L'utilizzo di versioni di prodotti non più supportate ed il mancato aggiornamento degli stessi incrementa il rischio di attacchi cibernetici.

Per essere informati sugli aggiornamenti dei prodotti, potete iscrivervi a Siemens Industrial Security RSS Feed al sito
https://www.siemens.com/industrialsecurity (https://new.siemens.com/global/en/products/services/cert.html#Subscriptions).

Ulteriori informazioni sono disponibili in Internet:

Manuale di progettazione Industrial Security (https://support.industry.siemens.com/cs/ww/it/view/108862708/en)

⚠ AVVERTENZA

Stati operativi non sicuri dovuti a manipolazione del software

Qualsiasi alterazione del software, come ad es. virus, cavalli di Troia, malware o bug, può provocare stati operativi non sicuri dell'impianto e comportare il rischio di morte, lesioni gravi e danni materiali.

- Mantenere aggiornato il software.
- Integrare i componenti di automazione e azionamento in un concetto di Industrial Security globale all'avanguardia dell'impianto o della macchina.
- Tutti i prodotti utilizzati vanno considerati nell'ottica di questo concetto di Industrial Security globale.
- Adottare le opportune contromisure per proteggere i file sui supporti di memoria rimovibili da eventuali software dannosi, ad es. installando un programma antivirus.
- Al termine della messa in servizio, verificare le impostazioni rilevanti ai fini della sicurezza.

2.5 Rischi residui di sistemi di azionamento (Power Drive System)

Nell'ambito della valutazione dei rischi della macchina o dell'impianto, da eseguire conformemente alle prescrizioni locali (ad es. Direttiva Macchine CE), il costruttore della macchina o dell'impianto deve considerare i seguenti rischi residui derivanti dai componenti impiegati per il controllo e l'azionamento di un sistema di azionamento:

1. Movimenti incontrollati di parti motorizzate della macchina o dell'impianto durante la messa in servizio, il funzionamento, la manutenzione e la riparazione, ad es. a causa di:
   - Errori hardware e/o software nei sensori, nel controllore, negli attuatori e nella tecnica di collegamento
   - Tempi di reazione del controllo e dell'azionamento
   - Funzionamento e/o condizioni ambientali fuori specifica
   - Condensa / imbrattamenti conduttivi
   - Errori durante la parametrizzazione, la programmazione, il cablaggio e il montaggio
   - Utilizzo di apparecchiature radio / telefoni cellulari nelle immediate vicinanze di componenti elettronici
   - Influenze esterne / danneggiamenti
   - Raggi X, radiazioni ionizzanti e radiazioni da raggi cosmici secondari
2. In caso di guasto possono verificarsi temperature eccezionalmente elevate, incluso fuoco aperto, all'interno e all'esterno dei componenti, nonché emissioni di luce, rumore, particelle, gas ecc., ad esempio a causa di:
   - Guasto di componenti
   - Errori software
   - Funzionamento e/o condizioni ambientali fuori specifica
   - Influenze esterne / danneggiamenti
3. Tensioni di contatto pericolose, ad es. a causa di:
   - Guasto di componenti
   - Influenza in caso di cariche elettrostatiche
   - Induzione di tensioni con motori in movimento
   - Funzionamento e/o condizioni ambientali fuori specifica
   - Condensa / imbrattamenti conduttivi
   - Influenze esterne / danneggiamenti
4. Campi elettrici, magnetici ed elettromagnetici in condizioni di esercizio che, ad esempio, possono essere pericolosi per portatori di pacemaker, impianti od oggetti metallici in caso di distanza insufficiente
5. Rilascio di sostanze ed emissioni dannose per l'ambiente in caso di utilizzo non appropriato e/o smaltimento non corretto dei componenti
6. Interferenze di sistemi di comunicazione in rete, ad es. trasmettitori centralizzati o trasmissione dati in rete.

Per ulteriori informazioni sui rischi residui derivanti dai componenti di un sistema di azionamento, consultare i rispettivi capitoli della documentazione tecnica per l'utente.

Creazione finestre di dialogo utente 3

3.1 Dotazione funzionale

Panoramica

La funzione "Creazione finestre di dialogo utente" garantisce un'apertura che consente all'utente di progettare in SINUMERIK Operate interfacce specifiche per cliente e per applicazione.

Per creare finestre di dialogo utente il controllore offre un linguaggio script basato su XML.

Questo linguaggio script consente di visualizzare menu specifici per la macchina e finestre di dialogo in SINUMERIK Operate nel settore operativo <CUSTOM>.

Tutte le finestre di dialogo possono essere realizzate indipendentemente dalla lingua. In questo caso il sistema legge il testo da visualizzare dal database di lingue fornito.

Utilizzo

Le istruzioni XML definite consentono le seguenti proprietà:

1. Visualizzazione di finestre di dialogo e messa a disposizione di:
   - softkey
   - variabili
   - testo e testo di help
   - grafici e figure di help
2. Richiamo delle finestre di dialogo mediante:
   - azionamento del softkey (di accesso)
3. Ristrutturazione dinamica delle finestre di dialogo:
   - modifica e cancellazione di softkey
   - definizione e strutturazione dei campi di variabili
   - visualizzazione, sostituzione, cancellazione di testi visualizzati (dipendenti o non dipendenti dalla lingua)
   - visualizzazione, sostituzione, cancellazione di grafici
   - creazione dinamica di parti di script eseguibili e integrazione delle stesse nell'elaborazione dello script
4. Avvio di azioni mediante:
   - visualizzazione di finestre di dialogo
   - immissione di valori (variabili)
   - azionamento di softkey
   - uscita dalle finestre di dialogo

5. Scambio di dati tra finestre di dialogo
6. variabili
   – lettura (variabili NC, PLC, utente)
   – scrittura (variabili NC, PLC, utente)
   – combinazione logica con operatori matematici, comparativi o logici
7. Esecuzione di funzioni:
   – sottoprogrammi
   – funzioni file
   – servizi PI
8. Considerazione dei livelli di protezione secondo i gruppi di utenti

Gli elementi validi (tag) per il linguaggio script sono descritti nel capitolo "Tag XML" (Pagina 46).

Nota

La sezione seguente "Principi di programmazione" non pretende di essere esaustiva riguardo alla descrizione del linguaggio XML (Extensible Markup Language). Per maggiori informazioni si rimanda alla letteratura specifica.

3.2 Fondamenti per la progettazione

File di progettazione

La definizione di nuove interfacce operative viene salvata nei file di progettazione. Questi file sono interpretati automaticamente e il risultato è visualizzato sullo schermo. I file di progettazione non sono forniti e devono quindi essere creati o caricati a parte.

Per creare i file di progettazione può essere utilizzato un editor XML o un altro editor di testo.

Nota

I nomi del file possono contenere solo lettere minuscole.

Principio della struttura di menu

Più finestre di dialogo collegate tra loro costituiscono una struttura di menu. È presente un collegamento quando è possibile passare da una finestra di dialogo a un'altra. Attraverso nuovi softkey orizzontali o verticali definiti all'interno della finestra di dialogo è possibile passare alla finestra di dialogo precedente o a un'altra a scelta.

Dopo il menu di accesso è possibile creare un'altra struttura di menu tramite softkey di accesso progettati:

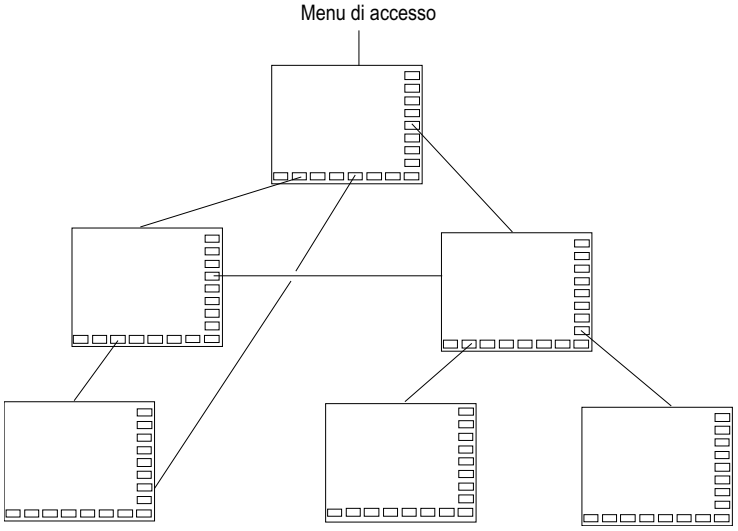


Figura 3-1 Struttura di menu delle finestre di dialogo utente

Menu di accesso

Nel file "xmldial.xml" il menu di accesso viene definito con il nome "main". Il menu di accesso è il punto di partenza di alcuni processi operativi.

Al menu "main" può essere collegato il caricamento di proprie finestre di dialogo o di altre barre di softkey che consentono di eseguire ulteriori azioni.

La figura seguente mostra la directory del costruttore "System CF-Card/oem/sinumerik/hmi" sul controllore.

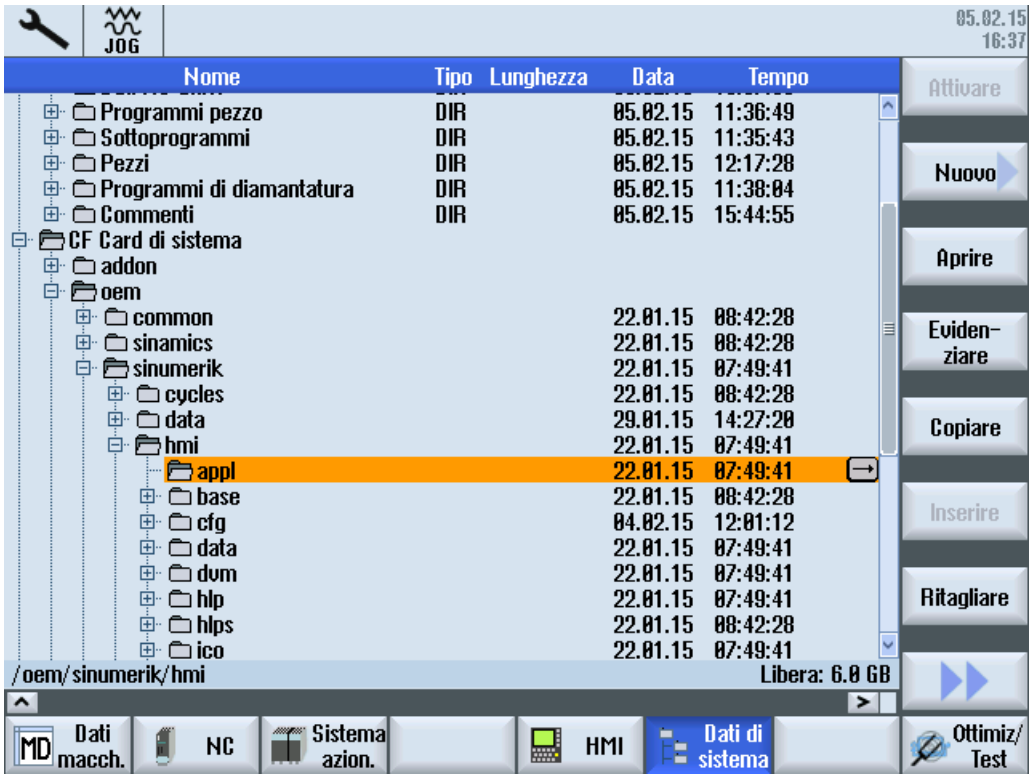


Figura 3-2 Directory costruttore

Per la configurazione di finestre di dialogo utente, nel controllore devono essere presenti i seguenti file nella directory costruttore "System CF-Card/oem/sinumerik/hmi":

Tabella 3-1 File per la configurazione

Tipo di file	Nome del file	Significato	Percorso di salvataggio nel settore operativo Messa in servizio > Dati di sistema
File di script	"xmldial.xml"	File di testo standard Questo file di script gestisce tramite i tag XML l'immagine dei menu softkey e delle finestre di dialogo progettati in SINUMERIK Operate nel settore operativo <CUSTOM>.	Directory costruttore > sottodirectory "appl" per le applicazioni
File di script specifico dell'applicazione	"xxx.xml"	Nome file liberamente selezionabile. Il riferimento a questo nome viene stabilito attraverso i rispettivi file INI.	

Tipo di file	Nome del file	Significato	Percorso di salvataggio nel settore operativo Messa in servizio > Dati di sistema
File di testo	"oem_xml_screens_xxx.ts"	File di testo standard per il settore operativo Customer Questo file di testo contiene i testi per i menu e le finestre di dialogo per le singole lingue.	Directory costruttore > Sottodirectory "lng" per le lingue desiderate
File di testo specifico dell'utente	"yyy_xxx.ts"	Nome file liberamente selezionabile. Il riferimento a questo nome viene stabilito attraverso i rispettivi attributi con i tag DialogGUI, Menu o Form. Per la descrizione vedere il capitolo "File di testo specifici del progetto (Pagina 260)". Nelle applicazioni Sidescreen o Display Manager il nome del file di testo viene ricavato dal nome della pagina o del widget. Un esempio si trova nel capitolo "Gestione della lingua e dei testi (Pagina 236)".	
Bitmap		Il controllore supporta i formati BMP e PNG.	Directory costruttore > sottodirectory "ico"
File XML che vengono inseriti nel file di controllo "xmldial.xml" con il tag XML "INCLUDE".	Ad es. "machine_settings.xml"	Questi file contengono anche istruzioni programmate per la rappresentazione di finestre di dialogo e parametri in SINUMERIK Operate.	Directory costruttore > sottodirectory "appl" per le applicazioni

Punti di collegamento Easy XML

Per gli script Easy XML esistono i seguenti punti di collegamento:

Hardkey/softkey	Punto di collegamento
Softkey Operating Menu	Nome file di script standard "xmldial.xml" Si attiva nel file "slamconfig.ini" **Esempio:** `[CustomXML]` `TextId=SL_AM_CUSTOM` `SidescreenTextId=SL_SIDESCREEN_CUSTOM` `TextFile=slam` `TextContext=SlAmAreaMenu` `SoftkeyPosition=12` `Visible=true`
Menu User	Nome file di script standard "menu_user.xml"
Menu Function	Nome file di script standard "menu_function.xml"

Hardkey/softkey	**Punto di collegamento**
Hardkey PLC	Il nome del file di script si riferisce alla descrizione del tasto. **Esempio:** `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"`
Comando MMC	Il nome del file di script si riferisce alla descrizione del comando NC. **Esempio:** `MMC("POPUPDLG, XML_ON, TEST.XML, cmd1","A)`

Hardkey/softkey	**Punto di collegamento**
Softkey riservati di un settore operativo	Per incorporare un'applicazione OEM in un settore operativo si possono usare i vari modelli presenti nella directory siemens\sinumerik\hmi\template\cfg\ (sl<settore_operativo>_oem.xml) Il file XML appartenente al settore operativo va copiato nella directory oem\sinumerik\hmi\cfg e modificato come segue: Come reazione del softkey si deve programmare la funzione **switchToDialog**. Come argomenti si devono specificare le parole chiave **area** con il valore **CustomXML** e **dialog** con il valore **SlEECustomDialog**. **Sintassi:** `<FUNCTION args="-area CustomXML -dialog SlEECustomDialog -mainModule <name of the main module> -entry <menu name>" name="switchToDialog"/>` **Esempio:** Integrazione di un'applicazione Easy XML nel settore di diagnostica: Come file di script è stato utilizzato "dg.xml". In questo file deve essere richiamato il menu con il nome **main**. `<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	Il nome del file di script si riferisce alla descrizione del sidescreen del file "slsidescreen.ini".
Softkey Easy Extend	Nome file di script standard "agm.xml"

3.3 File di progettazione

Introduzione

La figura seguente rappresenta la directory costruttore "System CF-Card/oem/sinumerik/hmi" sul controllore.

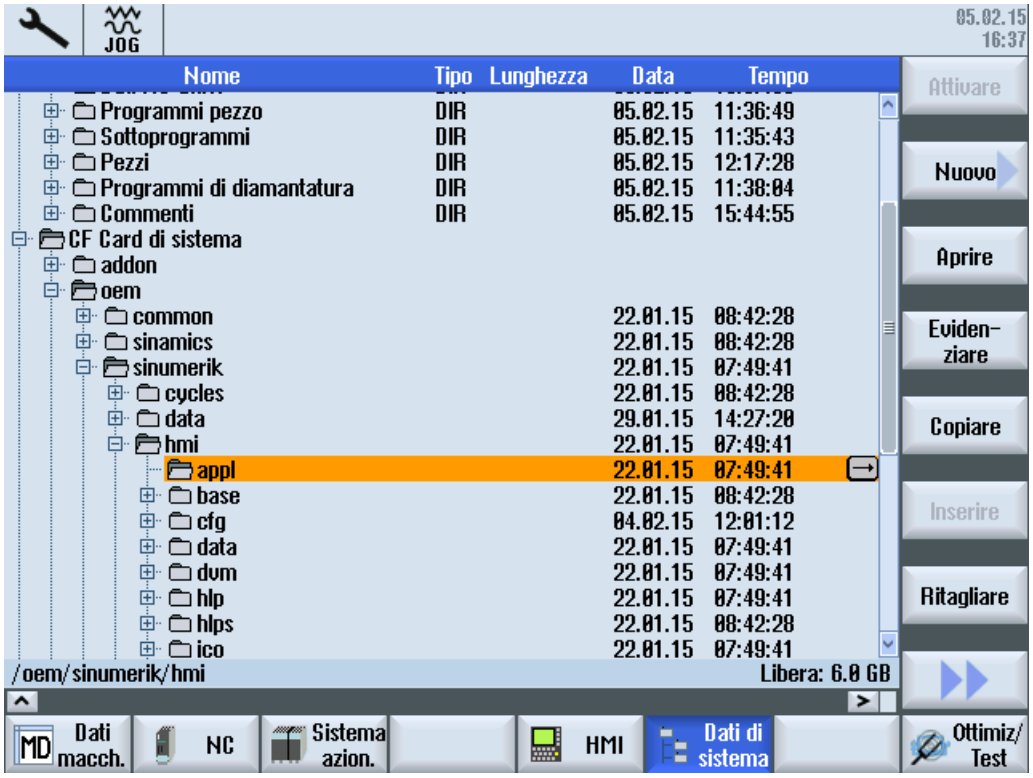


Figura 3-3 Directory costruttore

Per la configurazione di finestre di dialogo utente, nel controllo numerico devono essere presenti i seguenti file nella directory costruttore "System CF-Card/oem/sinumerik/hmi":

Tabella 3-2 File per la configurazione

Tipo di file	Nome del file	Significato	Percorso di archiviazione nel settore operativo Messa in servizio > Dati di sistema
File di script	"xmldial.xml"	Questo file di script gestisce tramite i tag XML l'immagine dei menu softkey e delle finestre di dialogo progettati in SINUMERIK Operate nel settore operativo <CUSTOM>.	Directory costruttore > sottodirectory "appl" per le applicazioni
File di testo	"oem_xml_screens_xxx.ts"	Questo file di testo contiene i testi per i menu e le finestre di dialogo per le singole lingue.	Directory costruttore > sottodirectory "lng" per le lingue

Tipo di file	Nome del file	Significato	Percorso di archiviazione nel settore operativo Messa in servizio > Dati di sistema
Bitmap		Il controllo numerico supporta i formati BMP e PNG.	Directory costruttore > sottodirectory "ico" I bitmap vengono salvati nelle sottodirectory per la risoluzione dello schermo propria del controllore. Nota: se sul file bitmap viene specificato un percorso, i file possono essere salvati direttamente in questa directory.
File XML che vengono inseriti nel file di controllo "xmldial.xml" con il tag XML "INCLUDE".	ad es. "machine_settings.xml"	Questi file contengono anche istruzioni programmate per la rappresentazione di finestre di dialogo e parametri in SINUMERIK Operate.	Directory costruttore > sottodirectory "appl" per le applicazioni

Dipendenze dei file per la configurazione delle finestre di dialogo utente

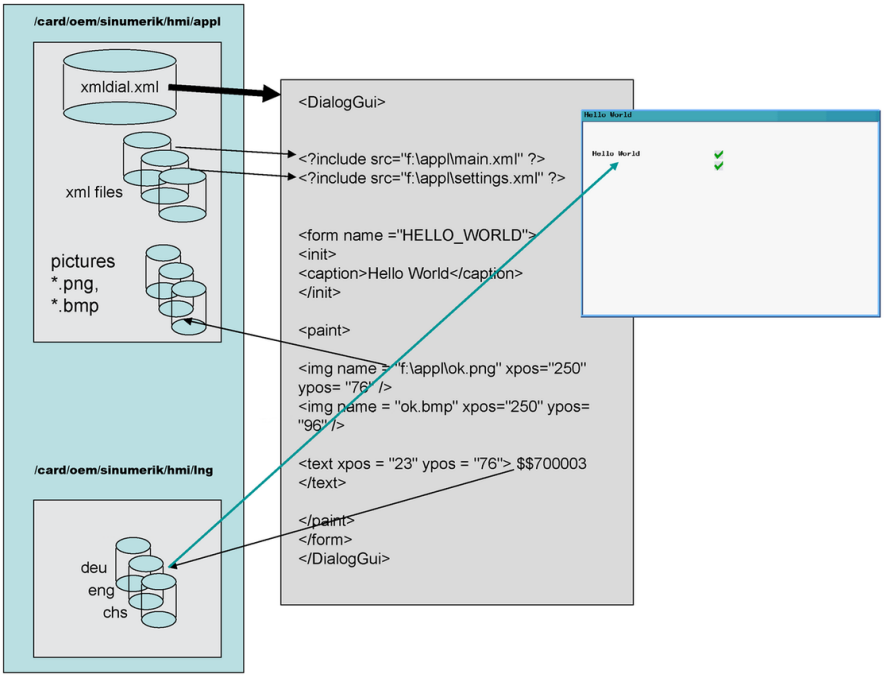


Figura 3-4 Dipendenze

Caricamento della configurazione

Come descritto nella tabella precedente "File per la configurazione" nella colonna "Percorso di archiviazione nel settore operativo", i file creati devono essere copiati nelle sottodirectory corrispondenti all'interno della directory costruttore.

Nota

Appena nella sottodirectory per le applicazioni si trova un file di script "xmldial.xml", l'utente può avviare questa finestra di dialogo utente nel settore operativo <CUSTOM>.

Dopo la prima copia occorre effettuare un reset dell'HMI con "Avvio normale".

Esempio di una finestra di dialogo utente in SINUMERIK Operate

Al richiamo del settore operativo <CUSTOM> vengono visualizzati i menu softkey progettati. L'utente può inoltre usare le finestre di dialogo progettate.

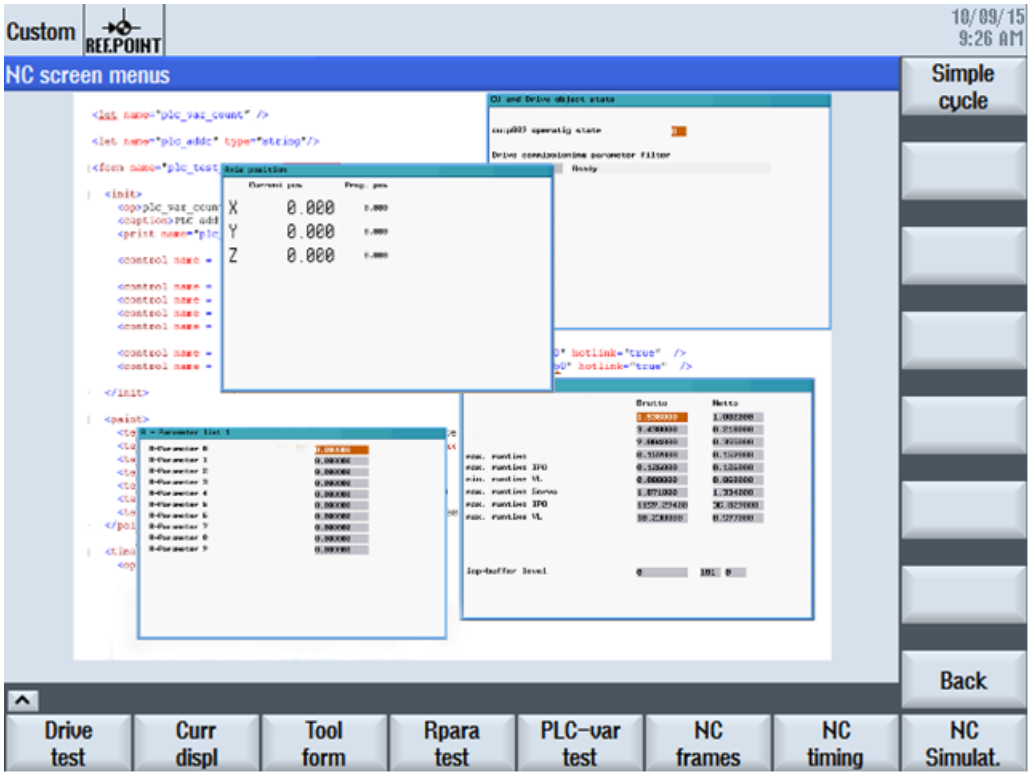


Figura 3-5 Esempio di finestra di dialogo utente nel settore operativo <CUSTOM>

Altri esempi si possono trovare nella Toolbox.

Vedere anche

Funzioni predefinite (Pagina 158)

3.4 Struttura del file di progettazione

Panoramica

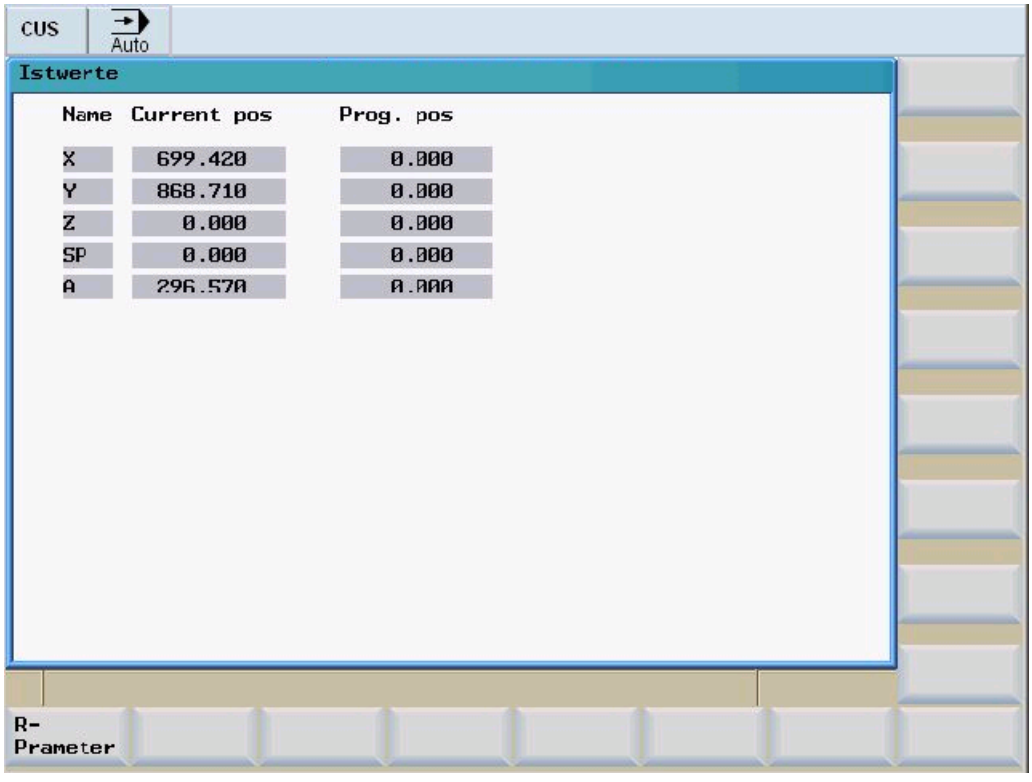
Un file di progettazione è costituito dai seguenti elementi:

- Descrizione del menu di accesso "main" con softkey di accesso
- Definizione delle finestre di dialogo
- Definizione delle variabili
- Descrizione dei blocchi
- Definizione delle barre dei softkey

Gli esempi seguenti rappresentano lo script XML del file "xmldial.xml" e gli screenshot corrispondenti.

Lo script contiene le finestre di dialogo per la visualizzazione di valori reali e percorsi residui, oltre a una lista di parametri R.

Sezione maschere di dialogo, valori reali



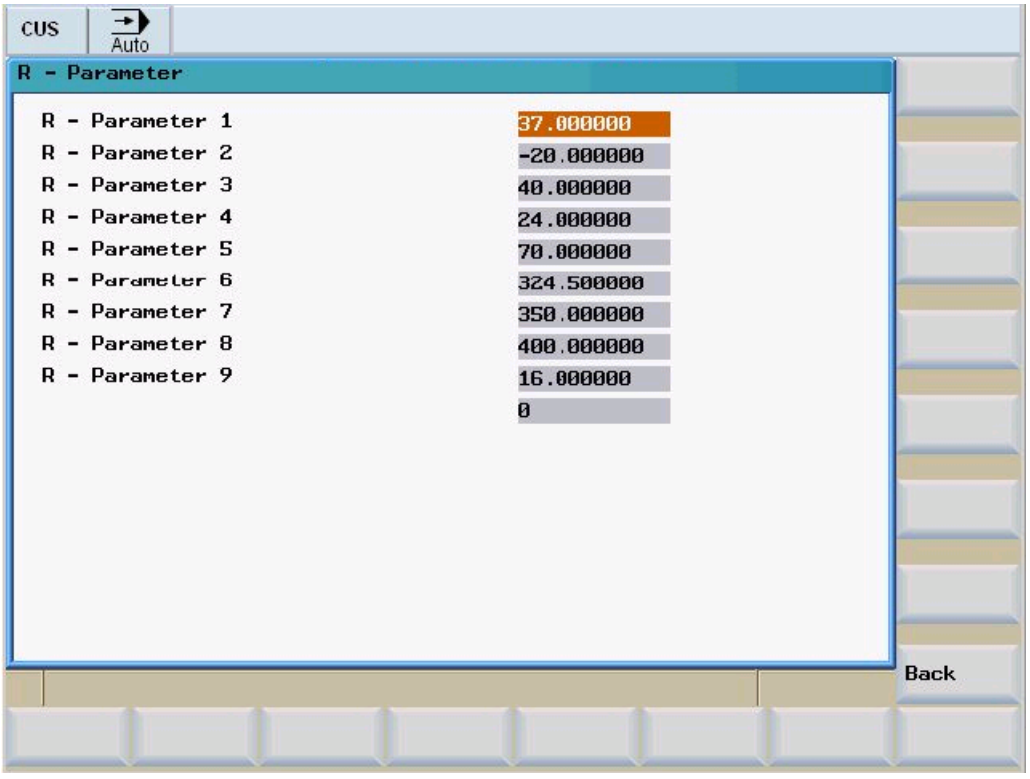
xmldial.xml

```
<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>

.....
```

Sezione maschere di dialogo, parametri 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 Dipendenza dalla lingua

I testi dipendenti dalla lingua vengono utilizzati per:

- Diciture dei softkey
- Intestazioni
- Testi di aiuto
- Altri testi qualsiasi

I testi dipendenti dalla lingua sono memorizzati in file di testo.

Nota

Quando si usano questi file di testo occorre effettuare le seguenti operazioni:

- Mettere a disposizione i file nelle lingue necessarie.
- Copiare i file nelle directory delle lingue corrispondenti del controllore.

3.6 Diagnostica XML

Per la diagnostica e il rilevamento di errori di script il sistema propone la funzione Diagnostica Easy XML.

L'applicazione della funzione di diagnostica verifica la sintassi XML ed esamina tutti gli script appartenenti al progetto. A questo scopo vengono esaminate anche funzioni, menu e form, nonché l'esistenza e la validità delle variabili. Qui sono elencati gli errori che si sono verificati.

La funzione Diagnostica Easy XML può essere attivata con un attributo nel tag **DialogGui** o tramite dato macchina di visualizzazione.

Attivazione della Diagnostica Easy XML

Esempio:

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

oppure

Dato macchina di visualizzazione:

MD9113 $MM_EASY_XML_DIAGNOSE		Supporto di diagnostica e correzione per script Easy XML
= 0	Nessuna diagnostica attiva	
= 1	Verifica della sintassi attiva	
= 0x100	I messaggi vengono inoltre salvati nel file error_log.txt.	

Diagnostica mediante output Trace

Gli esiti di un Trace si possono integrare in uno script con il tag **debug**. La memorizzazione avviene solo se è specificato l'attributo **debug_msg** con il valore **on** nel tag **DialogGui**.

Panoramica delle funzioni dei softkey

Finestra di dialogo	Softkey	Funzione
Menu principale "Diagnostica EasyXML"	Avvio script	Il progetto caricato viene avviato direttamente.
	Avvio verifica	La verifica della sintassi viene avviata. Tutti i messaggi generati possono essere visualizzati. È possibile ripetere la verifica.

Finestra di dialogo	Softkey	Funzione
Finestre di dialogo "Errori" e "Avvisi"	Errori	Il risultato viene visualizzato ordinato in base agli errori o agli avvisi.
	Avvisi	
	Vai a errore	Nel caso in cui vengano rilevati errori o avvisi, viene visualizzato questo softkey. Il softkey apre il file .xml selezionato dalla Lista degli errori per la modifica. Il cursore viene posizionato automaticamente sulla riga errata.
	Risultato verifica	Tutti i messaggi generati possono essere visualizzati.
Finestra di dialogo "Documento"	Salvare	Le modifiche del file XML vengono salvate.
	Interruz.	Il file XML viene chiuso senza salvare le modifiche. Viene nuovamente visualizzato il risultato della verifica.

3.7 Identificatori XML

3.7.1 Struttura generale

Struttura e istruzioni dei file di script per la progettazione delle finestre di dialogo

Tutte le progettazioni delle finestre di dialogo devono essere memorizzate tra tag **DialogGui**.

```
<DialogGui>
…
</DialogGui>
```

Esempio:

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

</DialogGui>
```

Istruzioni

Per l'elaborazione di istruzioni condizionate e di loop di programma, la lingua offre le seguenti istruzioni:

- Loop For
- Loop While
- Loop Do-While
- Elaborazione condizionata
- Istruzioni Switch e Case
- Elementi di comando in una finestra di dialogo
- Descrizioni dei softkey
- Definizione delle variabili

Una descrizione dettagliata dell'uso delle istruzioni si trova nel capitolo "Descrizioni di istruzioni/ identificatori (Pagina 47)":

3.7.2 Descrizioni di istruzioni/identificatori

Per la creazione di finestre di dialogo e menu e per l'elaborazione di sequenze di programma sono definiti i seguenti **tag XML**:

Nota

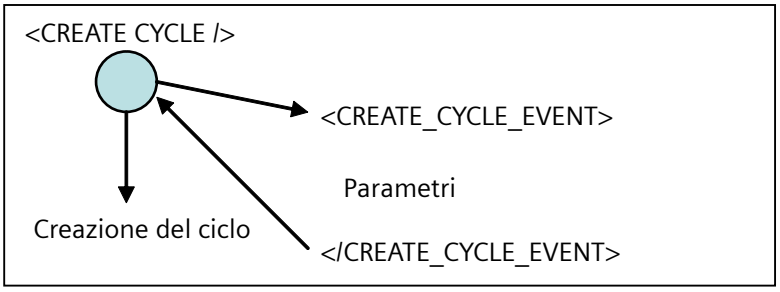
I valori degli attributi contrassegnati con "<...>" tra virgolette devono essere sostituiti con le espressioni utilizzate attualmente.

Esempio:
<DATA_LIST action="read/write/append" id="<list name>">
viene programmato nel seguente modo:

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

Quando si copiano tag XML ed esempi su più righe, accertarsi che le stringhe non vengano involontariamente spezzate da interruzioni di riga.

Identificatori dei tag	Significato
BREAK	Interruzione condizionata di un loop.
CONDITION	Programmare il tag su una sola riga oppure, in caso di notazione su più righe, separare le condizioni dal nome del tag. **Esempio:** `<if>` `<condition>test <= 2</condition>` `...` oppure `<if>` `<condition>` `test <= 2` `</condition>` `...`
CONTROL	Il tag permette di creare elementi di controllo. Per la descrizione vedere il capitolo "Creazione di menu softkey e finestre di dialogo (Pagina 79)".
CONTROL_RESET	Il tag consente di riavviare uno o più componenti del controllore. **Sintassi:** `<CONTROL_RESET resetnc="TRUE" />` **Attributi:** • **RESETNC = "TRUE"** Il componente NC viene riavviato • **RESETDRIVE = "TRUE"** I componenti dell'azionamento vengono riavviati.

Identificatori dei tag	Significato
CREATE_CYCLE_EVENT	Se il parser inizia l'elaborazione del tag CREATE_CYCLE, viene inviato prima il messaggio <CREATE_CYCLE_EVENT> al Form attivo. Questo messaggio può essere utilizzato per preparare i parametri dei cicli prima che il parser generi l'istruzione NC dalla lista dei parametri e dalla norma di generazione.  **Sintassi:** `<CREATE_CYCLE_EVENT>` `…` `…` `</CREATE_CYCLE_EVENT>` **Esempio:** `<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>`

Identificatori dei tag	Significato
DATA	Il tag consente la scrittura su NC, PLC, GUD e i dati dell'azionamento. Per informazioni sulla formazione dell'indirizzo vedere il capitolo "Indirizzamento dei componenti (Pagina 144)". **Attributo:** • **name** Indirizzo della variabile **Valore del tag:** Come valore del tag è prevista una costante, **una variabile o un termine**. **Sintassi:** `<DATA name="<variable name>"> value </DATA>` `<DATA name="<variable name>"> <term> </DATA>` **Esempio:** `<DATA name = "plc/mb170"> 1 </DATA>` … `<LET name = "tempVar"> 7 </LET>` `<!– il contenuto della variabile locale "tempVar" viene scritto nel byte merker 170 →` `<DATA name = "plc/mb170">$tempVar</DATA>`
DATA Continuazione	**Esempio:** Calcolo di un termine. Il risultato viene assegnato alla variabile indicata. `<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>`

Identificatori dei tag	Significato
DATA_LIST	Il tag consente di effettuare il backup o il ripristino dei dati dell'azionamento e dei dati macchina riportati. Gli indirizzi sono elencati riga per riga. Per informazioni sulla formazione dell'indirizzo vedere il capitolo "Indirizzamento dei componenti (Pagina 144)". Possono essere create fino a 20 liste di dati temporanei. **Attributi:** • **action** *read* – I valori delle variabili elencate vengono salvati in una memoria temporanea *append* – I valori delle variabili elencate vengono aggiunti a una lista esistente *write* – I valori salvati vengono copiati nei corrispondenti dati macchina • **id** Questo identificatore permette di identificare la memoria temporanea **Sintassi:** `<DATA_LIST action="<read/write/append>"  id="<list name>">` `NC/PLC Composizione dell'indirizzo` `</DATA_LIST>` **Esempio:** `<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	Questo è il tag root per la funzione Easy XML. Tutte le istruzioni incorporate vengono elaborate dal parser Easy XML. **Attributi:** È possibile specificare gli attributi elencati per attivare delle funzioni centrali: • **diagnose** - Il valore **true** attiva la diagnostica XML • **debug_msg** - Con il valore "on" vengono salvate le informazioni Trace **Per ulteriori informazioni** vedere il capitolo "Diagnostica XML (Pagina 44)". • **textfile** - Nome del file di testo da utilizzare • **textcontext** - Contesto testuale da impiegare valido solo in relazione all'attributo textfile • **textfilelist** - Consente di caricare una lista con diversi file di testo **Per ulteriori informazioni** vedere il capitolo "File di testo specifici del progetto (Pagina 260)".
DIALOGGUI Continuazione	• **restoresession** - valore **true** Se si seleziona ripetutamente il progetto Easy XML, il parser apre l'ultimo menu selezionato e l'ultimo form selezionato. Questo comportamento viene mantenuto fino al riavvio dell'HMI. **Per ulteriori informazioni** vedere il capitolo "Ripristino della sessione (Pagina 266)".
DEBUG_MSG	L'attributo controlla la memorizzazione dei dati Trace emessi. Per la descrizione vedere il capitolo "Creazione di menu softkey e finestre di dialogo (Pagina 79)", al tag DEBUG. Se è assegnato il valore **on**, il parser salva tutti i dati di output in un file. Questa operazione può rallentare l'elaborazione dello script. Per impostazione predefinita è impostato il valore **off**.

Identificatori dei tag	Significato
DO_WHILE	Loop Do-While DO Istruzioni WHILE (Test) **Sintassi:** <DO_WHILE> Istruzioni … <CONDITION>…</CONDITION> </DO_WHILE> Il loop Do-While è costituito da un blocco di istruzioni e una condizione. Prima viene eseguito il codice racchiuso nel blocco di istruzioni, quindi viene valutata la condizione. Se la condizione è vera (true), la funzione esegue nuovamente la porzione di codice. Questo si ripete finché la condizione non diventa errata (false). **Esempio:** <DO_WHILE> <DATA name = "PLC/qb11"> 15 </DATA> <CONDITIION> "plc/ib9" == 0 </CONDITION> </DO_WHILE>
DYNAMIC_INCLUDE	Il tag include un file di script XML. Contrariamente al tag INCLUDE, la lettura viene eseguita solo con l'elaborazione dell'istruzione corrispondente. In caso di progetti voluminosi, l'uso del tag provoca una riduzione del tempo di caricamento del settore Customer o del supporto cicli. Inoltre diminuisce il fabbisogno medio di risorse, dato che non possono essere sempre richiamate tutte le finestre di dialogo durante una sessione. **Sintassi:** <DYNAMIC_INCLUDE src="path name"/> **Esempio:** <SOFTKEY POSITION="3"> <CAPTION>MY_MENU</CAPTION> <DYNAMIC_INCLUDE src="f:\appl\sk3_script.xml"/> <NAVIGATION>MY_MENU</NAVIGATION> </SOFTKEY>
ELSE	Istruzione se la condizione non è stata soddisfatta (**IF, THEN, ELSE**)

Identificatori dei tag	Significato
FOR	Loop For `for (inizializzazione; test; continuazione) istruzione/i` **Sintassi:** `<FOR>` `<INIT>…</INIT>` `<CONDITION>…</CONDITION>` `<INCREMENT>…</INCREMENT>` `Istruzioni` `…` `</FOR>` Il loop For viene eseguito come descritto qui di seguito: 1. Valutazione dell'espressione **Inizializzazione** (INIT). 2. Valutazione dell'espressione **Test** (CONDITION) come espressione booleana. Se il valore è errato (false), il loop For viene terminato. 3. Esecuzione delle istruzioni seguenti. 4. Valutazione dell'espressione **Continuazione** (INCREMENT). 5. Passare al punto 2. Tutte le variabili che vengono utilizzate nella diramazione INIT, CONDITION e INCREMENT devono essere create all'esterno del loop For. **Esempio:** `<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	Il tag contiene la descrizione di una finestra di dialogo utente. Per la descrizione vedere il capitolo "Creazione di menu softkey e finestre di dialogo (Pagina 79)".
HMI_RESET	Il tag inizializza un riavvio dell'HMI. L'interpretazione viene interrotta dopo questa istruzione.

Identificatori dei tag	Significato
IF	Istruzione condizionata (**IF, THEN, ELSE**) I tag THEN e ELSE sono racchiusi nel tag IF. Al tag IF segue la condizione che viene eseguita nel tag CONDITION. Il risultato dell'operazione determina l'ulteriore elaborazione delle istruzioni. Se il risultato della funzione è vero, la diramazione THEN viene eseguita e la diramazione ELSE viene saltata. Se il risultato della funzione non è vero, il parser elabora la diramazione ELSE. **Sintassi:** `<IF>` `<CONDITION> Condizione != 7 </CONDITION>` `<THEN>` `Istruzione per il caso: Condizione soddisfatta` `</THEN>` `<ELSE>` `Istruzione per il caso: Condizione non soddisfatta` `</ELSE>` `</IF>` **Esempio:** `<IF>` `<CONDITION> "plc/mb170" != 7 </CONDITION>` `<THEN>` `<OP> "plc/mb170" = 7 </OP>` … `</THEN>` `<ELSE>` `<OP> "plc/mb170" = 2 </OP>` … `</ELSE>` `</IF>`
IF Continuazione	Se nella condizione sono utilizzate solo variabili locali, è possibile specificare la condizione come attributi nel tag IF. **Esempio:** `<let name="var1">1</let>` `<if condition="var == 1">` `<then>` `...` `...` `</then>` `</if>` **Nota:** ai controlli che non sono accoppiati con una variabile di riferimento deve essere assegnato il tipo di dati Integer. Eccezioni: Ai Control del tipo **imagebox** e **multiline** viene assegnato il tipo di dati **string**.

Identificatori dei tag	Significato
INCLUDE	L'istruzione include una descrizione XML. (vedere anche DYNAMIC_INCLUDE in questa tabella) **Attributo:** • **src** Contiene il nome del percorso. **Sintassi:** `<?INCLUDE src="<Path name>" ?>`

Identificatori dei tag	Significato
LET	L'istruzione crea una variabile locale con il nome specificato. **Campi:** Con l'attributo **dim** (dimensione) si possono creare campi unidimensionali o bidimensionali. L'indirizzamento dei singoli elementi del campo avviene tramite l'indice del campo. In caso di campo bidimensionale viene specificato prima l'indice delle righe e poi l'indice delle colonne. • Campo monodimensionale: Indice da 0 a 4 0 1 2 3 4 • Campo bidimensionale: Indice riga da 0 a 3 e indice colonna da 0 a 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 **Attributi:** • **name** Nome della variabile • **type** Il tipo di variabile può essere Integer (INT), Unsigned Integer (UINT), Double (DOUBLE), Float (FLOAT), String (STRING) o STRUCT. Inoltre è possibile utilizzare come tipo di variabile una struttura creata mediante **typedef** (vedere l'identificatore del tag TYPEDEF). Se non è specificata una istruzione di tipo, il sistema crea una variabile Integer. `<LET name = "VAR1" type = "INT" />` • **permanent** Se l'attributo è impostato a **true**, il valore della variabile viene salvato in modo permanente. Questo attributo vale solo per le variabili globali. • **poweroff** Se il valore dell'attributo è **true**, i valori si mantengono fino alla disinserzione del controllore. **Per ulteriori informazioni** vedere il capitolo "Ripristino della sessione (Pagina 266)". • **dim** Va specificato il numero di elementi del campo. In caso di campo bidimensionale la seconda dimensione viene specificata dopo la prima dimensione, separata da una virgola. L'accesso a un elemento del campo avviene tramite l'indice del campo, che va definito tra parentesi angolari dopo il nome della variabile. name[index] oppure name[row,column] – Campo monodimensionale: dim="<Numero di elementi>" – Campo bidimensionale: dim="<Numero di righe>,<Numero di colonne>" Gli elementi di campo non inizializzati sono preimpostati a "0".

Identificatori dei tag	Significato
LET Continuazione	**Esempio:** Campo monodimensionale: `<let name="array" dim="10"></let>` Campo bidimensionale: `<let name="list_string" dim="10,3" type="string"></let>` **Preassegnazione:** Una variabile può essere inizializzata con un valore. `<LET name = "VAR1" type = "INT"> 10 </LET>` Se valori dell'NC o variabili PLC vengono memorizzati in una variabile locale, l'operazione di assegnazione adatta automaticamente il formato a quello della variabile letta. • Preassegnazione di una variabile String: A una variabile String possono essere assegnati testi di più righe trasmettendo il testo formattato come valore. Se una riga deve essere conclusa con un line feed <LF> (avanzamento riga) , i caratteri "\\n" devono essere aggiunti alla fine della riga. `<LET name = "text" type = "string"> F4000 G94\\n` `G1 X20\\n` `Z50\\n` `M2\\n` `</LET>>` **`Campi (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>` **Assegnazione:** I valori derivati dai dati macchina o dalle subroutine possono essere assegnati a una variabile con l'operazione di assegnazione "=". La validità di una variabile si estende fino alla fine del blocco XML sovraordinato. Le variabili che devono essere disponibili a livello globale vanno create direttamente dopo il tag **DialogGUi**. Per una finestra di dialogo occorre osservare quanto segue: • L'elaborazione dei messaggi apre il tag corrispondente. • Dopo l'esecuzione del messaggio il tag viene chiuso. • Tutte le variabili all'interno di un tag vengono cancellate con la chiusura.

Identificatori dei tag	Significato
LET Continuazione	**Tipo di variabile struct**: Questo tipo di variabile contiene un insieme di variabili che possono essere richiamate utilizzando il nome della struttura. Una struttura può contenere tutti i tipi di variabile e tutte le strutture. All'interno di una struttura una variabile viene dichiarata con il tag "element". Gli attributi del tag e l'inizializzazione corrispondono agli attributi e all'inizializzazione dell'istruzione let. `<let name="name" type="struct">` `<element name="<nome>" type="<tipo di variabile>" />` … … … `</let>` Ad un elemento di una struttura è possibile assegnare un valore predefinito, che viene utilizzato come valore iniziale per la creazione di una variabile. **Ulteriori informazioni** nella sezione "Inizializzazione di strutture" Questo valore viene sovrascritto se alla creazione della variabile viene specificato un valore iniziale.

Identificatori dei tag	Significato
LET Continuazione	L'accesso a una variabile della struttura avviene tramite il nome della struttura e della variabile. Entrambi i nomi sono separati dall'operatore punto. **Sintassi:** `<op>` `Nome_struttura.nome_variabile = valore;` `</op>` **Esempio:** `<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>`

Identificatori dei tag	Significato
LET Continuazione	**Inizializzazione di strutture:** Le strutture possono essere inizializzate durante la generazione delle variabili specificando un valore iniziale per ciascun elemento strutturale. Nel caso di un campo di strutture, ciascuna struttura va separata dalle altre mediante parentesi graffe. **Esempio:** `<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" >` Con la creazione delle variabili **rect**, gli elementi della struttura vengono inizializzati con i seguenti valori: `rect.left= 22` `rect.top = 122` `rect.right = 220` `rect.bottom = 56` Se si assegnano a una struttura dei valori iniziali, questi sovrascrivono i valori predefiniti. `<let name="rect" type="StructRect" >` `  {11,12}` `</let>` Con la creazione delle variabili rect, gli elementi della struttura vengono inizializzati con i seguenti valori: `rect.left= 11` `rect.top = 12` `rect.right = 220` `rect.bottom = 56`

Identificatori dei tag	Significato
LET Continuazione	**Strutture annidate:** Le strutture sono variabili di un tipo definito dall'utente e possono pertanto essere utilizzate in una struttura come elementi della struttura. L'inizializzazione degli elementi avviene secondo le regole prescritte. **Esempio:** `<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" >` Con la creazione delle variabili **rect_rect_array**, gli elementi della struttura vengono inizializzati con i seguenti valori: `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>` Con la creazione delle variabili **rect_rect_array**, gli elementi della struttura vengono inizializzati con i seguenti valori: `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 Continuazione	**Inizializzazione di variabili String globali:** Se per l'inizializzazione di una variabile String globale si utilizza un identificativo di testo, l'identificativo e il testo da sostituire devono trovarsi nel file di testo standard **oem_xml_screens_xxx.ts** oppure nel file di testo indicato nel tag **DialogGUI**. Al caricamento dell'applicazione, il testo della lingua attiva sostituisce l'identificativo di testo. Se viene eseguita una commutazione della lingua mentre l'applicazione è attiva, non avviene una nuova inizializzazione delle variabili String globali. Il testo può essere sostituito nel messaggio **LANGUAGE_CHANGED**.
LOCK_OPERATING_AREA	La commutazione del settore operativo viene bloccata. Con il tag **UNLOCK_OPERATING_AREA** si rimuove il blocco della commutazione del settore operativo. **Sintassi:** `<LOCK_OPERATING_AREA />`

Identificatori dei tag	Significato
MSG	Il componente operativo definisce il messaggio specificato nel tag. Se viene usato un numero di allarme, la finestra di dialogo definisce il testo memorizzato per il numero. **Esempio:** `<MSG text ="my message" />`
MSGBOX	L'istruzione apre una finestra di messaggio, il cui valore di ritorno può essere usato per la diramazione. **Sintassi:** `<MSGBOX text="<Message>" caption="<caption>" retvalue="<variable name>" type="<button type>" />` **Attributi:** • **text** Testo • **caption** Intestazione • **retvalue** Nome della variabile in cui viene copiato il valore di ritorno: 1 – OK 0 – CANCEL • **type** Possibilità di conferma: "BTN_OK" "BTN_CANCEL" "BTN_OKCANCEL" Se per l'attributo **"text"** o **"caption"** viene usato un numero di allarme, la finestra di messaggio definisce il testo memorizzato per il numero. **Esempio:** `<MSGBOX text="Messaggio di prova" caption="Informazioni" retvalue="result" type="BTN_OK" />`

Identificatori dei tag	Significato
OP	Il tag esegue le operazioni specificate. Possono essere eseguite le operazioni riportate nel capitolo "Operatori (Pagina 77)". Per l'accesso ai dati NC/PLC e dell'azionamento il nome della variabile completo deve essere impostato tra **virgolette**. Per informazioni sulla formazione dell'indirizzo, vedere il capitolo "Indirizzamento dei componenti" (Pagina 144). `PLC: "PLC/MB170"` `NC: "NC/Channel/…"` **Esempio:** `<LET name = "tmpVar" type="INT"> </LET>` `<OP> tmpVar = "plc/mb170" </OP>` `<OP> tmpVar = tmpVar *2 </OP>` `<OP> "plc/mb170" = tmpVar </OP>` All'interno di un tag di operazione possono essere utilizzate più equazioni. Un punto e virgola marca la fine dell'istruzione. **Esempio:** `<op>` `  x = x+1;` `  y = y+1;` `</op>` **Elaborazione delle stringhe:** L'istruzione dell'operazione è in grado di elaborare stringhe e di assegnare la stringa risultante alla variabile di stringa specificata nell'equazione. Per identificare un'espressione testuale occorre anteporre l'identificativo _T al testo. Dopodiché è possibile formattare i valori delle variabili. L'istruzione di formattazione deve essere preceduta dall'identificativo _F. Al termine va specificato l'indirizzo della variabile. **Esempio:** `<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>`

Identificatori dei tag	Significato
OPERATION	**Operazione** Un'operazione di spostamento può essere utilizzata all'interno di un'equazione. **Spostamento dell'operatore a sinistra "<<"** I bit vengono spostati a sinistra con la funzione <<. Il valore da spostare e il numero di incrementi di spostamento possono essere definiti direttamente o tramite una variabile. Quando viene raggiunto il limite del formato di dati, i bit vengono spinti oltre il limite senza alcuna segnalazione di errore. **Regola di esecuzione** Esecuzione da sinistra a destra e '+' e '-' prima di '<<' o '>>' **Esempio:** `<op> idx = idx << 2 </op>` `<op> idx = 3 + idx << 2 </op>` **Spostamento dell'operatore a destra ">>"** I bit vengono spostati a destra con la funzione >>. Il valore da spostare e il numero di incrementi di spostamento possono essere definiti direttamente o tramite una variabile. Quando viene raggiunto il limite del formato di dati, i bit vengono spinti oltre il limite senza alcuna segnalazione di errore. **Regola di esecuzione** Esecuzione da sinistra a destra e '+' e '-' prima di '<<' o '>>' **Esempio:** `<op> idx = idx >> 2 </op>` `<op> idx = 3+idx >> 2</op>`
PASSWORD	Per la funzione "EasyExtend" questo tag serve per attivare i gruppi aggiuntivi. La stringa di caratteri immessa viene scritta nella variabile di riferimento indicata e deve essere elaborata nel programma utente del PLC. **Sintassi:** `<PASSWORD refVar ="<variable name>" />` **Attributi:** • **refVar** Nome della variabile di riferimento • **text** Un'indicazione di attributo sostituisce il testo standard (opzionale) **Esempio:** `<PASSWORD refvar="plc/mw107" />` **Esempio opzionale:** `<password refVar = "plc/MD108" text="Password" />`
POWER_OFF	Un messaggio invita l'utente a spegnere la macchina. Il testo del messaggio è memorizzato nel sistema in modo permanente.

Identificatori dei tag	Significato
PRINT	Il tag esporta un testo nella riga della finestra di dialogo oppure copia il testo nella variabile specificata. Se il testo contiene codici di formattazione, i valori delle variabili vengono inseriti nelle posizioni corrispondenti. **Sintassi:** `<PRINT name="Nome variabile" text="text %Formattazione"> Variable, … </PRINT>` `<PRINT text="text %Formattazione"> Variable, … </PRINT>` **Attributi:** • **name** Nome della variabile in cui il testo deve essere memorizzato (opzionale) • **text** Testo **Formattazione:** Il carattere "%" formatta la variabile specificata come valore. %[flags] [larghezza] [.cifre decimali] tipo • Flag: Carattere opzionale per la definizione della formattazione di output: – allineato a destra o a sinistra ("-" per allineato a sinistra) – aggiunta di zeri non significativi ("0") – riempimento con spazi • Larghezza: L'argomento stabilisce la larghezza minima di emissione di un numero non negativo. Se il valore da emettere ha meno cifre di quanto stabilito dall'argomento, quelle mancanti vengono riempite di spazi. • Cifre decimali: Per un numero a virgola mobile il parametro opzionale stabilisce la quantità di cifre decimali. • Tipo: Il carattere di tipo stabilisce quali formati di dati dell'istruzione Print vengono trasferiti. Questi caratteri devono essere specificati. – d: valore intero – f: numero a virgola mobile – s: stringa

Identificatori dei tag	Significato
PRINT Continuazione	**Valori:** Numero di variabili i cui valori devono essere inseriti nel testo. I tipi di variabili devono coincidere con l'identificatore di tipo corrispondente dell'istruzione di formattazione e vanno separati con virgole. **Esempio:** Emissione di un testo nella riga di informazione `<PRINT text="Testo informativo" />` Emissione di un testo con formattazione della variabile `<LET name="trun_dir"></LET>` `<PRINT text="M%d">trun_dir</PRINT>` Emissione di un testo in una variabile String con formattazione della variabile `<LET name="trun_dir"></LET>` `<LET name="str" type="string" ></LET>` `<print name="str" text="M%d ">trun_dir</print>`
PROGRESS_BAR	Il tag apre o chiude una barra di avanzamento. La barra viene visualizzata sotto la finestra dell'applicazione. **Sintassi:** `<PROGRESS_BAR type="<true/false>"> value </ PROGRESS_BAR>` **Attributi:** • **type =** "TRUE" - Apre la barra di avanzamento • **type** = "FALSE" - Chiude la barra di avanzamento • **min** (opzionale) – Valore minimo • **max** (opzionale) – Valore massimo **Valore:** • **Value** Posizione percentuale della barra **Esempio:** `<PROGRESS_BAR type="true" min="0" max="101" >20< /` `PROGRESS_BAR>......<PROGRESS_BAR >50< /` `PROGRESS_BAR>......<PROGRESS_BAR type="false" >100< /PROGRESS_BAR>`

Identificatori dei tag	Significato
SEND_MESSAGE	Il tag invia un messaggio con due parametri al Form attivo che viene elaborato nel tag Message. **Sintassi:** `<SEND_MESSAGE>p1, p2</SEND_MESSAGE>` **Esempio:** `<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	Questo tag consente di gestire la visibilità di un controllo. **Sintassi:** `<SHOW_CONTROL name="<name>" type="<type>" />` **Attributi:** • **name** Nome del controllo • **type =** "TRUE" - Il controllo diventa visibile • **type** = "FALSE" - Il controllo diventa invisibile (nascosto) **Esempio:** `<SHOW_CONTROL name="myEditfield" type="false" />` … … `<SHOW_CONTROL name="myEditfield" type="true" />`

Identificatori dei tag	Significato
SLEEP	Il tag interrompe l'elaborazione dello script per l'intervallo di tempo specificato. Il tempo di interruzione si ricava dal valore trasmesso moltiplicato per la base di tempo di 50ms. **Sintassi:** `<SLEEP value="Tempo di interruzione" />` **Esempio:** Tempo di attesa 1,5 sec. `<SLEEP value="30" />`
STOP	L'interpretazione viene interrotta in questa posizione.
SWITCH	L'istruzione **SWITCH** descrive una scelta multipla. Un'espressione viene valutata una volta e confrontata con un numero di costanti. Se l'espressione coincide con la costante, le istruzioni vengono elaborate all'interno dell'istruzione **CASE**. L'istruzione **DEFAULT** viene elaborata se nessuna costante coincide con l'espressione. **Sintassi:** `<SWITCH>` `<CONDITION> Valore </CONDITION>` `<CASE value="<costante 1>">` `Istruzioni` `…` `</CASE>` `<CASE value="<costante 2>">` `Istruzioni` `…` `</CASE>` `<DEFAULT>` `Istruzioni` `…` `</DEFAULT>` `</SWITCH>`

Identificatori dei tag	Significato
SWITCHTOAREA	Il tag SWITCHTOAREA passa dal settore Customer al settore operativo specificato. Il parametro viene specificato come valore di attributo. **Sintassi:** `<switchToArea name="area" args="argument "/>` **Attributi:** **name** I seguenti nomi sono ammessi per i settori operativi: • **AreaMachine** - Macchina • **AreaParameter** - Parametri • **AreaProgramEdit** - Editor • **AreaProgramManager** - Program Manager • **AreaDiagnosis** - Diagnostica • **AreaStartup** - Messa in servizio **args** (riservato) Per l'attivazione delle finestre di dialogo, al settore operativo vengono conferiti anche i relativi argomenti di runtime. **Esempio:** `<switchToArea name="AreaMachine" />`
SWITCHTODYNAMICTARGET	Se la finestra di dialogo precedente si definisce come destinazione dinamica di salto, il tag SWITCHTODYNAMICTARGET attiva questa finestra di dialogo e termina l'elaborazione dello script. **Sintassi:** `<switchToDynamicTarget />`
THEN	Istruzione nel caso in cui la condizione sia stata soddisfatta (**IF, THEN, ELSE**)
TYPE_CAST	Questo tag consente di modificare il tipo di dati di una variabile locale. **Sintassi:** <type_cast name="variable name" type=" new type" /> **Attributi:** • **name** Nome della variabile • **type** Alla variabile viene assegnato il nuovo tipo di dati. • **convert** Alla variabile viene assegnato il nuovo tipo di dati. Inoltre il valore della variabile viene convertito nel nuovo tipo di dati.

Identificatori dei tag	Significato
TYPEDEF	Il tag permette di definire un nuovo identificatore per un tipo di dati. Per le definizioni di strutture, questo ha il vantaggio che prima si definisce il tipo di dati e poi lo si può usare in una istruzione LET. Come attributi sono previsti l'identificatore e il tipo. Il parser supporta solo l'impostazione di definizioni di strutture. All'interno della definizione di tipo, una variabile viene dichiarata con il tag "element". Gli attributi del tag corrispondono agli attributi dell'istruzione LET. `<typedef name="<identificatore>" type="struct">` `  <element name="<nome>" type="<tipo variabili>" />` … … … `</typedef>` Dopo la definizione l'identificatore può essere utilizzato come tipo di dati per l'istruzione LET. `<let name="<nome variabile>" type="<identificatore>"></let>` **Esempio**: `<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>`

Identificatori dei tag	Significato
TYPEDEF Continuazione	Alcune funzioni predefinite richiedono come parametri di richiamo le variabili del tipo di struttura RECT, POINT oppure SIZE. Queste strutture sono definite nel file struct_def.xml. **Per ulteriori informazioni** vedere il capitolo "Creazione del file struct_def.xml (Pagina 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	Rimozione del blocco della commutazione del settore operativo **Sintassi:** `<UNLOCK_OPERATING_AREA />`
WAITING	Il tag attende il riavvio del componente dopo un reset dell'NC o dell'azionamento. **Attributi:** • **WAITINGFORNC** = "TRUE" - Il sistema attende il riavvio dell'NC • **WAITINGFORDRIVE** = "TRUE" - Il sistema attende il riavvio degli azionamenti **Sintassi:** `<WAITING WAITINGFORNC = "TRUE"/>` **Esempio:** ... `<CONTROL_RESET resetnc = "true" resetdrive = "true"/>` `<WAITING waitingfornc = "true" waitingfordrive = "true" />` ...

Identificatori dei tag	Significato
WHILE	Loop While `WHILE (Test)` `  Istruzione` **Sintassi:** `<WHILE>` `<CONDITION>…</CONDITION>` `Istruzioni` … `</WHILE>` Il loop While permette di eseguire più volte una sequenza di istruzioni per tutto il tempo in cui una condizione è soddisfatta. Questa condizione viene verificata prima dell'elaborazione della sequenza di istruzioni. **Esempio:** `<WHILE>` `<CONDITION> "plc/ib9" == 0 </CONDITION>` `<DATA name = "PLC/qb11"> 15 </DATA>` `</WHILE>`
XML_PARSER	Il tag "XML_PARSER" può essere usato per il parsing di file XML. Il parser interpreta un file XML e richiama funzioni di richiamo definite. Ogni funzione di richiamo appartiene a un evento predefinito. All'interno di questa funzione il programmatore può elaborare i dati XML. **Eventi predefiniti:** • start document Il parser apre il documento e inizia il parsing. • end document Il parser chiude il documento. • start element Il parser ha trovato un elemento e crea una lista con tutti gli attributi e i loro valori. Queste liste vengono inoltrate alla funzione di richiamo. • end element La fine dell'elemento è stata trovata. • characters Il parser inoltra tutti i caratteri di un elemento. • error Il parser ha rilevato un errore di sintassi. Se si verifica un evento, il parser attiva la funzione di richiamo e verifica il valore di ritorno della funzione. Se la funzione restituisce il valore "true", il parser prosegue il processo.

Identificatori dei tag	Significato
XML_PARSER Continuazione	**Interfacce** Il valore dell'attributo **name** contiene il percorso al file XML. Per assegnare eventi alle funzioni di richiamo, devono essere definite le seguenti proprietà: **Standard** startElementHandler endElementHandler charactersHandler **Opzionale** errorHandler documentHandler Il valore di un attributo definisce il nome della funzione di richiamo. **Esempio:** `<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>`

Identificatori dei tag	Significato
XML_PARSER Continuazione	Il parser fornisce inoltre variabili in modo che le funzioni di richiamo possano accedere ai dati degli eventi. **startElementHandler:** **Parametri di funzione** tag_name - Nome del tag num - Numero di attributi trovati **Variabili di sistema** $xmlAttribute Array di stringhe con il numero di elementi indicati con num. $xmlValue Array di stringa che contiene il campo di valori attributo 0-num. **Esempio:** `<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:** **Parametri di funzione** tag_name - Nome del tag **Esempio:** `<function_body name="endElementHandler" parameter="tag_name">` `<print text="name %s"> tag_name </print>` `</function_body>`

Identificatori dei tag	Significato
XML_PARSER Continuazione	**charactersHandler:** **Variabili di sistema** $xmlCharacters Stringa con i dati $xmlCharactersStart Sempre 0 $xmlCharactersLength Numero dei byte **Esempio:** `<function_body name="charactersHandler" return="true" >   <print text="chars %s"> $xmlCharacters </print> </function_body>` **documentHandler:** **Parametri di funzione** state 1 inizio documento, 2 fine documento **errorHandler:** **Variabili di sistema** $xmlErrorString Contiene la riga non valida (variabile di sistema) **Esempio:** `<function_body name="errorHandler" >   <print text="error %s">$xmlErrorString</print> </function_body>` **Risultato del richiamo:** $return Se 1 (true), il parser prosegue il parsing del file

3.7.3 Codifiche di colore

L'attributo color utilizza lo schema di codifiche di colore del linguaggio HTML.

Un'indicazione di colore si compone sintatticamente del carattere "#" (diesis) e di sei cifre del sistema esadecimale; ogni colore è rappresentato da due cifre.

R – rosso

G – verde

B – blu

#RRGGBB

Esempio:

color= "#ff0011"

Colori di esempio:

Rosso	Verde	Blu	Giallo	Bianco	Nero
#FF0000	#00FF00	#0000FF	#ffff00	#FFFFFF	#000000

3.7.4 Sintassi XML speciale

I caratteri che hanno un significato speciale nella sintassi XML devono essere trascritti se vengono poi rappresentati da un editor XML generico.

Questo riguarda i caratteri seguenti:

Carattere	Notazione in XML
<	<
>	>
&	&
"	"
'	'

3.7.5 Caratteri speciali HTML

Per la rappresentazione dei caratteri o per l'uso dei caratteri di controllo è disponibile la valutazione dei caratteri speciali HTML.

Possono essere utilizzate le codifiche decimali e esadecimali di Codepage Latin 1.

Esempio

Carattere	Descrizione	Notazione decimale	Notazione esadecimale
"	Virgolette alte	"	"
LF	Line Feed

3.7.6 Operatori

L'istruzione di operazione elabora i seguenti operatori:

Operatore	Significato
=	Assegnazione
==	uguale a
<, <	minore di
>, >	maggiore di
<=, <=	minore o uguale a
>=, >=	maggiore o uguale a
|	combinazione OR a bit
||	combinazione OR logica
&, &	combinazione AND a bit
&&, &&	combinazione AND logica
+	addizione
-	sottrazione
*	moltiplicazione
/	divisione
!	non
!=	diverso

Le istruzioni di operazione vengono elaborate da sinistra verso destra. In alcuni casi può essere opportuno racchiudere le espressioni tra parentesi per definire la priorità di elaborazione delle espressioni parziali.

3.7.7 Variabili di sistema

Le variabili di sistema sono variabili che sono disponibili in ogni script e che consentono lo scambio dati tra il parser e l'esecuzione dello script.

La tabella seguente fornisce una panoramica dei tag per i quali vengono create automaticamente delle variabili:

Nome della variabile	Significato	Validità nel tag
$actionresult	Segnala al parser se l'evento deve essere elaborato dal parser stesso o meno	KEY_EVENT
$focus_name	Contiene il nome del campo che è attivo	FOCUS_IN INDEX_CHANGED EDIT_CHANGED
$focus_item_data	Contiene il valore numerico item_data che è stato assegnato al campo	
$gestureinfo	In presenza di azioni delle dita, il parser fornisce le informazioni relative alle azioni nella variabile ed esegue il tag gesture_event.	GESTURE_EVENT
$return	La variabile consente di trasferire il valore di ritorno di una sotto-funzione	FUNCTION_BODY
$message_par1 $message_par2	Contiene i parametri di richiamo della funzione SEND_MESSAGE	MESSAGE
$xmlAttribute	Contiene un elenco degli attributi dei tag rilevati	XML_PARSER startElementHandler
$xmlValue	Contiene un elenco dei valori dei tag rilevati	
$xmlCharacters	Contiene il flusso dati	XML_PARSER charactersHandler
$xmlCharactersStart	Contiene l'indice iniziale	
$xmlCharactersLength	Contiene il numero di caratteri memorizzati nel flusso dati	
$mouse_event.type $mouse_event.x $mouse_event.y $mouse_event.id $mouse_event.buttons $mouse_event.button	Struttura per il trasferimento dei parametri degli eventi mouse	MOUSE_EVENT

3.7.8 Creazione di menu softkey e finestre di dialogo

Per consentire l'inserimento di finestre di dialogo, nella descrizione XML deve essere presente un tag di menu con il nome "main". Questo tag viene richiamato dal sistema dopo l'attivazione del settore operativo <CUSTOM>. All'interno del tag è possibile definire altre diramazioni di finestre di dialogo e l'attivazione di una finestra di dialogo.

```
<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>
```

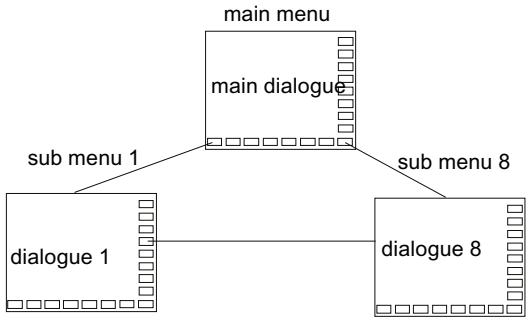
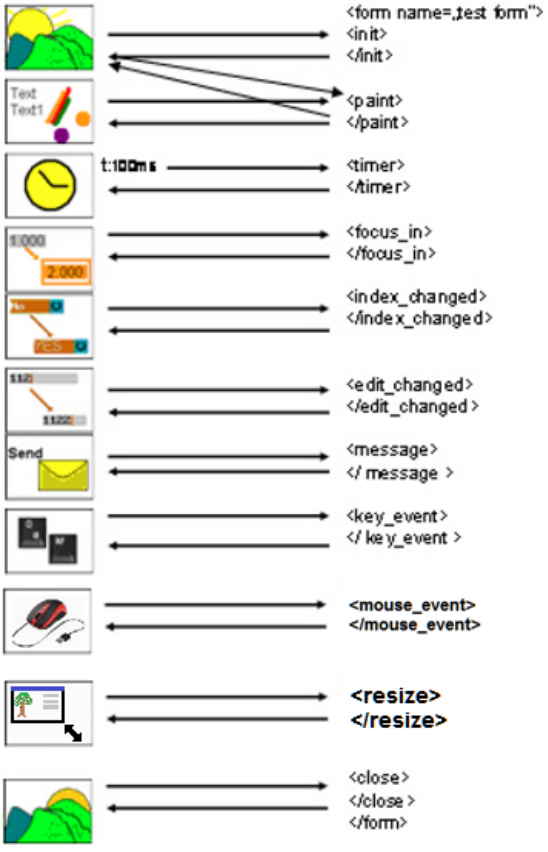


Figura 3-6 Struttura dei menu

Identificatori dei tag	Significato
FORM	Il tag contiene la descrizione di una finestra di dialogo utente. I tag corrispondenti sono descritti nel capitolo Creazione di menu e finestre di dialogo. **Sintassi:** `<FORM name="<dialog name>" color="#ff0000">` **Attributi:** • **color** Colore di sfondo della finestra di dialogo **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)". – Bianco per impostazione predefinita • **name** Identificatore del form • **type** Il valore consentito è *cycle* che contrassegna una finestra di ciclo utente • **xpos** Posizione X dell'angolo superiore sinistro della finestra di dialogo (opzionale) • **ypos** Posizione Y dell'angolo superiore sinistro (opzionale) • **width** Estensione in direzione X (in pixel) (opzionale) • **height** Estensione in direzione Y (in pixel) (opzionale)
FORM Continuazione	Con l'attributo AUTOSCALE_CONTENT è possibile definire il comportamento del controllo di un Form con diverse risoluzioni dello schermo. Come standard i controlli vengono adattati automaticamente alla risoluzione dello schermo esistente. Se si desidera gestire anche il posizionamento e il dimensionamento, è necessario specificare il tag Form con il valore OFF. **Attributo:** **autoscale_content** **Valori:** • **on** Le coordinate dei controlli vengono adattate automaticamente alla risoluzione dello schermo (valore predefinito) • **off** Le coordinate dei controlli vengono utilizzate senza modifiche **Esempio:** `<form name="main_form" autoscale_content="off">` … … … `</form>`

Identificatori dei tag	Significato
FORM Continuazione	Attributi: • **textfile** Questo attributo permette di specificare un file di testo riferito al form. Come contesto testuale standard il sistema usa l'identificatore EASY_XML. • **textcontext** Questo attributo permette di specificare un identificatore per il contesto testuale da impiegare. Questo attributo è valido solo insieme all'attributo **textfile**.
FORM Continuazione	Per utilizzare i layout standard o i layout salvati nel file easyscreen.ini, si deve utilizzare l'attributo LAYOUT. Come valore va indicato il nome del layout da utilizzare. **Nota:** Una modifica delle impostazioni standard è utile solo per le finestre di dialogo popup, dato che in questo caso la finestra di dialogo da richiamare non viene nascosta. **Attributo:** **layout** **Sintassi:** `<form name="Nome del Form" layout="Nome del layout" >` … … `</form>` **Esempio:** `<form name="form0" layout="slstandardscreenlayout.` `SlStandardScreenLayout.LowerForm" >` … … `</form>` Oltre alle posizioni delle maschere e alle dimensioni predefinite, nel file easyscreen.ini è possibile memorizzare le proprie definizioni. Registrazione in easyscreen.ini: `[640x480]` `MyPanel = x:=0, y:=220, width:=340, height:=174` `[800x480]` `MyPanel = x:=0, y:=220, width:=420, height:=174` **Esempio:** `<form name="form0" layout="MyPanel" >` … … `</form>`

Identificatori dei tag	Significato
FORM Continuazione	**Messaggi della finestra di dialogo:** • **INIT** • **PAINT** • **TIMER** • **CLOSE** • **FOCUS_IN** • **INDEX_CHANGED** • **EDIT_CHANGED** • **GESTURE_EVENT** • **KEY_EVENT** • **MESSAGE** • **MOUSE_EVENT** • **RESIZE** • **CHANNEL_CHANGED** • **LANGUAGE_CHANGED**
FORM Continuazione	

Identificatori dei tag	Significato
FORM Continuazione	**Sintassi:** `<FORM name = "<dialog name>" color = "#ff0000">` **Esempio:** `<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	**Messaggio della finestra di dialogo** Il tag viene elaborato immediatamente dopo la creazione della finestra di dialogo. Qui vanno creati tutti gli elementi di immissione e gli "hotlink" delle finestre di dialogo.

Identificatori dei tag	Significato
KEY_EVENT	Messaggio della finestra di dialogo Il tag KEY_EVENT può essere inserito nel Form per valutare eventi da tastiera. Se il tag è presente in un Form, il sistema invia il codice tastiera MF2 al Form attivo. Se la variabile **$actionresult** non è impostata a zero, il sistema elabora quindi l'evento da tastiera. Il codice tastiera viene messo a disposizione nella variabile **$keycode** come valore intero. **Esempio:** Il carattere immesso nella variabile **exclude_key** deve essere filtrato a partire dalla corrente di ingresso. `<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>`

Identificatori dei tag	Significato
MOUSE_EVENT	Il tag può essere inserito nello script per l'elaborazione degli eventi mouse. Viene eseguito se sono state effettuate le seguenti attività con il mouse: • Un tasto è stato premuto • Un tasto è stato rilasciato • Il mouse è stato spostato Il parser mette a disposizione le informazioni in una struttura e crea la variabile di struttura **$mouse_event** con i seguenti elementi: **Elementi della struttura:** • **type** Codifica dell'attività – 2 - Un tasto è stato premuto – 3 - Un tasto è stato rilasciato – 5 - Il mouse è stato spostato • **x** Posizione X del cursore in pixel; riferita alla risoluzione corrente dello schermo • **y** Posizione Y del cursore in pixel; riferita alla risoluzione corrente dello schermo • **id** Identificatore – -1 se la posizione non può essere assegnata ad alcun controllo – != -1: se il cursore del mouse si trova all'interno di un controllo, viene restituito il contenuto dell'attributo **idemdata** • **button** Contiene lo stato dei tasti al momento dell'evento – 0 - Nessun tasto – 1 - Tasto sinistro – 2 - Tasto destro – 4 - Tasto centrale I tasti possono essere messi in connessione logica con un'operazione OR a bit. **Esempio:** `<MOUSE_EVENT>` `  <print text="button %d type %d x %d y %d ">$mouse_event.button, $mouse_event.type, $mouse_event.x, $mouse_event.y</print>` `</MOUSE_EVENT>`

Identificatori dei tag	Significato
MESSAGE	**Messaggio della finestra di dialogo** Se nello script viene eseguita l'istruzione Send_message, il parser elabora il tag Message. Vengono messi a disposizione i valori p1 e p2 nelle variabili $message_par1 e $message_par2 (vedere il tag "SEND_MESSAGE"). **Sintassi:** `<MESSAGE>` `</MESSAGE>` **Esempio:** `<LET name="user_selection" />` `<SOFTKEY POSITION="3">` `  <CAPTION>Set%nParameter</CAPTION>` `  <SEND_MESSAGE>1, 10</SEND_MESSAGE>` `</SOFTKEY>` … … `<FORM>` … … `<MESSAGE>` `<SWITCH>` `<CONDITION>$message_par1</CONDITION>` `<CASE value="1">` `<OP> user_selection = $message_par2 </OP>` … … `</CASE>` `</SWITCH>` `</MESSAGE>` … … `</FORM>`

Identificatori dei tag	Significato
SEND_MESSAGE	Il tag invia un messaggio con due parametri al Form attivo che viene elaborato nel tag Message (vedere anche MESSAGE). **Sintassi:** `<SEND_MESSAGE>p1, p2</SEND_MESSAGE>` **Esempio:** `<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>`

Identificatori dei tag	Significato
RESIZE	**Messaggio della finestra di dialogo** Il tag può essere inserito nello script per l'elaborazione di un evento RESIZE. Questo evento viene generato da una commutazione dinamica della risoluzione. 

Identificatori dei tag	Significato
RESIZE Continuazione	**Esempio:** `<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>`

Identificatori dei tag	Significato
CHANNEL_CHANGED	**Messaggio della finestra di dialogo** Il tag consente l'elaborazione dell'evento Cambio canale. Viene eseguito quando l'utente preme il tasto per il cambio di canale. **Sintassi:** `<channel_changed>` `</channel_changed>` **Esempio:** Per un cambio canale si deve chiudere e poi riaprire il Form per poter visualizzare i dati del canale selezionato. `<channel_changed>` `  <open_form name = "form1" reopen="true"/>` `</channel_changed>` oppure Per un cambio canale viene attivato un altro menu. `<channel_changed>` `  <navigation>menu2</navigation>` `</channel_changed>`
LANGUAGE_CHANGED	**Messaggio della finestra di dialogo** Il tag consente l'elaborazione di una richiesta di commutazione della lingua. Viene eseguito quando l'utente cambia la lingua con una maschera di dialogo aperta. **Sintassi:** `<language_changed>` `</language_changed>` **Esempio:** Per una commutazione della lingua si deve chiudere e poi riaprire il Form per poter assegnare ai controlli gli elementi di testo della lingua attivata. `<language_changed>` `  <open_form name = "form1" reopen="true"/>` `</language_changed>`
FOCUS_IN	**Messaggio della finestra di dialogo** Il tag viene richiamato quando il sistema si focalizza su un controllo. Per identificare il controllo, il sistema copia il nome del controllo nella variabile **$focus_name** e il valore dell'attributo **item_data** nella variabile **$focus_item_data**. Questo messaggio può essere usato ad es. per emettere immagini in base alla posizione di focalizzazione. **Esempio:** `<focus_in>` `<PRINT text="focus on filed:%s, %d">$focus_name,` `$focus_item_data </PRINT>` `</focus_in>`

Identificatori dei tag	Significato
PAINT	**Messaggio della finestra di dialogo** Il tag viene elaborato alla visualizzazione della finestra di dialogo. Qui vanno specificati tutti i testi e le immagini che la finestra di dialogo deve visualizzare. Inoltre il tag viene eseguito quando il sistema rileva che parti della finestra di dialogo devono essere nuovamente visualizzate. Questo può avvenire ad es. a seguito della chiusura di finestre sovrapposte.
TIMER	**Messaggio della finestra di dialogo** Il tag viene elaborato ciclicamente. A ogni Form è assegnato un temporizzatore che avvia l'elaborazione del tag Timer all'incirca ogni 100 ms.
CAPTION	Il tag contiene il titolo della finestra di dialogo. Questo tag deve essere utilizzato all'interno del tag INIT. La riga del titolo può essere suddivisa in più colonne. Per suddividere la riga del titolo, prima di emettere il testo occorre programmare il tag caption con l'attributo define_section. L'attributo define_section definisce il numero di colonne Con l'attributo property si definisce per ogni colonna la lunghezza, la posizione di partenza e l'orientamento del testo. L'indicazione di posizione e lunghezza è un valore percentuale che si riferisce all'ampiezza del Form. Il valore dell'attributo value indica il numero di colonna (che inizia con zero) `<caption define_sections="3">` `<property value="0" length="20" position="2" alignment="left" />` `<property value="1" length="20" position="79" alignment="right" />` `</caption>` Al termine il testo può essere assegnato alla colonna specificando inoltre l'attributo index con l'indice di colonna. `<caption index="0">colonna1</caption>` `<caption index="1">colonna2</caption>` **Sintassi:** `<CAPTION>Titel</CAPTION>` **Esempio:** `<CAPTION>my first dialogue</CAPTION>`
CLOSE	**Messaggio della finestra di dialogo** Questo tag viene elaborato prima della chiusura della finestra di dialogo.

Identificatori dei tag	Significato
CLOSE_FORM	Il tag chiude la finestra di dialogo attiva. Questa istruzione è necessaria solo se la finestra di dialogo è stata aperta con il comando MMC e se all'operatore è stata proposta una funzione softkey per chiudere la finestra di dialogo. In generale le finestre di dialogo vengono gestite automaticamente e non devono essere chiuse esplicitamente. **Sintassi:** `<CLOSE_FORM/>` **Esempio:** `<softkey_ok>` `<caption>OK</caption>` `<CLOSE_FORM />` `<navigation>main_menu</navigation>` `</softkey_ok>`

Identificatori dei tag	Significato
CONTROL	Il tag permette di creare elementi di controllo. **Sintassi:** `<CONTROL name = "<control name>" xpos = "<Posizione X>" ypos = "<Posizione Y>" refvar = "<Variabile NC>" hotlink = "true" format = "<Formato>" />` **Nota**: Le variabili che vengono utilizzate per l'impostazione dei valori degli attributi devono essere dichiarate come variabili globali. **Attributi:** • **name** Identificatore del campo. L'identificatore rappresenta contemporaneamente una variabile locale e non può essere usato più volte nel Form. • **xpos** Posizione X dell'angolo superiore sinistro • **ypos** Posizione Y dell'angolo superiore sinistro • **fieldtype** Tipo di campo Se non è specificato un tipo, il campo viene configurato come campo di Edit. – **edit** I dati possono essere modificati – **readonly** I dati non possono essere modificati **combobox** Invece di valori numerici il campo mostra l'identificatore corrispondente. Se viene selezionato il tipo di campo **combobox**, è inoltre necessario assegnare al campo le espressioni da rappresentare. A questo scopo va utilizzato il tag <ITEM> . La casella combinata memorizza l'indice del testo selezionato attualmente nella variabile facente parte del controllo (vedere l'attributo **refvar**). Dopo la creazione del controllo è possibile inserire altri elementi con le funzioni **addItem** o **insertItem**. – **progressbar** Viene rappresentata una barra di avanzamento con valori compresi tra 0 e 100. L'intervallo di valori può essere adattato al dato da rappresentare con le proprietà Valore minimo e massimo.

Identificatori dei tag	Significato
CONTROL Continuazione	• **fieldtype edit** e **readonly** Nei campi di tipo **edit** e **readonly**, con l'attributo **multiline** è possibile visualizzare testi su più righe. L'interruzione di riga avviene in corrispondenza di un limite di parola oppure con il carattere Line Feed <LF>. **Esempio:** `<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>`

Identificatori dei tag	Significato
CONTROL Continuazione	• **fieldtype** – **graphicbox** Il tipo di campo genera un controllo in grafica tratteggiata bidimensionale. Con il tag <ITEM> è possibile inserire un elemento grafico nel controllo. I parametri **width** e **height** specificano la larghezza e l'altezza del box. Dopo la creazione del controllo è possibile inserire altri elementi con le funzioni **addItem** o **insertItem**. Il parametro **itemdata** non viene valutato per questo controllo. Se al controllo viene assegnata una variabile di riferimento, la registrazione dei suoi valori avviene ogni 100 ms. Al massimo è possibile registrare fino a 10000 valori. Con la proprietà **max** ha luogo una registrazione infinita dei valori; al raggiungimento della durata massima di registrazione viene cancellato il valore meno recente. La durata di registrazione deve essere specificata in millisecondi. I valori salvati con discrezione temporale sono rappresentati dal controllo come linee reciprocamente collegate. L'attributo **channel** consente la registrazione di fino a tre altre variabili di riferimento. L'indice inizia con uno. L'indice zero serve all'impostazione delle proprietà delle variabili assegnate nel tag Control. **Esempio:** `<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>`

Identificatori dei tag	Significato
CONTROL Continuazione	**Esempio di graphicbox:** `<CONTROL name= "graphic" xpos = "8" ypos="23" width="300" height="352" fieldtype="graphicbox" />` • Aggiunta di elementi: Gli elementi vengono aggiunti con la funzione **additem** o **loaditem**. Possono essere utilizzati i seguenti elementi 2d: – Linea - l(inc) – Settore circolare - c(ircle) – Punto - p(oint) **Struttura di un elemento:** <tipo di elemento>; coordinate • Line: l; xs; ys; xe, ye I - identificatore della linea: Xs - posizione iniziale X Ys - posizione iniziale Y Xe - posizione finale X Ye - posizione finale Y • Circle: C, xs, ys, xe, ye, cc_x, cc_y, r C - identificatore del settore circolare Xs - posizione iniziale X Ys - posizione iniziale Y Xe - posizione finale X Ye - posizione finale Y Cc_x - centro del cerchio coordinata X Cc_y - centro del cerchio coordinata Y • Radius: R • Point: P, x, y P - identificatore del punto X - posizione X Y - posizione Y • Eliminazione della grafica: Il contenuto viene eliminato con la funzione **empty**.

Identificatori dei tag	Significato
CONTROL Continuazione	Nel capitolo "Progettazione personalizzata dei pulsanti (Pagina 214)" sono descritti i seguenti tipi di campi: • **fieldtype** – **pushbutton** Il tipo di campo può essere impiegato come tasto o come interruttore. – **radiobutton** Il tipo di campo consente di selezionare un'opzione fra più opzioni disponibili. – **checkbox** Il tipo di campo consente di selezionare più opzioni. – **groupbox** Il tipo di campo riunisce visivamente un gruppo di controlli in un riquadro con un titolo. – **switch** Il tipo di campo è un elemento grafico che segnala uno stato di due stati possibili mediante un'icona. – **scrollarea** Il tipo di campo viene utilizzato per visualizzare i controlli all'interno di un'area definita.

Identificatori dei tag	Significato
CONTROL Continuazione	• **fieldtype** – **listbox** Il tipo di campo genera un controllo vuoto di una casella di riepilogo. Con il tag <ITEM> è possibile inserire un elemento di casella di riepilogo nella casella di riepilogo. L'attributo ITEM **value** permette di assegnare un valore univoco a questo elemento. Può servire, ad esempio, per l'identificazione dell'elemento. I parametri **width** e **height** indicano la larghezza e l'altezza della casella di riepilogo. Dopo la creazione del controllo è possibile inserire altri elementi della casella di riepilogo con le funzioni **addItem**, **insertItem** o **loadItem**. Dopo la creazione del controllo è possibile inserire altri elementi della casella di riepilogo con la funzione **addItem** o **insertItem**. Il parametro **itemdata** non viene valutato per questo controllo. – **itemlist** Il tipo di campo genera un controllo Static che invece dei valori numerici visualizza gli identificatori corrispondenti. Con il tag <ITEM> è possibile assegnare un identificatore al campo. • **item_data** All'attributo può essere assegnato un valore intero specifico dell'utente. Questo valore viene comunicato al messaggio FOCUS_IN per identificare il campo attivo. • **refvar** Identificatore della variabile di riferimento che può essere collegata al campo (opzionale). • **hotlink** = "TRUE" " Se il valore della variabile di riferimento cambia, il campo viene aggiornato automaticamente (opzionale). • **format** L'attributo definisce il formato di visualizzazione della variabile specificata. Per informazioni sulla formattazione vedere print-Tag (opzionale). L'attributo **format** consente anche la rappresentazione di un valore intero in altri formati numerici, ad es. la rappresentazione dei tipi di campo **edit** e **readonly** come valore booleano. La formattazione tramite la quantità di caratteri e l'allineamento non è supportata. È possibile utilizzare le seguenti rappresentazioni: • **%o** - ottale • **%x** - esadecimale • **%n** - binario • **%b** - booleano
CONTROL Continuazione	**Esempio per la creazione di colonne, attributo 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>`

Identificatori dei tag	Significato
CONTROL Continuazione	**Esempio di attributo format:** `<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 Continuazione	**Attributi:** • **color_bk** L'attributo imposta il colore di sfondo del controllo. • **color_fg** L'attributo imposta il colore di sfondo del controllo. **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)". • **display_format** L'attributo definisce il formato di elaborazione della variabile specificata. Questo attributo deve essere applicato in caso di accesso a una variabile PLC – Float, in quanto l'accesso avviene con la lettura di una parola doppia. Sono ammessi i seguenti formati di dati: – FLOAT – INT – DOUBLE – STRING Espressioni (ad es. testo da visualizzare o elemento grafico) di una casella di riepilogo, casella grafica o casella combinata: **Sintassi:** <ITEM>Espressione</ITEM> <ITEM value ="<Valore>">Espressione</ITEM>

Identificatori dei tag	Significato
CONTROL Continuazione	**Esempio:** `<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = " combobox ">` `<ITEM>text1</ITEM>` `<ITEM>text2</ITEM>` `<ITEM>text3</ITEM>` `<ITEM>text4</ITEM>` `</CONTROL>` Se si deve assegnare un valore intero a piacere a un'espressione, occorre aggiungere l'attributo **value = "valore"** al tag. Al posto della numerazione progressiva, ora la variabile Control contiene il valore assegnato dell'item. **Esempio:** `<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> **Esempio di barra di avanzamento:** `<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>` **Esempio di casella di riepilogo:** `<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" />` • Aggiunta di elementi: Gli elementi vengono aggiunti con la funzione **additem** o **loaditem**. • Eliminazione del contenuto: Il contenuto viene eliminato con la funzione **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>`

Identificatori dei tag	Significato
CONTROL Continuazione	**Esempio 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 Continuazione	Il controllo Imagebox gestisce un'immagine in formato bitmap o GIF. Se l'immagine è più grande dell'area visualizzabile, il controllo visualizza delle barre di scorrimento. Per il controllo dell'area visibile, il sistema mette a disposizione la funzione CONTROL.IMAGEBOXSET. **Per ulteriori informazioni** vedere il capitolo "Funzioni predefinite (Pagina 158)". **Sintassi:** `<function name="control.imageboxget"> ... </function>`
CONTROL Continuazione	**Attributi:** • **disable** L'attributo blocca/consente l'immissione in un controllo Edit. • **tooltip** Un testo descrittivo viene visualizzato quando il cursore è posizionato sul controllo. • **factor** Fattore di conversione • **font** Indicazione di un tipo di carattere • **fontpixelsize** Altezza dei caratteri - Il valore deve essere indicato in numero di pixel e si riferisce a una risoluzione di 640x480 pixel. L'altezza carattere rappresentata è definita dal rapporto tra la risoluzione reale e la risoluzione di riferimento. **Esempio:** `<control name="c2" xpos="180" ypos="120" fontname="arial" fontpixelsize="26" height="100" type="string" />`

Identificatori dei tag	Significato
CONTROL Continuazione	**Attributo:** • **fontname** L'attributo permette di definire i font da utilizzare. I font installati possono essere definiti tramite la funzione HMI.GET_FONT_FAMILIES. **Valore:** Nome del font Nel sistema sono installati i seguenti font Siemens: • 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 I sistemi basati su MS Windows possono contenere ulteriori font. **Esempio:** `<control name="c2" xpos="180" ypos="120" fontname="arial" fontpixelsize="26" height="100" type="string" />`

Identificatori dei tag	Significato
CONTROL Continuazione	**Modifica di un controllo dopo la creazione** Un tag Control può essere usato per modificare le proprietà di un controllo esistente dopo la creazione. Il tag deve essere specificato con il nome del controllo da modificare e con le nuove proprietà. Può essere eseguito solo all'interno di un tag Form. Le seguenti proprietà possono ad es. essere modificate: • name • xpos • ypos • width • height • color_bk • color_fg • access level • fieldtype • itemdata • min • max • default • disable • tooltip • font • factor La variabile di riferimento non può essere modificata. Se una proprietà deve essere modificata da un evento softkey tramite trigger, si deve utilizzare il tag Send message per trasferire questa richiesta nel contesto del Form. Il tag message viene usato per rilevare il messaggio.

Identificatori dei tag	Significato
CONTROL Continuazione	**Modifica di un controllo in una istruzione di operazione** Un'altra possibilità di modifica di un controllo durante il runtime consiste nell'effettuare la modifica in una istruzione di operazione. A questo scopo occorre indicare il nome del controllo e la proprietà a cui va assegnato un nuovo valore. La proprietà è separata dal nome del controllo con un punto. **Sintassi:** <nome>.<proprietà> **Esempio:** `...` `...` `<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>`

Identificatori dei tag	Significato
HELP_CONTEXT	Questo tag definisce l'argomento della guida che si deve richiamare. Va programmato nel blocco INIT. **Sistemi 808/802D sl** Il nome indicato nell'attributo viene completato con il prefisso XmlUserDlg_ e inoltrato al sistema della guida. Per informazioni sulla struttura del file della guida vedere l'argomento Creazione della guida in linea. **Procedura per l'attivazione del sistema della guida:** 1. Premere il tasto "Info". 2. La finestra di dialogo fornisce l'espressione "my_dlg_help". 3. Il parser converte l'espressione in "XmlUserDlg_my_dlg_help" . 4. Attivazione del sistema della guida. 5. Immissione del termine di ricerca "XmlUserDlg_my_dlg_help". **Sistemi 840D sl/828D** **Ulteriori informazioni** sulla struttura e la creazione di una guida: Manuale per la messa in servizio SINUMERIK Operate **Attributi:** • **name** Il nome indicato nell'attributo viene inoltrato al sistema della guida. Deve essere indicato il nome file utilizzato nel registro della guida nel tag **entry**. • **anchor** L'attributo consente di specificare la parola chiave. **Sintassi:** `<HELP_CONTEXT name="<file name>" anchor="key word"/>` **Esempio:** `<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>`

Identificatori dei tag	Significato
DATA_ACCESS	Il tag comanda il comportamento delle finestre di dialogo nel salvataggio delle immissioni dell'utente. Il comportamento deve essere definito all'interno del tag INIT. Se il tag non viene utilizzato, i dati vengono sempre salvati in modo temporaneo. Eccezione: I controlli per i quali l'attributo **hotlink** è impostato a **true** vengono sempre scritti e letti direttamente. **Attributo:** • **type** = "TRUE" – i valori immessi non vengono salvati in modo temporaneo. I valori immessi copia la finestra di dialogo direttamente nella variabile di riferimento. • **type =** "FALSE" – i valori vengono copiati nella variabile di riferimento solo con il tag **UPDATE_CONTROLS** type = "FALSE" **Esempio:** `<DATA_ACCESS type = "true" />`

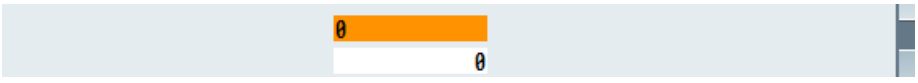
Identificatori dei tag	Significato
DEBUG	Il tag permette di salvare i dati Trace emessi che possono essere programmati nello script. Per la descrizione del testo da salvare si deve utilizzare l'attributo **text**. Questo attributo e i rispettivi valori corrispondono all'attributo **text** e ai valori dell'istruzione Print. Di norma un tag debug non viene eseguito, dato che provoca un rallentamento del sistema. Se si desidera attivare l'emissione di dati di Debug, nel tag **DialogGui** o **AGM** si deve programmare l'attributo **debug_msg** con il valore **on**. Il salvataggio dei dati Debug emessi avviene insieme ai messaggi di errore del parser nel seguente file: card/oem/sinumerik/hmi/dvm/log/dvm.log **Sintassi:** `<debug text="<per il testo vedere l'istruzione Print>">per le variabili vedere l'istruzione Print</debug>` **Esempio 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>` Immissione nel file dvm.log: `date: 10/22/ 2018 time: 09:38:44:351` `Debug xmldial.xml 17: main_form control test_bool created value:255` **Esempio 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>` Immissione nel file dvm.log: `date: 10/22/ 2018 time: 09:38:44:351` `Debug xmldial.xml 17: main_form control test_bool created value:255`

Identificatori dei tag	Significato
EDIT_CHANGED	**Messaggio della finestra di dialogo** Il tag viene richiamato quando cambia il contenuto di un controllo Edit. Per identificare il controllo, il sistema copia il nome del controllo nella variabile **$focus_name** e il valore dell'attributo **item_data** nella variabile **$focus_item_data**. **Esempio:** `<EDIT_CHANGED>` `<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>` … … `</EDIT_CHANGED>`
GESTURE_EVENT	Il tag viene utilizzato per l'esecuzione delle azioni delle dita nel comando Multitouch. Il tag è descritto nel capitolo "Comandi Multitouch (Pagina 206)".
INDEX_CHANGED	**Messaggio della finestra di dialogo** Il tag viene richiamato quando l'operatore modifica la selezione di una casella combinata. Per identificare il controllo, il sistema copia il nome del controllo nella variabile **$focus_name** e il valore dell'attributo **item_data** nella variabile **$focus_item_data**. **Nota:** Una variabile di riferimento assegnata al controllo non è ancora attualmente sintonizzata con la variabile Control e contiene l'indice della selezione precedente della casella combinata. **Esempio:** `<INDEX_CHANGED>` `<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>` … … `</INDEX_CHANGED>`
MENU	Il tag definisce un menu che contiene la descrizione dei softkey e la finestra di dialogo da aprire. **Attributo:** • **name** Nome del menu **Sintassi:** `<MENU name = "<menu name>">` … `<open_form …>` … `<SOFTKEY …>` `</SOFTKEY>` `</MENU>`

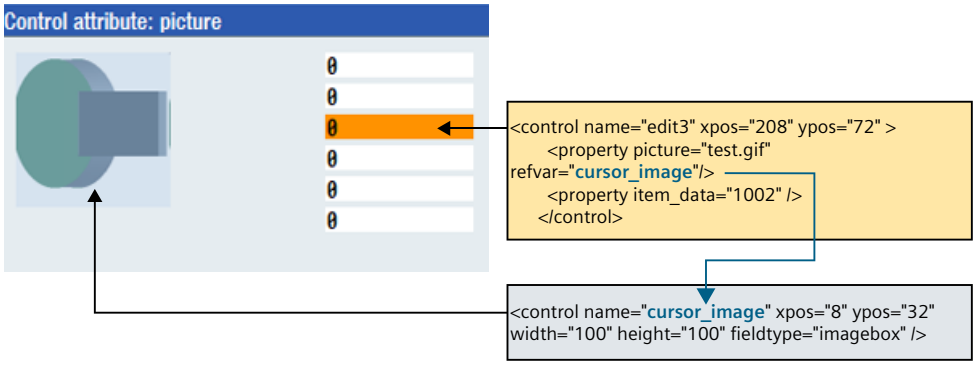
Identificatori dei tag	Significato
NAVIGAZIONE	Questo tag definisce il menu che si deve richiamare. Può essere impiegato solo all'interno di un blocco di softkey, di un blocco di menu e in un Form. Se al tag viene assegnato come valore un nome di variabile, il parser attiva il menu evidenziato nella variabile. In un blocco di menu avviene la navigazione sulla posizione dell'istruzione. Le istruzioni successive non vengono più eseguite. **Nota**: Per definire la destinazione di navigazione mediante il contenuto di una variabile, questa va dichiarata come variabile globale. **Sintassi:** `<NAVIGATION>menu name</NAVIGATION>` **Esempio:** `<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>`

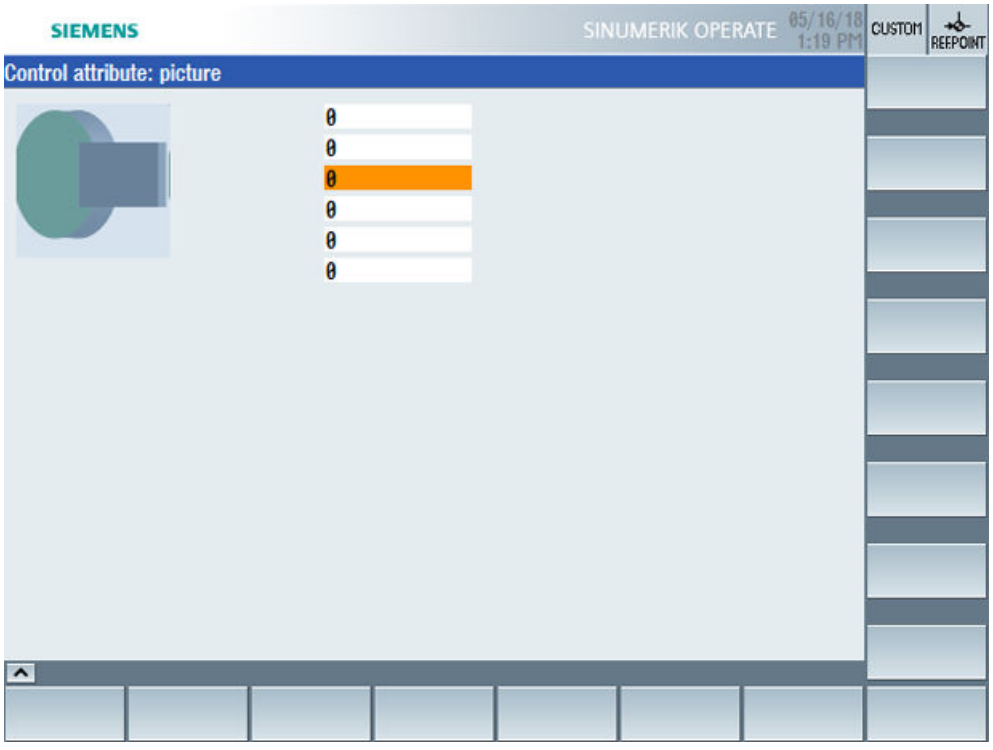
Identificatori dei tag	Significato
OPEN_FORM	Il tag apre la finestra di dialogo specificata con il nome. Se la maschera di dialogo specificata è già attiva, questo tag non viene eseguito. La riapertura ripetuta della finestra di dialogo può essere forzata specificando l'attributo **reopen**. Questa operazione può essere necessaria in caso di cambiamento degli stati sistema che presuppongono una modifica dei contenuti della maschera. **Attributo:** • **name** Nome della finestra di dialogo • **reopen** Se il valore dell'attributo è impostato a **true**, in caso di richiamo ripetuto del tag **open_form** viene verificato se si tratta della maschera di dialogo attiva. In caso affermativo, il parser chiude il form attivo quindi lo riapre. **Sintassi:** `<OPEN_FORM name = "<form name>" />` **Esempio:** `<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>`

Identificatori dei tag	Significato
PROPERTY	Questo tag permette di definire proprietà aggiuntive per un elemento di comando. Il tag viene elaborato ciclicamente nel tag Control. **Attributi:** • **max** = "<valore massimo>" • **min** = "<valore minimo>" • **default** = "<preassegnazione>" • **factor** = "fattore di conversione" • **color_bk** = "<codifica colore sfondo>" • **color_fg** = "<codifica colore scrittura>" • **password** = "<true>" - I caratteri immessi sono visualizzati con "*" • **multiline** = "<true>" - Consente di immettere dati su più righe in un controllo Edit • **disable** = "<true/false>" - Consente/blocca l'immissione di dati in un controllo Edit • **transparent** = "colore trasparente di un bitmap" **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)". • **tooltip** = il testo di avvertenza viene visualizzato quando il cursore è posizionato sul controllo. La sintassi è la seguente: <property tooltip="testo avvertenza" /> • **abscissa** = "nome del primo asse delle coordinate" (ammesso solo per casella grafica) • **ordinate** = "nome del secondo asse delle coordinate" (ammesso solo per casella grafica) • **fix_time_area** = "on" - Gli attributi **min** e **max** definiscono l'intervallo di tempo nel quale devono essere rappresentati i segnali. I segnali rappresentati vengono spostati da destra a sinistra. **Esempio:** `<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>`

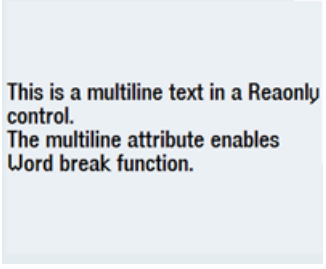
Identificatori dei tag	Significato
PROPERTY continua	**Attributo:** **alignment** - Questa proprietà consente l'allineamento del testo a sinistra o a destra. Normalmente un testo viene allineato a sinistra. **Valori:** • **left** • **right** **Sintassi:** `alignment="<right/left>"` **Esempio:** `<control name="edit2" xpos="208" ypos="52" >` `<property alignment="right"/>` `</control>` 

Identificatori dei tag	Significato
PROPERTY continua	**Attributo:** **parent** - Questa proprietà definisce l'appartenenza del controllo. Normalmente i controlli sono assegnati al form. Se si desidera assegnare i controlli a una scrollview, questa deve essere specificata come finestra sovraordinata. **Valore:** Nome della finestra sovraordinata **Esempio:** `...` `<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"/>` **test** **test1** `<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" />` `...`

Identificatori dei tag	Significato
PROPERTY continua	**Attributo:** **picture** - Questo attributo definisce una figura da visualizzare La figura indicata viene visualizzata (o rispettivamente il nome della figura viene salvato in una variabile) quando il controllo di immissione viene a trovarsi su questo campo. Per poter definire il ricevitore dei dati si deve usare questo attributo insieme all'attributo **refvar**. Per visualizzare una figura o un'animazione si deve specificare il nome di un controllo del tipo **imagebox**. Questo controllo assume la gestione della figura. Se si specifica il nome di una variabile locale, quest'ultima deve avere il tipo di dati **String**. La figura viene visualizzata fintanto che non si assegna un nuovo nome alla variabile di riferimento. **Sintassi:** `<property picture="<Name der Grafik>" refvar="<Datensenke>"/>`
PROPERTY continua	**Esempio di interconnessione di campi:** 

Identificatori dei tag	Significato
PROPERTY continua	**Esempio:** `<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>` 

Identificatori dei tag	Significato
PROPERTY continua	**Attributo:** **cursortext** - Questo attributo definisce il testo del cursore da visualizzare Il testo viene emesso nella riga del titolo quando il controllo di immissione viene a trovarsi su questo campo. Per questo attributo si possono indicare altri due attributi che definiscono l'orientamento del testo e il campo di testo del cursore. Per impostazione predefinita il testo è allineato a sinistra. Il campo del cursore di testo occupa di default il 50 % della larghezza della riga del titolo. `alignment="<right/left>"` - Allineamento del testo a sinistra/destra `length="<Larghezza>"` - Quota percentuale che il testo del cursore deve occupare nella riga del titolo **Sintassi:** `<property cursortext="<text>" />` oppure `<property cursortext="<text>" alignment="<right/left>" />` oppure `<property cursortext="text2" alignment="r="<text>" alignment="<right/left>" length="<Rapporto>"/>` **Esempio:** `<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>` 

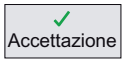
Identificatori dei tag	Significato
PROPERTY continua	**Attributo:** **multiline** - Consente di emettere un testo su più righe in un controllo Edit o Readonly **Sintassi:** `<property multiline="true" />` **Esempio:** `...` `...` `<op>` `textlist[2] = _T"This is a multiline text in a Reaonly control.\` `\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 continua	**Attributi:** • **help_context** L'attributo consente di assegnare un argomento della guida a un controllo. Deve essere indicato il nome file utilizzato nel registro della guida nel tag **entry**. • **anchor** L'attributo consente di specificare la parola chiave. Questo attributo è valido solo insieme all'attributo **help_context**. **Sintassi:** `<property help_context="<file name>" anchor="<key word>"/>` **Esempio:** `<control name = "cpwd" xpos = "322" ypos = "64" >` `<property password="true" />` `<property help_context="chapter_1.html" anchor="Keyword_2" />` `</control>`

Identificatori dei tag	Significato
SOFTKEY	Il tag definisce le proprietà e le reazioni di un softkey. **Attributi:** • **position** Numero del softkey. 1-8 softkey orizzontali, 9-16 softkey verticali Gli attributi seguenti diventano efficaci a partire da: • **type** Definisce la proprietà di un softkey. user_controled - La rappresentazione del softkey è definita dallo script toggle_softkey - Il softkey viene rappresentato alternativamente premuto o non premuto • **refvar** Va utilizzato solo in combinazione con il tipo toggle_softkey . Variabile di riferimento nella quale viene copiata la proprietà del softkey attuale. Va specificata una variabile del tipo "String" che contiene le proprietà premuto, non premuto o bloccato (vedere il tag state). • **picture** Questo attributo permette di emettere un bitmap allineato a sinistra sul softkey. Occorre specificare il nome del percorso completo. Il numero di caratteri di testo visualizzabili si riducono in funzione della larghezza del bitmap.

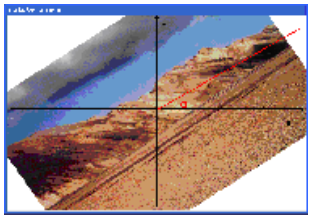
Identificatori dei tag	Significato
SOFTKEY Continuazione	Inoltre all'interno del blocco softkey possono essere definite le seguenti azioni: • **picturealignment** Questo attributo permette di impostare l'allineamento dell'immagine. Di norma l'immagine è allineata alla sinistra del softkey. Per l'allineamento possono essere specificati i seguenti valori: – **top** Bordo superiore – **bottom** Bordo inferiore – **left** Bordo sinistro – **right** Bordo destro – **center** Centrato • **caption** Testo del softkey • **state** Va utilizzato solo in combinazione con il tipo user_controlled . Il tag assegna al softkey la visualizzazione desiderata. **Sintassi:** <state type="<state>" /> Possono essere specificate le seguenti stringhe: – **notpressed** Il softkey viene visualizzato non premuto. – **pressed** Il softkey viene visualizzato premuto. – **disabled** Il softkey è bloccato e viene visualizzato in grigio. • **navigation** • **update_controls** • **function**

Identificatori dei tag	Significato
SOFTKEY Continuazione	**Sintassi:** Softkey standard: `<state type="<softkey state>" />` `<softkey position = "<1>">` … … `</softkey>` oppure Softkey comandato da script: `<softkey position = "<1>" type="<user_defined>" >` `<state type="<softkey state>" />` … … `</softkey>` oppure Softkey toggle: `<softkey position = "<1>" type="<toggle_softkey>" refvar="<variable name>" >` … … `</softkey>` **Esempio:** `<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>`

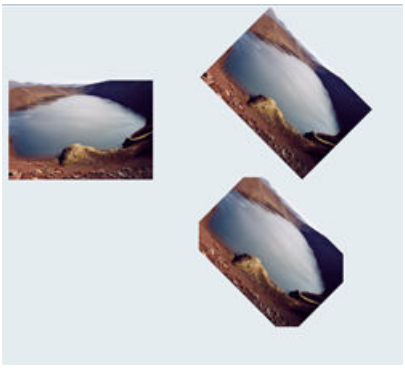
Identificatori dei tag	Significato
SOFTKEY Continuazione	**Esempio:** oppure `<let name="curr_softkey_state" type="string">PRESSED</let>` `</softkey>` `<softkey POSITION="3" type="toggle_softkey" refvar="curr_softkey_state">` `<caption>Toggle%nSK</caption>` ... </softkey>  
SOFTKEY_OK	Il tag definisce la reazione del softkey "OK".  Inoltre all'interno del blocco softkey possono essere definite le seguenti azioni: • **navigation** • **update_controls** • **function** **Sintassi:** `<SOFTKEY_OK>` ... ... `</SOFTKEY_OK>`
SOFTKEY_CANCEL	Il tag definisce la reazione del softkey "Interruzione".  Inoltre all'interno del blocco softkey possono essere definite le seguenti azioni: • **navigation** • **update_controls** • **function** **Sintassi:** `<SOFTKEY_CANCEL>` ... ... `</SOFTKEY_CANCEL>`

Identificatori dei tag	Significato
SOFTKEY_BACK	Il tag definisce la reazione del softkey "Indietro".  Inoltre all'interno del blocco softkey possono essere definite le seguenti azioni: • **navigation** • **update_controls** • **function** **Sintassi:** `<SOFTKEY_BACK>` ... ... `</SOFTKEY_BACK>`
SOFTKEY_ACCEPT	Il tag definisce la reazione del softkey "Accettazione".  Inoltre all'interno del blocco softkey possono essere definite le seguenti azioni: • **navigation** • **update_controls** • **function** **Sintassi:** `<SOFTKEY_ACCEPT>` ... ... `</SOFTKEY_ACCEPT>`

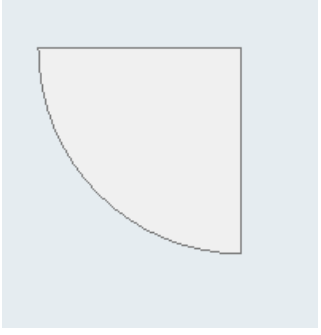
Identificatori dei tag	Significato
TESTO	Il tag permette di visualizzare un testo nella posizione specificata. Se viene usato un numero di allarme, la finestra di dialogo definisce il testo memorizzato per il numero. **Sintassi:** `<TEXT xpos = "<Posizione X>" ypos = "<Posizione Y>"> Text </TEXT>` **Attributi:** • **xpos** Posizione X dell'angolo superiore sinistro • **ypos** Posizione Y dell'angolo superiore sinistro • **color** Colore del testo **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)". **Valore:** testo da visualizzare
IMG	Il tag permette di visualizzare un'immagine nella posizione specificata. Sono supportati i formati immagine BMP e PNG. **Sintassi:** `<IMG xpos = "<Posizione X>" ypos = "<Posizione Y>" name = "<nome>" />` `<IMG … xrot= "angolo asse X" yrot= "angolo asse Y" zrot= "angolo asse Z" />` **Attributi:** • **xpos** Posizione X dell'angolo superiore sinistro • **ypos** Posizione Y dell'angolo superiore sinistro • **name** Nome del percorso completo. Il nome del percorso deve essere indicato in lettere minuscole. • **transparent** Colore trasparente del bitmap **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)".

Identificatori dei tag	Significato
IMG Continuazione	**Esempio**: L'immagine viene ruotata di 34 gradi intorno all'asse Z: `<img xpos = "5" ypos = "23" name = "f:/appl/pic3.bmp" height="355" width="550" zrot="34" color_bk="#ffffff" ></img>` **Nota**: La denominazione del drive varia a seconda del sistema.  **Opzionale:** Se la visualizzazione dell'immagine deve essere diversa dalle dimensioni originali, è possibile definire le dimensioni con gli attributi **width** e **height**. • **width** Larghezza in pixel • **height** Altezza in pixel **Esempi:** `<IMG xpos = "20" ypos = "40" name = "f:/appl/test.bmp" />` `<IMG xpos = "5" ypos = "23" name = "f:/appl/test.bmp" height = "355" width = "550"/>`

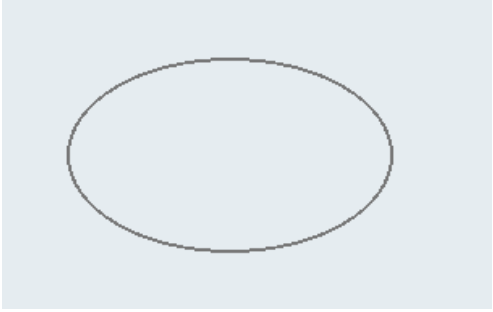
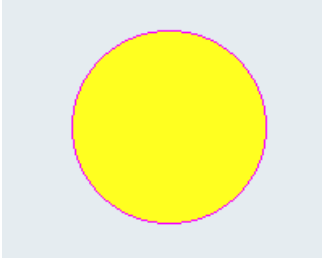
Identificatori dei tag	Significato
IMG Continuazione	**Attributo:** **AspectRatioMode** Questo attributo permette di controllare il rapporto dei lati di una figura quando si effettua uno zoom non proporzionale. **Valori:** • **Ignore** Il rapporto dimensionale dei lati della bitmap viene adattato all'altezza e alla larghezza impostati. • **Keep** (Standard) Il rapporto dimensionale dei lati viene mantenuto affinché l'immagine venga scalata su un rettangolo che occupi la massima estensione possibile entro l'altezza e la larghezza definita. • **KeepByExpanding** Il rapporto dimensionale dei lati viene mantenuto affinché l'immagine venga scalata su un rettangolo che occupi la minima estensione possibile al di fuori dell'altezza e della larghezza definita. **Esempio:** `<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>`  • In alto - è impostata la proprietà **KeepbyExpanding** • A sinistra in basso - è impostata la proprietà **Ignore** • A destra in basso - è impostata la proprietà **Keep**

Identificatori dei tag	Significato
IMG Continuazione	**Attributi:** **ExpandingForRotation** Viene preservato il rapporto dimensionale tra i lati. L'immagine viene scalata su un rettangolo che corrisponde alla Bounding-Box dell'immagine ruotata e scalata. **Esempio:** `<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>`  • Sinistra - immagine scalata alla dimensione di 150x155 pixel • Destra in alto - immagine scalata alla dimensione di 150x155 pixel e ruotata di 45 gradi con la proprietà **ExpandingForRotation** • Destra in basso - immagine scalata alla dimensione di 150x155 pixel e ruotata di 45 gradi

Identificatori dei tag	Significato
ARC	Il tag permette di disegnare un settore circolare in un form. **Attributi:** Posizione all'interno del form in pixel: • **xpos1** - Punto iniziale del segmento di cerchio coordinata X • **ypos1** - Punto iniziale del segmento di cerchio coordinata Y • **xpos2** - Punto finale del segmento di cerchio coordinata X • **ypos2** - Punto finale del segmento di cerchio coordinata Y Posizione all'interno del form in pixel: • **ccx** - Punto centrale del segmento di cerchio coordinata X • **ccy** - Punto centrale del segmento di cerchio coordinata Y • **radius** - Raggio del cerchio, valore in pixel **Nota:** **Punto centrale o raggio** Se si indicano entrambi i parametri, viene utilizzato il punto centrale del cerchio.
ARC continuazione	• **penwidth** Questo attributo definisce lo spessore della linea in pixel. • **color** Colore della linea • **color_bk** Colore di riempimento del segmento di cerchio **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)". • **style** Questo attributo definisce il tipo di linea: – **"SolidLine"** - Linea continua (predefinito) – **"DashLine"** - Linea tratteggiata – **"DotLine"** - Linea a puntini – **"DashDotLine"** - Punto-tratto-linea – **"DashDotDotLine"** - Tratto-punto-punto linea

Identificatori dei tag	Significato
ARC continuazione	• **type** – **cw** - clockwise (senso orario) – **ccw** - counterclockwise (senso antiorario) • **arrow** Alle estremità dei segmenti sono rappresentate delle frecce: – **"P1"** - Freccia sul punto iniziale della linea – **"P2"** - Freccia sul punto iniziale della linea – **"P1P2"** - Freccia sul punto iniziale e sul punto finale della linea – **"P1_FILLED"** - Freccia sul punto iniziale della linea riempita – **"P2_FILLED"** - Freccia sul punto iniziale della linea riempita – **"P1P2_FILLED"** - Freccia sul punto iniziale e sul punto finale della linea riempita • **arrow_size** Se si utilizza l'attributo Arrow, è possibile indicare la lunghezza della freccia in pixel.
ARC continuazione	**Esempi:**  `<arc xpos1="20" ypos1="200" xpos2="180" ypos2="200" ccx="100" ccy="200" type="ccw" PENWIDTH="1" color="#777777" arrow="p1p2_filled"/>` oppure `<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"/>`

Identificatori dei tag	Significato
BOX	Il tag traccia un rettangolo pieno nella posizione specificata nel colore specificato. **Sintassi:** `<BOX xpos = "<Posizione X>" ypos = "<Posizione Y>" width = "<Estensione X>" height = "<Estensione Y>" color = "<Codifica colore>" />` **Attributi:** • **xpos** Posizione X dell'angolo superiore sinistro • **ypos** Posizione Y dell'angolo superiore sinistro • **width** Estensione in direzione X (in pixel) • **height** Estensione in direzione Y (in pixel) • **color** Codifiche di colore **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)".
ELLIPSE	Il tag permette di disegnare un'ellisse in un form. **Attributi:** Posizione all'interno del form in pixel: • **xpos** - Punto iniziale della linea coordinata X • **ypos** - Punto iniziale della linea coordinata Y • **width** - Larghezza del rettangolo circoscritto (bounding box) • **height** - Altezza del rettangolo circoscritto (bounding box) • **penwidth** Questo attributo definisce lo spessore della linea in pixel. • **color** Colore della linea • **color_bk** Colore di riempimento dell'ellisse **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)". • **style** Questo attributo definisce il tipo di linea: – **"SolidLine"** - Linea continua (predefinito) – **"DashLine"** - Linea tratteggiata – **"DotLine"** - Linea a puntini – **"DashDotLine"** - Punto-tratto-linea – **"DashDotDotLine"** - Tratto-punto-punto linea

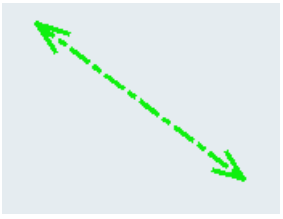
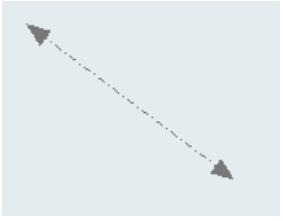
Identificatori dei tag	Significato
ELLIPSE continuazione	**Esempi:**  `<ellipse xpos="302" ypos="160" width="100" height="60" PENWIDTH="2" color="#777777" />`  `<ellipse xpos="402" ypos="360" width="60" height="60" PENWIDTH="1" color="#f800ff"  color_bk="#ffff20"/>`

Identificatori dei tag	Significato
FUNCTION	**Richiamo della funzione** Il tag esegue il corpo della funzione specificato con l'attributo "**name**" . **Attributi:** • **name** = "Nome del corpo della funzione" • **return** = "Nome della variabile per la memorizzazione del risultato della funzione" **Valori:** Lista delle variabili che devono essere trasferite al corpo della funzione. Le variabili devono essere separate tra loro con una virgola. Possono essere trasferiti al massimo 10 parametri. Inoltre è possibile indicare costanti o espressioni testuali come parametri di richiamo. Per identificare un'espressione testuale occorre anteporre l'identificativo _T al testo. **Sintassi:** `<FUNCTION name = "<function name>" />` La funzione da richiamare prevede un valore di ritorno `<FUNCTION name = "<function name>" return = "<Variablenname>" />` Assegnazione di parametri `<FUNCTION name = "<function name>"> var1, var2, var3 </FUNCTION>` `<FUNCTION name = "<function name>"> _T"Text", 1.0, 1 </FUNCTION>` **Esempi:** Vedere "FUNCTION_BODY"

Identificatori dei tag	Significato
FUNCTION_BODY	**Corpo della funzione** Il tag contiene il corpo della funzione di una sottofunzione. Il corpo della funzione deve essere programmato all'interno del tag DialogGui. **Attributi:** • **name** = "Nome del corpo della funzione" • **parameter** = "Lista parametri" (opzionale) L'attributo elenca i parametri di trasferimento necessari. I parametri devono essere separati tra loro con una virgola. Al richiamo del corpo della funzione i valori dei parametri specificati nel richiamo della funzione vengono copiati nei parametri di trasferimento indicati. • **return** = "true" (opzionale) Se l'attributo è impostato a true, viene creata la variabile locale $return. In questa variabile va copiato il valore di ritorno della funzione, che al momento della chiusura della funzione viene inoltrato alla funzione da richiamare. **Sintassi:** **Corpo della funzione senza parametro** `<FUNCTION_BODY name = "<function name>">` … … … `</ FUNCTION_BODY>` **Corpo della funzione con parametro** `<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>">` … `<LET name = "tmp></LET>` `<OP> tmp = p1 </OP>` … `</FUNCTION_BODY>` **Corpo della funzione con valore di ritorno** `<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>`

Identificatori dei tag	Significato
FUNCTION_BODY Continuazione	**Esempio:** <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	Il tag permette di disegnare una linea in un form. **Attributi:** Posizione all'interno del form in pixel: • **xpos1 -** Punto iniziale della linea coordinata X • **ypos1** - Punto iniziale della linea coordinata Y • **xpos2** - Punto finale della linea coordinata X • **ypos2** - Punto finale della linea coordinata Y • **penwidth** Questo attributo definisce lo spessore della linea in pixel. • **color** Colore della linea **Per ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)". • **style** Questo attributo definisce il tipo di linea: – **"SolidLine"** - Linea continua (predefinito) – **"DashLine"** - Linea tratteggiata – **"DotLine"** - Linea a puntini – **"DashDotLine"** - Punto-tratto-linea – **"DashDotDotLine"** - Tratto-punto-punto linea

Identificatori dei tag	Significato
LINE continuazione	• **arrow** Alle estremità delle linee sono rappresentate delle frecce: – **"P1"** - Freccia sul punto iniziale della linea – **"P2"** - Freccia sul punto iniziale della linea – **"P1P2"** - Freccia sul punto iniziale e sul punto finale della linea – **"P1_FILLED"** - Freccia sul punto iniziale della linea riempita – **"P2_FILLED"** - Freccia sul punto iniziale della linea riempita – **"P1P2_FILLED"** - Freccia sul punto iniziale e sul punto finale della linea riempita • **arrow_size** Se si utilizza l'attributo Arrow, è possibile indicare la lunghezza della freccia in pixel. **Sintassi:** `<line xpos1="<punto iniziale coordinata X>" ypos1="<punto iniziale coordinata Y>" xpos2="<punto finale coordinata X>" ypos2="<punto iniziale coordinata Y>"` `PENWIDTH="<spessore tratto>" color="<colore linea>" style="<stile linea>" arrow="<tipo di freccia>" arrow_size="<dimensioni freccia>"/>`

Identificatori dei tag	Significato
LINE continuazione	**Esempi:** `<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"/>` 

Identificatori dei tag	Significato
REQUEST	Questo tag consente di inserire una variabile nel servizio di lettura ciclico (hotlink). Si riduce così il tempo di accesso alle variabili che non sono legate a un controllo. Se in caso di modifica del valore deve essere richiamata automaticamente una funzione, come ulteriore attributo va specificato il nome della funzione. Questo tag viene elaborato solo all'interno dell'istruzione INIT. **Attributi:** • **name** Identificatore di indirizzo • **function** Nome della funzione **Sintassi:** `<REQUEST name = "<NC-Variable>" />` oppure `<REQUEST name = "<NC-Variable>" function="<function name>"/>` **Esempio:** `<request name ="plc/mb10" />` oppure `<function_body name="my_function" >` `<print text="value changed" />` `</function_body>` … … … `<request name ="plc/mb10" function="my_function"/>`

Identificatori dei tag	Significato
RECALL	Questo tag può essere utilizzato in un menu se deve avvenire una navigazione tramite il tasto Recall. Se il tag è programmato, il simbolo Recall viene visualizzato e l'elaborazione del tasto Recall viene autorizzata dal parser. **Sintassi:** `<recall>` ... ... `</recall>`
UPDATE_CONTROLS	Il tag esegue una compensazione tra gli elementi di comando e le variabili di riferimento. **Attributo:** • **type** L'attributo definisce la direzione della compensazione dati. **= TRUE** – I dati vengono letti dalle variabili di riferimento e copiati negli elementi operativi. **= FALSE** – I dati vengono letti dagli elementi operativi e copiati nelle variabili di riferimento. **Sintassi:** `<UPDATE_CONTROLS type = "<Direzione>"/>` **Esempio:** `<SOFTKEY_OK>` `< UPDATE_CONTROLS type="false"/>` `</SOFTKEY_OK>`

3.8 Creazione di menu utente

3.8.1 Creazione di maschere di cicli di lavorazione

La funzione Supporto cicli consente di creare e ricompilare automaticamente un richiamo ciclo attraverso la finestra di dialogo Form.

Per gestire questa funzionalità sono disponibili i seguenti tag:

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

Per contrassegnare una maschera di ciclo, nel tag **FORM** deve essere specificato l'attributo **type** con il valore **cycle**. Questa identificazione consente di elaborare l'istruzione **NC_INSTRUCTION**.

Esempio

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

Il tag **NC_INSTRUCTION** contiene il richiamo di ciclo da creare. Tutti i parametri di ciclo devono essere riservati mediante segnaposti.

Esempio

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

Il tag **CREATE_CYCLE** prepara i valori memorizzati nelle variabili dei segnaposti e genera l'istruzione NC.

Questa viene poi copiata nella variabile specificata.

Identificatori dei tag	Significato
NC_INSTRUCTION	Con questo tag viene definita l'istruzione NC da creare. Tutti i parametri di ciclo richiamati vengono creati automaticamente come variabili String di **FORM** e sono a disposizione di **FORM**. Presupposto: L'attributo FORM **type** è impostato al valore **CYCLE**. **Attributo:** • **refvar** Se al tag è assegnata una variabile di riferimento, tutti i parametri vengono preimpostati con i valori del blocco NC memorizzato nella variabile di riferimento. **Sintassi:** `<NC_INSTRUCTION> NC – istruzione con segnaposti </NC_INSTRUCTION>` **Esempio:** `<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>`

Identificatori dei tag	Significato
CREATE_CYCLE	Il tag genera un blocco NC la cui sintassi è definita dal valore del tag **NC_INSTRUCTION** . Prima della creazione dell'istruzione NC il parser richiama il tag CYCLE_CREATE_EVENT di FORM. Questo tag può essere usato per il calcolo dei parametri di ciclo. **Sintassi:** `<CREATE_CYCLE/>` **Opzione:** Se viene specificata una variabile di riferimento, l'istruzione copia in questa variabile la chiamata generata. `<create_cycle refvar="name" />` **Attributo:** • **refvar** Se al tag è assegnata una variabile di riferimento, l'istruzione NC viene copiata in questa variabile. **Esempio:** `<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>` oppure `<SOFTKEY_OK>` `<caption>OK</caption>` `<CREATE_CYCLE refvar= "cyc_string" />` `<close_form />` `<navigation>main_menu</navigation>` `</SOFTKEY_OK>`

3.8.2 Caratteri sostitutivi

Il sistema offre la possibilità di definire le proprietà del controllo (valori degli attributi) durante il runtime. Per poter utilizzare questa funzione, occorre rendere disponibile la proprietà desiderata in una variabile locale e trasferire il nome della variabile al tag come valore dell'attributo preceduto da un **carattere $**.

Se il tag prevede una stringa come valore dell'attributo o valore, occorre anteporre i caratteri $ $$ al nome della variabile.

Esempio:

```
<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 Creazione del file struct_def.xml

Procedura

In alcune funzioni, per la definizione del tipo viene utilizzato il file **struct_def.xml**. Se manca questo file XML, si deve creare una copia del file con la seguente istruzione e integrarla nel progetto funzionale.

Nota

Creare il file XML in formato 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 Indirizzamento di componenti

Per indirizzare variabili NC, blocchi PLC o dati dell'azionamento, occorre creare identificatori di indirizzo sul dato desiderato. Un indirizzo è costituito dai percorsi parziali **nome del componente** e **indirizzo della variabile**. Come carattere di separazione va utilizzata una barra inclinata.

3.10.1 Indirizzamento PLC

L'indirizzamento del PLC inizia con la parte di percorso **plc**.

Tabella 3-3 Sono ammessi i seguenti indirizzi:

DBx.DB(f)	Blocco dati
I(f)x	Ingresso
Q(f)x	Uscita
M(f)x	Merker
V(f)x	Variabile

DBx.DBXx.b	Blocco dati
Ix.b	Ingresso
Qx.b	Uscita
Mx.b	Merker
Vx.b	Variabile

Tabella 3-4 Formato dati **f**:

B	Byte
W	parola
D	Parola doppia

In un indirizzamento bit manca l'identificazione del formato dati.

Indirizzo **x**:

Identificatore di indirizzo S7 200 valido

Indirizzamento bit:

b – numero bit

Esempi:

```
<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 Indirizzamento di variabili NC

L'indirizzamento di variabili NC inizia con la parte di percorso **nck**.

A questa parte segue l'indirizzo del dato, la cui struttura è descritta nel Manuale delle liste Variabili NC.

Esempio:

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

3.10.3 Indirizzamento specifico per canale

Se nel token dell'indirizzo non è specificato un numero di canale, l'accesso avviene sempre al canale 1 del software operativo.

Se necessario leggere i dati da un canale speciale, viene aggiunto all'indirizzo l'identificativo **u** (Unit) con il numero di canale desiderato.

Esempio:

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

3.10.4 Creazione di indirizzi NC/PLC durante il runtime

È possibile creare un identificatore di indirizzo durante il runtime.

Per fare questo il contenuto di una variabile String viene utilizzato come indirizzo in un'istruzione di operazione, oltre che nelle funzioni nc.cap.read e nc.cap.write.

Per questa modalità di indirizzamento occorre fare attenzione a quanto segue:

- Scrivere il nome della variabile tra virgolette.
- Anteporre tre caratteri ‚$' al nome della variabile.

Sintassi:

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

Esempio:

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

3.10.5 Indirizzamento di componenti dell'azionamento

L'indirizzamento di componenti dell'azionamento inizia con la parte di percorso **drive**.

Segue l'identificazione dell'apparecchio di azionamento:

CU – Control Unit

DC – Drive Control (Motor Module)

CULNK – Moduli di ampliamento (HUB)

TM – Terminal Module

LM – Line Module

A questa parte viene aggiunto il parametro da impostare.

Esempio:

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

<!— lettura del valore r0002 sulla CU ->

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

<!— lettura del valore r0002 sulla NX1 ->
<OP> r0002_content = "drive/cu/r0002[CU2]" </OP>

<!— lettura del valore p107[0] sulla CU ->
<OP> p107_content = "drive/cu/p107[0]" </OP>
<PRINT text="%d"> p107_content </PRINT>

<!— lettura del valore p107[0] sulla CU ->

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

<!— lettura del valore p107[0] sulla NX1 ->

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

Indirizzamento degli oggetti dell'azionamento:

Per indirizzare singoli oggetti, dopo il parametro occorre specificare l'oggetto desiderato tra parentesi angolari.

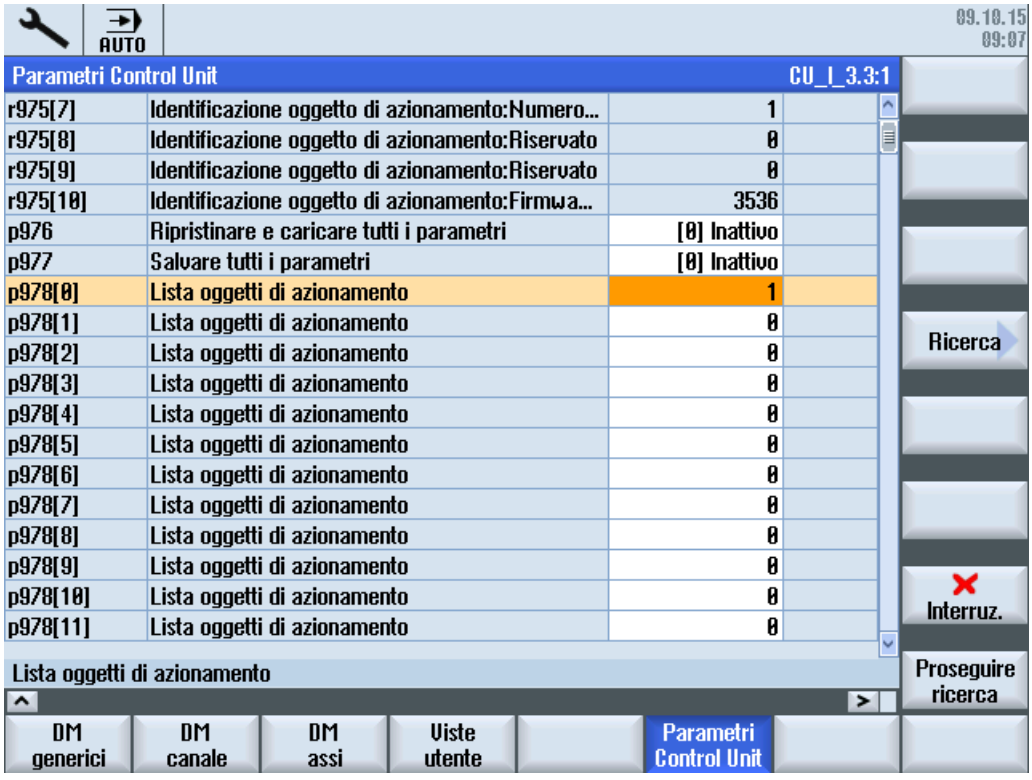


Figura 3-7 Parametro dell'azionamento p0978 [...] sul controllore

```
Numero parametro[do<DO-index>]
```

Esempio:
```
p0092[do1]
```

In alternativa è possibile leggere l'indice dell'azionamento da una variabile locale tramite "caratteri sostitutivi" **$<nome variabile>**.
```
z.B. DO$localVariable
```

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

Indirizzamento indiretto:
```
<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 Esempio: Rilevamento del numero DO per Motor Module

Il numero di DO di un Motor Module del tipo 11 (servo) può essere calcolato nel seguente modo:

Tutti i Drive Object collegati vengono elencati in base al numero di slot nel campo p978 della CU corrispondente. Contemporaneamente vengono elencati i numeri di componenti nel campo p101 e i tipi di componenti nel campo p107.

Per i seguenti tipi di componenti va utilizzata un'indicizzazione specifica:

CU – Control Unit

DC – Drive Control (Motor Module)

CULNK – Moduli di ampliamento (HUB)

TM – Terminal-Module

LM – Line-Module

L'indice di indirizzamento può essere determinato percorrendo, per ogni CU collegata, il campo p0107 in ordine crescente e incrementando l'indice di tipo di uno ad ogni occorrenza del tipo ricercato. Il valore di base è 1. Se in questo campo vengono trovati componenti NX, il conteggio prosegue fino al componente NX solo se l'array attuale è stato percorso completamente. L'elaborazione di componenti NX e CU avviene in questa sequenza.

Determinazione dell'indice: CUI con NX

CUI		NX1		Indice di indirizzamento	
				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	

In questa topologia sono contenuti sette Motor Module. Gli indici da 1 a 4 indirizzano i Motor Module assegnati alla CU. Gli indici da 5 a 7 indirizzano i Motor Module dell'NX. Per l'accesso alla CU va utilizzato l'indice 1. L'NX viene indirizzata con l'indice 2.

Esempi di script: xmldial.xml e 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 Indirizzamento di dati macchina e dati di setting

I dati dell'azionamento e i dati di setting sono contrassegnati con il carattere $, seguito dal nome del dato.

Dati macchina:

$Mx_<Nome[**index, AX<numero asse>**]>

Dati setting:

$Sx_<Nome[index, AX<numero asse>]>

x:

N – dati macchina e dati di setting generici

C – dati macchina e dati di setting specifici per canale

A – dati macchina e dati di setting specifici per asse

Indice:

Per un campo il parametro specifica l'indice del dato.

AX<numero asse>:

Per i dati specifici per asse deve essere specificato l'asse desiderato (**<numero asse>**).

In alternativa è possibile leggere l'indice asse da una variabile locale tramite "caratteri sostitutivi" **$<nome variabile>**.

es. AX$variabilelocale

Esempio:

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

Indirizzamento diretto dell'asse:

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

Indirizzamento indiretto dell'asse:

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

3.10.8 Dati macchina specifici per canale

Se nel token dell'indirizzo non è specificato un numero di canale, l'accesso avviene sempre al canale attualmente impostato del software operativo.

Se necessario leggere i dati da un canale speciale, viene aggiunto all'indirizzo l'identificativo **u** (Unit) con il numero di canale desiderato tra parentesi quadre. Nel caso di un indirizzamento di array l'indicazione del canale avviene come ultimo argomento tra parentesi quadre.

Esempio:

```
$MC_RESET_MODE_MASK
```

oppure

```
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]
```

```
$MC_RESET_MODE_MASK[u1]
```

oppure

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

3.10.9 Indirizzamento di dati utente

L'indirizzamento dei dati utente inizia con la parte del percorso **gud**, seguita dalla designazione che indica se si tratta di GUD globali o specifiche del canale. A questa parte di indirizzo seguono l'intervallo GUD e il nome GUD.

Se si tratta di un campo, dopo il nome occorre specificare l'indice di campo desiderato tra parentesi angolari.

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

Indirizzamento dei dati utente globali

L'indirizzamento inizia con la parte del percorso gud, seguita dall'identificatore di intervallo NCK. A questa parte di indirizzo segue l'identificatore degli intervalli GUD:

Intervalli GUD	Assegnazione
sgud	GUD di Siemens
mgud	GUD del costruttore della macchina
ugud	GUD dell'utente

Indirizzamento dei dati utente specifici del canale

L'indirizzamento inizia con la parte del percorso gud, seguita dall'identificatore di intervallo CHANNEL. A questa parte di indirizzo segue l'identificatore degli intervalli GUD.

Al termine va specificato il nome GUD. Se il campo deve essere indirizzato, al nome segue l'indice di campo tra parentesi angolari.

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

Indirizzamento di array pluridimensionali

Nell'indirizzamento di array pluridimensionali, è previsto che l'indice delle righe sia seguito dall'indice delle colonne. Gli indici devono essere separati tra di loro da un punto.

Per i GUD specifici del canale vale:

- Il numero di canale deve essere registrato con l'identificativo **u** (Unit) seguito dal numero prima o dopo la specifica della riga o della colonna.
- Le sequenze devono essere separate tra loro da una virgola.
- Se il numero di canale non è indicato, l'accesso avviene al primo canale.

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

oppure

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

3.11 Funzioni predefinite

Il linguaggio dello script offre varie funzioni di elaborazione delle stringhe e funzioni matematiche standard. I nomi delle funzioni elencati di seguito sono riservati e non possono essere utilizzati.

Nome della funzione	Significato
Ncfunc cap read	La funzione copia un valore dall'indirizzo specificato in una variabile locale. Se la lettura è avvenuta senza errori, la variabile Return contiene il valore zero. Contrariamente all'istruzione di operazione, questa funzione non interrompe l'elaborazione delle istruzioni script in caso di errore. **Attributi:** • **return** - Stato di esecuzione – Valore = 0 - Senza errori – Valore = 1 - Impossibile eseguire la lettura della variabile • **rows** - Numero delle righe aggiuntive da leggere di un array (opzionale) Se viene specificato un indice di array per la variabile di riferimento, la funzione copia i valori letti nella variabile di destinazione a partire da questo indice. **Sintassi:** `<function name="ncfunc.cap.read" return="error"> lokale variable, "address"</function>` **Esempio:** `<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>` oppure `<let name="cu_p978" dim="25" ></let>` `...` `...` `<function name="ncfunc.cap.read" return="error" rows="23">cu_p978, "drive/cu/p0978[0, 1]"` `</function>`

Nome della funzione	Significato
Ncfunc cap write	La funzione scrive un valore nella variabile specificata. Se la scrittura è avvenuta senza errori, la variabile Return contiene il valore zero. Contrariamente all'istruzione di operazione, questa funzione non interrompe l'elaborazione delle istruzioni script in caso di errore. **Attributi:** • **return** - Stato di esecuzione – Valore = 0 - Senza errori – Valore = 1 - Impossibile eseguire la lettura della variabile • **rows** - numero delle righe aggiuntive da scrivere di un array (opzionale) Se viene specificato un indice di array per la variabile di riferimento, la funzione copia i valori nella variabile di destinazione a partire da questo indice. **Sintassi:** `<function name="ncfunc.cap.read" return="error"> local variable or constant, "address"</function>` **Esempio:** `<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>` oppure `<let name="cu_p978" dim="25" ></let>` `...` `...` `<function name="ncfunc.cap.write" return="error" rows="23">cu_p978, "drive/cu/p0978[0, 1]"` `</function>`

Nome della funzione	Significato
Ncfunc PI-Service	Il servizio PI (istanza di programma) permette di trasferire ordini all'NCK. Se il servizio viene eseguito senza errori, la funzione restituisce il valore 1 nella variabile Return. **Manipolazione della lista utensili:** _N_CREATO - Creazione di un utensile _N_DELETO - Cancellazione di un utensile _N_CREACE - Creazione di un tagliente utensile _N_DELECE - Cancellazione di un tagliente utensile **Attivazione degli spostamenti origine**: _N_SETUFR - Attivazione dell'User Frame attuale _N_SETUDT - Attivazione dei dati utente attuali **Ricerca blocco**: _N_FINDBL - Attivazione della ricerca blocco _N_FINDAB - Interruzione della ricerca blocco **Riconfigurazione dei dati macchina:** _N_CONFIG - I dati macchina con il grado di efficacia NEW_CONF, immessi in successione dall'operatore vengono attivati contemporaneamente **Sintassi:** `<function name="ncfunc.pi_service" return="return var"> pi name, var1, var2, var3, var4, var5` `</function>` **Attributi:** name - Nome della funzione return - Nome della variabile in cui è memorizzato il risultato dell'esecuzione: • Valore = 1 - Job eseguito correttamente • Valore = 0 - Job errato **Valori del tag:** pi name - Nome del servizio PI (stringa) var1 ... var5 - Argomenti specifici di PI

Nome della funzione	Significato
Ncfunc PI-Service Continuazione	**Argomenti:** • _N_CREATO var1 - Numero dell'utensile • _N_DELETO var1 - Numero dell'utensile • _N_CREACE var1 - Numero dell'utensile var2 - Numero del tagliente • _N_DELECE var1 - Numero dell'utensile var2 - Numero del tagliente • _N_SETUFR Nessun argomento • _N_SETUDT var1 - Zona da attivare dei dati utente – 1 - Dati di correzione utensile – 2 - Frame di base attivo – 3 - Frame impostabile attivo • _N_FINDBL var1 - Modalità ricerca blocco – 2 - Ricerca blocco con calcolo del profilo – 4 - Ricerca blocco fino al punto finale del blocco – 1 - Ricerca blocco senza calcolo • _N_FINDAB Nessun argomento • _N_CONFIG Nessun argomento **Esempio:** Creazione di un utensile - Numero utensile 3 `<function name="ncfunc.pi_service">_T"_N_CREATO", 3</function>` Cancellazione del tagliente 1 dell'utensile 5 `<function name="ncfunc.pi_service">_T"_N_DELECE", 5, 1` `</function>`

Nome della funzione	Significato
ncfunc chan PI-Service	La funzione esegue un servizio PI riferito al canale. Il numero di canale viene trasferito dopo il nome di servizio PI. Dopodiché seguono tutti gli altri parametri di richiamo. **Parametri:** **channel** - Numero del canale **Sintassi:** `<function name="ncfunc.chan_pi_service" return="error">` `_T"_N_SETUFR", channel, ...` `</function>` **Esempio:** `<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 funzione fornisce la norma di conversione definita nel controllore per i numeri a virgola mobile. Come variabile va messa a disposizione una variabile String. Vedere anche i dati macchina di visualizzazione MD203 DISPLAY_RESOLUTION e MD204 DISPLAY_RESOLUTION_INCH **Sintassi:** `<function name="ncfunc.displayresolution" return="dislay_res" />` **Esempio:** `<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"/>`

Nome della funzione	Significato
Ncfunc Get drive by axis name	Questa funzione fornisce l'indice di indirizzamento di un Motor Module specificando i rispettivi nomi d'asse. La funzione restituisce il valore -1 se non è possibile determinare l'assegnazione. Questo può accadere, ad esempio, se non è avvenuta l'assegnazione asse/azionamento. **Parametri:** **str** - nome asse **Valore di ritorno:** Indice del Motor Module - Nome su una variabile in cui viene scritto l'indice calcolato. **Sintassi:** `<function name="NCFUNC.GETDRVBYAX_NAME" return="<index>">` `<axis name></function>` **Esempio:** `<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>`

Nome della funzione	Significato
Ncfunc Get drive by axis index	Questa funzione fornisce l'indice di indirizzamento di un Motor Module specificando il rispettivo indice d'asse. La funzione restituisce il valore -1 se non è possibile determinare l'assegnazione. Questo può accadere, ad esempio, se non è avvenuta l'assegnazione asse/azionamento. **Parametri:** **index** - indice asse **Valore di ritorno:** Indice del Motor Module - Nome su una variabile in cui viene scritto l'indice calcolato. **Sintassi:** `<function name="NCFUNC.GETDRVBYAX_IDX" return="<index>">` `<axis index></function>` **Esempio:** `<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>`

Nome della funzione	Significato
Ncfunc Get drive by drive name	Questa funzione fornisce l'indice di indirizzamento di un Motor Module specificando il rispettivo nome di Motor Module. La funzione restituisce il valore -1 se non è possibile determinare l'assegnazione. Questo può accadere, ad esempio, se non è avvenuta l'assegnazione asse/azionamento. **Parametri:** **string** - Nome del Motor Module **Valore di ritorno:** Indice del Motor Module - Nome su una variabile in cui viene scritto l'indice calcolato. **Sintassi:** `<function name="NCFUNC.GETDRVBYDRV_NAME" return="<index>">` `<drive name></function>` **Esempio:** `<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>`

Nome della funzione	Significato
Ncfunc Get drive by bud address	Questa funzione fornisce l'indice di indirizzamento di un Motor Module specificando l'indirizzo di bus dello stesso. La funzione restituisce il valore -1 se non è possibile determinare l'assegnazione. Questo può accadere, ad esempio, se non è avvenuta l'assegnazione asse/azionamento. **Parametri:** **bus** - Numero di bus **slave** - Numero di slave **component** - Numero di componente **Valore di ritorno:** Indice del Motor Module - Nome su una variabile in cui viene scritto l'indice calcolato. **Sintassi:** `<function name="NCFUNC.GETDRVBYBUS_ADDR" return="<index>"><bus>,<slave>,<component></function>` **Esempio:** `<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>`

Nome della funzione	Significato
Ncfunc Get MD limits	La funzione copia i limiti di un dato macchina nella variabile specificata del tipo di dati **StructMDLimits**. La struttura è definita nel file struct_def.xml. **Per ulteriori informazioni** vedere il capitolo "Creazione del file struct_def.xml (Pagina 142)". **Parametri:** **range** - Indicare una stringa che specifichi il settore dei dati macchina: • **displayMD** - Dati macchina di visualizzazione • **generalMD** - Dati macchina globali NC • **channelMD** - Dati macchina specifici NC • **axisMD** - Dati macchina NC specifici per asse • **generalSettingMD** - Dati setting globali NC • **channelSettingMD** - Dati setting NC specifici per canale • **axisSettingMD** - Dati setting NC specifici per asse • **channelGUD** - Dati utente NC specifici per canale • **globalGUD** - Dati utente globali NC **MD-number** - Numero di dato macchina del settore **variable name** - Una volta richiamata la funzione, questa variabile avrà i valori dei limiti **range/unit** - opzionale (ad es. numero di canale) **Sintassi:** `<function name="NCFUNC.GETMD_LIMITS"><range>, <MD-number, <variabel name>, <range/unit>></function>` **Esempio:** `<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 funzione converte in un valore intero una stringa specificata in formato BICO (vedere SINAMICS). **Sintassi:** `<function name="ncfunc.bicotoint" return="integer variable"> bico-string</function>` **Esempio:** `<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>`

Nome della funzione	Significato
Ncfunc int to bico	La funzione converte un valore intero in una stringa in formato BICO (vedere SINAMICS). **Sintassi:** `<function name="ncfunc.inttobico"` `return="string variable">integer variable</function>` **Esempio:** `<function name="ncfunc.inttobico"` `return="s_p0480_0">"drive/dc/p0480[0, DO2]"</function>`
Ncfunc is bico str valid	La funzione restituisce il valore zero se si tratta di una stringa specificata in formato BICO (vedere SINAMICS). **Sintassi:** `<function name="ncfunc.isbicostrvalid"` `return="integer variable">string varaible</function>` **Esempio:** `…` `<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>`
Ncfunc password	La funzione imposta o cancella un livello di password dell'NC. • Impostazione della password: Come parametro va impostata la password per il livello di password desiderato. • Eliminazione della password: Una stringa vuota elimina il livello di password. **Sintassi:** `<function name="ncfunc.password">password` `</function>` **Esempio:** `<let name="password" type="string"></let>` `<function name="ncfunc.password" > password` `</function>` `<function name="ncfunc.password" > _T"CUSTOMER"` `</function>` Eliminazione della password: `<function name="ncfunc.password" > _T"" </function>`

Nome della funzione	Significato
Control form color	La funzione fornisce il colore del testo o di sfondo della finestra di dialogo come stringa. Per **ulteriori informazioni** vedere il capitolo "Codifiche di colore (Pagina 75)". **Campo:** • BACKGROUND – richiesta del valore del colore dello sfondo • TEXT – richiesta del valore del colore del testo (primo piano) **Sintassi:** `<function name="control.formcolor" return="variable">_T"Settore"</function>` **Esempio:** `<let name="bk_color" type="string"></let>` `<function name="control.formcolor" return="bk_color">_T"BACKGROUND"</function>`
Control local time	La funzione copia l'ora locale in un campo con 7 elementi di campo. Come parametro di richiamo si prevede il nome della variabile. Nell'elemento di campo è memorizzato quanto segue: • Indice 0 - anno • Indice 1 - mese • Indice 2 - giorno della settimana • Indice 3 - giorno • Indice 4 - ora • Indice 5 - minuti • Indice 6 - secondi **Sintassi:** `<function name ="control.localtime">_T"time_array" </function>` **Esempio:** `<!-- 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>`

Nome della funzione	Significato
Control exist	La funzione restituisce il valore 1 se esiste il controllo specificato. **Sintassi:** `<function name="CONTROL.EXIST"` `return="<return variable>"><control name></function>` **Esempio:** `<let name="ret" /> <function name="CONTROL.EXIST"` `return="ret">_T"edit"</function>`
Control is modified	La funzione restituisce il valore 1 se il contenuto del controllo Edit specificato è stato modificato. **Sintassi:** `<function name="CONTROL.ISMODIFIED"` `return="<return variable>"><control name></function>` **Esempio:** `<let name="ret" /> <function name="CONTROL.ISMODIFIED"` `return="ret">_T"edit"</function>`
Control is visible	La funzione restituisce il valore 1 se il controllo specificato è visibile. **Sintassi:** `<function name="CONTROL.ISVISIBLE"` `return="<return variable>"><control name></function>` **Esempio:** `<let name="ret" /><function name="CONTROL.ISVISIBLE"` `return="ret">_T"edit"</function>`
Control set modified	La funzione modifica il flag Modified del controllo Edit modificato. **Parametri:** **control name** • Valore = 1 - contrassegnare il campo come modificato • Valore = 0 - contrassegnare il campo come non modificato **Sintassi:** `<function name="CONTROL.SETMODIFIED"` `return="<return variable>"><control name>,` `<Flag></function>` **Esempio:** `<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>`

Nome della funzione	Significato
Control set working area	Se si modifica il contenuto di una **scrollview**, ad es. mediante cancellazione o inserimento di controlli, l'area visibile della scrollview deve essere ricalcolata. Il calcolo viene eseguito mediante il richiamo di questa funzione. **Valore:** Nome della **scrollview** **Esempio:** `...` `<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	Due stringhe vengono confrontate tra loro dal punto di vista lessicografico. La funzione restituisce il valore zero se le stringhe sono uguali, minore di zero se la prima stringa è minore della seconda o maggiore di zero se la seconda stringa è minore della prima. **Parametri:** **str1** - stringa **str2** - stringa di confronto **Sintassi:** `<function name="string.cmp" return ="<int var>" > str1, str2 </function>` **Esempio:** `<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>` **Risultato:** rval= 0

Nome della funzione	Significato
String to compare senza differenza tra minuscole o maiuscole	Due stringhe vengono confrontate tra loro dal punto di vista lessicografico senza differenza tra minuscole o maiuscole. La funzione restituisce il valore zero se le stringhe sono uguali, minore di zero se la prima stringa è minore della seconda o maggiore di zero se la seconda stringa è minore della prima. **Parametri:** **str1** - stringa **str2** - stringa di confronto **Sintassi:** `<function name="string.icmp" return ="<int var>" > str1, str2 </function>` **Esempio:** `<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>` **Risultato:** rval= 0
String left	La funzione estrae i primi caratteri nCount dalla stringa 1 e li copia nella variabile Return. **Parametri:** **str1** - stringa **nCount** - numero di caratteri **Sintassi:** `<function name="string.left" return="<result string>"> str1,  nCount </function>` **Esempio:** `<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>` **Risultato:** str2="A brown bear"

Nome della funzione	Significato
String right	La funzione estrae gli ultimi caratteri nCount dalla stringa 1 e li copia nella variabile Return. **Parametri:** **str1** - stringa **nCount** - numero di caratteri **Sintassi:** `<function name="string.right" return="<result string>"> str1, nCount </function>` **Esempio:** `<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>` **Risultato:** str2="brown dog."
String middle	La funzione estrae a partire dall'indice iFirst il numero specificato di caratteri dalla stringa 1 e li copia nella variabile Return. **Parametri:** **str1** - stringa **iFirst** - indice di partenza **nCount** - numero di caratteri **Sintassi:** `<function name="string.middle"` `return="<result string>"> str1, iFirst, nCount </function>` **Esempio:** `<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>` **Risultato:** str2="brown"

Nome della funzione	Significato
String length	La funzione fornisce il numero di caratteri di una stringa. **Parametri:** **str1** - stringa **Sintassi:** `<function name="string.length" return="<int var>"> str1 </function>` **Esempio:** `<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>` **Risultato:** length = 31
Strings to replace	La funzione sostituisce tutte le stringhe parziali trovate con la nuova stringa. **Parametri:** **string** - variabile stringa **find string** - stringa da sostituire **new string** - nuova stringa **Sintassi:** `<function name="string.replace"> string, find string, new string </function>` **Esempio:** `<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>` **Risultato:** str1 = "A brown bear hunts a big salmon!"

Nome della funzione	Significato
Strings to remove	La funzione elimina tutte le stringhe parziali trovate. **Parametri:** **string** - variabile stringa **remove string** - stringa parziale da eliminare **Sintassi:** `<function name="string.remove"> string, remove string </function>` **Esempio:** `<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>` **Risultato:** str1 = "A brown bear hunts"
Strings to insert	La funzione inserisce una stringa nell'indice specificato. **Parametri:** **string** - variabile stringa **index** - indice (in base zero) **insert string** - stringa da inserire **Sintassi:** `<function name="string.insert"> string, index, insert string` `</function>` **Esempio:** `<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>` **Risultato:** str1 = "A brown bear hunts a brown dog"

Nome della funzione	Significato
String delete	La funzione elimina il numero di caratteri specificato a partire dalla posizione indicata. **Parametri:** **string** - variabile stringa **start index** - indice di partenza (in base zero) **nCount** - numero di caratteri da eliminare **Sintassi:** `<function name="string.delete"> string, start index , nCount` `</function>` **Esempio:** `<let name="str1" type="string">A brown bear hunts. </let>` `<function name="string.delete" > str1, 2, 5` `</function>` **Risultato:** str1 = "A bear hunts"
String find	La funzione esamina la stringa trasmessa alla ricerca della prima corrispondenza con la stringa parziale. Una volta trovata la stringa parziale, la funzione fornisce l'indice sul primo carattere (che inizia con zero), altrimenti -1. **Parametri:** **string** - Variabile stringa **findstring** - Stringa da cercare **startindex** – Indice di partenza (opzionale) **Sintassi:** `<function name="string.find" return="<int val>"> str1, find string` `</function>` **Esempio:** `<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>` **Risultato:** Indice = 2 **oppure** `<function name="string.find" return="index"> str1, _T"brown", 1` `</function>`

Nome della funzione	Significato
String reverse find	La funzione esamina la stringa trasmessa alla ricerca dell'ultima corrispondenza con la stringa parziale. Una volta trovata la stringa parziale, la funzione fornisce l'indice sul primo carattere (che inizia con zero), altrimenti -1. **Parametri:** **string** - Variabile stringa **find string** - Stringa da cercare **startindex** – Indice di partenza (opzionale) **Sintassi:** `<function name="string.reversefind"` `return="<int val>"> str1, find string </function>` **Esempio:** `<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>` **Risultato:** Indice = 21 **oppure** `<function name="string.reversefind"` `return="index"> str1, _T"brown" 10 </function>` **Risultato:** Indice = 2
String GetAt	Questa funzione legge un carattere dalla posizione specificata. **Parametri:** **str** - Stringa **index** - Indice in base zero sul carattere da leggere **Valore di ritorno:** Si deve specificare il nome di una variabile di stringa in cui deve essere salvato il carattere. **Sintassi:** `<function name="string.getat" return="<result string>"><string>,` `<index></function>` **Esempio:** `<let name="resStr" type="string" />` `<function name="string.getat"` `return="resStr">_T"brown", 2</function>`

Nome della funzione	Significato
String pixel height	Calcolo dell'altezza di una stringa di caratteri in pixel. Il calcolo viene eseguito utilizzando il font corrente del controllo o del form specificato. Il nome del controllo va trasmesso come stringa. Se si immette una stringa vuota, il calcolo si riferisce al form. **Parametri:** **control name** **Stringa di caratteri** Si può immettere la stringa di caratteri direttamente come testo oppure si deve indicare una variabile di stringa. **Valore restituito:** È atteso il nome di una variabile Integer nella quale viene scritta l'altezza. **Sintassi:** `<function name="STRING.PIXELHEIGHT"` `return="<return variable>"><control name>,` `<varaible or text></function>` **Esempio:** Calcolo della larghezza riferito al font di un form `<let name="pixelsh" />` `...` `<function name="STRING.PIXELHEIGHT"` `return="pixelsh">_T"", cedit1</function>` Calcolo della larghezza riferito al font di un controllo `<function name="STRING. PIXELHEIGHT"` `return="pixelsh"> cedit1, cedit1</function>`

Nome della funzione	Significato
Larghezza pixel stringa	Calcolo della larghezza della stringa di caratteri in pixel. Il calcolo viene eseguito utilizzando il font corrente del controllo o del form specificato. Il nome del controllo va trasmesso come stringa. Se si immette una stringa vuota, il calcolo si riferisce al form. **Parametri:** **control name** **Stringa di caratteri** Si può immettere la stringa di caratteri direttamente come testo oppure si deve indicare una variabile di stringa. **Valore restituito:** Il nome di una variabile Integer nella quale viene scritta la larghezza del testo. **Sintassi:** `<function name="STRING.PIXELWIDTH"` `return="<return variable>"><control name>,` `<varaible or text></function>` **Esempio:** Calcolo della larghezza riferito al font di un form `<let name="pixels" />` `...` `<function name="STRING. PIXELWIDTH"` `return="pixels">_T"", cedit1</function>` Calcolo della larghezza riferito al font di un controllo `<function name="STRING.PIXELWIDTH"` `return="pixels"> cedit1, cedit1</function>`
String SetAt	Questa funzione scrive un carattere nella posizione specificata. L'indice deve essere minore della lunghezza massima del testo. **Parametri:** **str** - Stringa **char** - Carattere che deve essere scritto nella posizione specificata **index** - Indice in base zero sulla stringa di caratteri della variabile di destinazione **Sintassi:** `<function name="string.setat" ><destination string>,` `<character string>, <index></function>` **Esempio:** `<let name="str" type="string" >br_wn</string>` `<function name="string.setat">str, ">_T"o", 2` `</function>`

Nome della funzione	Significato
String Split	Questa funzione scompone una stringa in una serie di stringhe parziali e le copia nell'array di stringhe specificato. La separazione avviene in corrispondenza del carattere separatore indicato. Questo carattere di separazione non viene salvato nella stringa parziale. La funzione espande automaticamente l'array se il numero di caratteri separatori trovati supera la dimensione predefinita dell'array. **Parametri:** **str** - Stringa **char** - Nome su una variabile che contiene il carattere di separazione **number** - Nome su una variabile che dopo aver eseguito la funzione contiene il numero di stringhe parziali generate. **Valore di ritorno:** Si deve indicare il nome di un array di stringhe che conterrà la stringa parziale una volta eseguita la funzione. **Sintassi:** `<function name=" string.split"` `return="<result string array>"><string>, <char>,` `<number></function>` **Esempio:** `<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 funzione elimina gli spazi iniziali in una stringa. **Parametri:** **str1** - variabile stringa **Sintassi:** `<function name="string.trimleft" > str1 </function>` **Esempio:** `<let name="str1" type="string">test trim left</let>` `<function name="string.trimleft" > str1 </function>` **Risultato:** str1 = "test trim left"

Nome della funzione	Significato
String trim right	La funzione elimina gli spazi finali in una stringa. **Parametri:** **str1** - variabile stringa **Sintassi:** `<function name="string.trimright" > str1 </function>` **Esempio:** `<let name="str1" type="string"> test trim right </let>` `<function name="string.trimright" > str1` `</function>` **Risultato:** str1 = "test trim right"
Seno	La funzione calcola il seno del valore trasmesso in gradi. **Parametri:** **double** - angolo **Sintassi:** `<function name="sin" return="<double val>"> double </function>` **Esempio:** `<let name= "sin_val" type="double"></let>` `<function name="sin" return="sin_val"> 20.0` `</function>`
Coseno	La funzione calcola il coseno del valore trasmesso in gradi. **Parametri:** **double** - angolo **Sintassi:** `<function name="cos" return="<double val>"> double </function>` **Esempio:** `<let name= "cos_val" type="double"></let>` `<function name="cos" return="cos_val"> 20.0` `</function>`

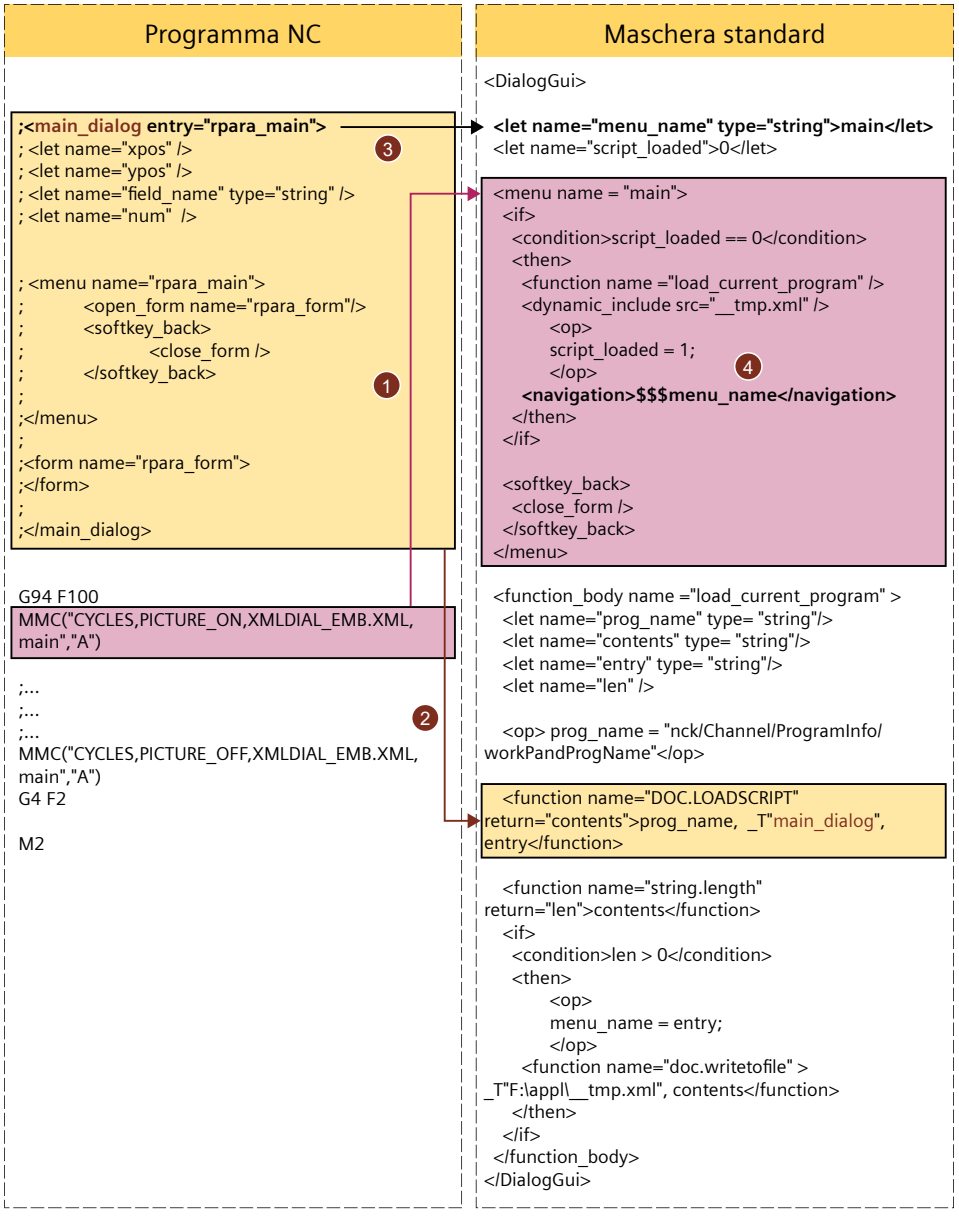
Nome della funzione	Significato
Tangente	La funzione calcola la tangente del valore trasmesso in gradi. **Parametri:** **double** - angolo **Sintassi:** `<function name="tan" return="<double val>"> double </function>` **Esempio:** `<let name= "tan_val" type="double"></let>` `<function name="tan" return="tan_val"> 20.0` `</function>`
ARCSIN	La funzione calcola l'arcoseno del valore trasmesso in gradi. **Parametri:** **double** - x nell'intervallo da -PI/2 a +PI/2 **Sintassi:** `<function name="arcsin" return="<double val>"> double </function>` **Esempio:** `<let name= "arcsin_val" type="double"></let>` `<function name="arcsin" return="arcsin_val"> 20.0` `</function>`
ARCOS	La funzione calcola l'arcocoseno del valore trasmesso in gradi. **Parametri:** **double -** x nell'intervallo da -PI/2 a +PI/2 **Sintassi:** `<function name="arcos" return="<double val>"> double </function>` **Esempio:** `<let name= "arccos_val" type="double"></let>` `<function name="arccos" return="arccos_val"> 20.0` `</function>`
ARCTAN	La funzione calcola l'arcotangente del valore trasmesso in gradi. **Parametri:** **double** - arcotangente di y/x **Sintassi:** `<function name="arctan" return="<double val>"> double </function>` **Esempio:** `<let name= "arctan_val" type="double"></let>` `<function name="arctan" return="arctan_val"> 20.0` `</function>`

Nome della funzione	Significato
Elaborazione file	
Lettura di di un file	La funzione legge il contenuto del file specificato in una variabile String. In opzione è possibile specificare, come secondo parametro, il numero dei caratteri da leggere. **Attributo:** **name -** Il nome del file deve essere indicato in lettere minuscole. L'accesso ai file di altre directory avviene tramite un'indicazione del percorso relativa, che utilizza come punto di partenza la directory appl o dvm. **return** - Nome della variabile locale **Parametri:** **progname** - Nome file **number of characters** - Numero dei caratteri da leggere in byte (opzionale) **Sintassi:** `<function name="doc.readfromfile"` `return="<string var>"> progname, number of characters` `</function>` **Esempio:** `<let name = "my_var" type="string" ></let>` File system NC `<function name="doc.readfromfile"` `return="my_var"> _T"n:\mpf\test.mpf"` `</function>` CF-Card `<function name="doc.readfromfile"` `return="my_var"> _T"f:\appl\test.mpf"` `</function>` oppure `<function name="doc.readfromfile"` `return="my_var"> _T".\test.mpf"` `</function>`

Nome della funzione	Significato
Scrittura di un file	La funzione scrive il contenuto di una variabile String nel file specificato. Il nome del file deve essere indicato in lettere minuscole. L'accesso ai file di altre directory avviene tramite un'indicazione del percorso relativa, che utilizza come punto di partenza la directory appl o dvm. **Parametri:** **progname** - Nome file **str1** - stringa **Sintassi:** `<function name="doc.writetofile" > progname, str1 </function>` **Esempio:** `<let name = "my_var" type="string" > file content </let>` File system NC `<function name="doc.writetofile">_T"n:\mpf\test.mpf", my_var </function>` CF-Card `<function name="doc.writetofile">_T"f:\appl\test.mpf", my_var </function>` oppure `<function name="doc.writetofile">_T".\test.mpf", my_var </function>`

Nome della funzione	Significato
Eliminazione di un file	La funzione elimina il file specificato dalla directory. Il nome del file deve essere indicato in lettere minuscole. L'accesso ai file di altre directory avviene tramite un'indicazione del percorso relativa, che utilizza come punto di partenza la directory appl o dvm. **Parametri:** **progname** - Nome file **Sintassi:** `<function name="doc.remove" > progname </function>` **Esempio:** File system NC `<function name="doc.remove">_T"n:\mpf\test.mpf"` `</function>` CF-Card `<function name="doc.remove">_T"f:\appl\test.mpf"` `</function>` oppure `<function name="doc.remove">_T".\test.mpf"` `</function>`

Nome della funzione	Significato
Estrazione di parti di script	La funzione copia nella variabile locale specificata una descrizione di finestra di dialogo integrata in un programma pezzo. Come parametri di richiamo è necessario specificare il nome del programma, il nome della finestra di dialogo e una variabile per la memorizzazione del nome del menu principale. Se il nome descrittivo della finestra di dialogo è stato trovato nel programma pezzo, la variabile Return contiene questa descrizione. Se il contenuto della variabile viene salvato in un file, è possibile eseguire lo script con un richiamo indiretto. Il sistema mette a disposizione uno script che estrae dal programma pezzo attivo la descrizione della finestra di dialogo e attiva la finestra di dialogo stessa. Questo script può essere richiamato in un comando MMC per attivare la maschera relativa al programma pezzo. **Sintassi:** `<function name="doc.loadscript"` `return="<name of script variable>">progname, _T"dialog part name",` `main menu </function>` **Attributo**: **return** - Variabile in cui viene archiviato lo script estratto **Parametri:** **progname** - Percorso completo del programma (è possibile trasferire il nome percorso in notazione DOS della funzione). **main menu** - Il nome di menu trovato viene copiato in questa variabile **dialog part name** - Nome del tag in cui è inserita la descrizione della finestra di dialogo **Esempio:** `<function name="doc.loadscript" return="contents">prog_name,` `_T"main_dialog", entry</function>`

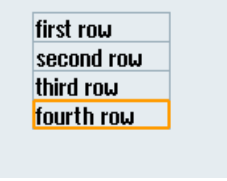
Nome della funzione	**Significato**
Estrazione di parti di script Continuazione	**Diagramma di esempio:** 

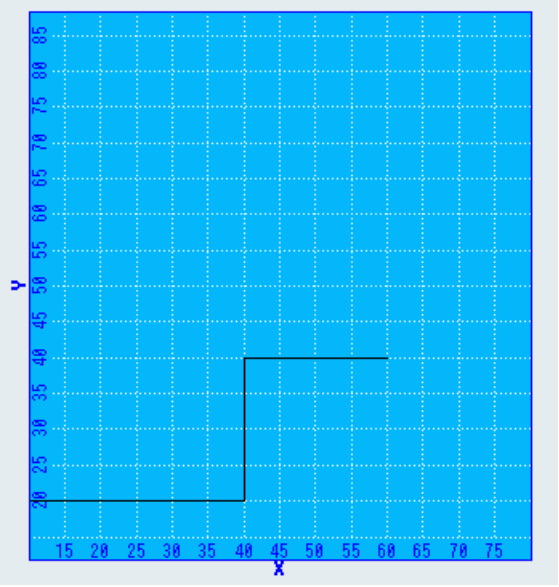
Nome della funzione	Significato
Exist	Se il file esiste, la funzione fornisce il valore 1. Il nome del file deve essere indicato in lettere minuscole. L'accesso ai file di altre directory avviene tramite un'indicazione del percorso relativa, che utilizza come punto di partenza la directory appl o dvm. **Parametri:** **progname** - Nome file **Sintassi:** `<function name="doc.exist" return="<int_var>" > progname </function>` **Esempio:** `<let name ="exist">0</let>` File system NC `<function name="doc.exist" return="exist">_T"n:\mpf\test.mpf" </function>` CF-Card `<function name="doc.exist" return="exist">_T"f:\appl\test.mpf" </function>` oppure `<function name="doc.exist" return="exist">_T".\test.mpf" </function>`
Selezione programma NC	La funzione seleziona il programma specificato per l'elaborazione. Il programma deve essere memorizzato nel file system NC. **Parametri:** **progname** - Nome file **Sintassi:** `<function name="ncfunc.select"> progname </function>` **Esempio:** File system NC `<function name="ncfunc.select"> _T"n:\mpf\test.mpf" </function>`

Nome della funzione	Significato
Impostazione di un bit singolo	La funzione consente di manipolare bit singoli delle variabili specificate. I bit possono essere impostati o resettati. **Sintassi:** `<function name="ncfunc.bitset" refvar="address" value="set/reset" > bit0, bit1, … bit9 </function>` **Attributi:** **refvar** - specifica il nome della variabile nella quale deve essere scritta la combinazione di bit **value** – campo di valore del bit 0 e 1 **Valori:** Come valori della funzione devono essere trasmessi numeri di bit che iniziano con 0. Al massimo possono essere modificati 10 bit per ogni richiamo. **Esempio:** `<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>`
Eliminazione controllo	La funzione elimina il controllo specificato. **Sintassi:** `<function name="control.delete"> control name </function>` **Attributo:** **name** – Nome della funzione **Valore:** **control name** – Nome del controllo **Esempio:** `<function name="control.delete"> _T"my_editfield" </function>`

Nome della funzione	Significato
Add Item	La funzione inserisce un nuovo elemento alla fine della lista. Nota: La funzione è disponibile solo per i tipi di controllo "listbox" e "graphicbox". **Sintassi:** `<function name="control.additem"> control name, item </function>` **Attributo:** **name** – Nome della funzione **Valori:** **control name** – Nome del controllo **item** - Espressione da inserire **itemdata** - Numero intero definito dall'utente **Esempio:** `<let name ="itemdata">1</let>` … … … `<op> item_string = _T"text1" </op>` `<function name="control.additem">_T"listbox1", item_string, itemdata </function>`

Nome della funzione	Significato
Insert Item	La funzione inserisce un nuovo elemento nella posizione specificata. Nota: La funzione è disponibile solo per il tipo di controllo "listbox". **Sintassi:** `<function name="control.insertitem"> control name, index, item, itemdata </function>` **Attributo:** **name** – Nome della funzione **Valori:** **control name** - Nome del controllo **index** - Posizione che inizia con zero **item** - Espressione da inserire **itemdata** - Numero intero definito dall'utente **Esempio:** `<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 funzione elimina un elemento nella posizione specificata. Nota: La funzione è disponibile solo per il tipo di controllo "listbox". **Sintassi:** `<function name="control.deleteitem"> control name, index </function>` **Attributo:** **name** - Nome della funzione **Valori:** **control name** - Nome del controllo **index**- Indice che inizia con 0 **Esempio:** `<function name="control.deleteitem">_T"listbox1", 1</function>`

Nome della funzione	Significato
Load Item	La funzione inserisce una lista di espressioni nel controllo. La funzione è disponibile solo per i tipi di controllo "listbox" e "graphicbox". **Sintassi:** `<function name="control.loaditem"> control name, list </function>` **Attributo:** **name** – Nome della funzione **Valori:** **control name** – Nome del controllo **list**- variabile String La stringa contenuta nelle variabili deve corrispondere alla struttura descritta di seguito. **Struttura della lista:** La lista contiene un numero di espressioni che devono essere separate tra di loro dal carattere \\n. **Esempio:** Nell'esempio, alla Listbox viene assegnato il contenuto tramite la funzione control.loaditem. Il carattere di controllo \\n separa l'una dall'altra le espressioni appartenenti alla riga. `<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>` 

Nome della funzione	Significato
Load Item Continuazione	**Esempio:** Nell'esempio, alla Graphicbox viene assegnato il contenuto tramite la funzione control.loaditem. Il carattere di controllo \\n separa l'uno dall'altro gli elementi grafici. `<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 funzione elimina il contenuto del controllo Listbox o Graphicbox. **Sintassi:** `<function name="control.empty"> control name, </function>` **Attributo:** **name** – Nome della funzione **Valori:** **control name** – Nome del controllo **Esempio:** `<function name="control.empty">_T"listbox1" </function>`

Nome della funzione	Significato
Get focus	La funzione fornisce il nome del controllo che è attivo per l'immissione. **Sintassi:** `<function name="control.getfocus" return="focus_name" />` **Attributi:** **name** – Nome della funzione **return** – Deve essere specificata una variabile String nella quale viene copiato il nome del controllo. **Esempio:** `<let name>="focus_field" type="string"></let>` `<function name="control.getfocus" return="focus_field"/>`
Set focus	La funzione rende attivo per l'immissione il controllo specificato. Il nome del controllo deve essere trasferito alla funzione come espressione di testo. **Sintassi:** `<function name="control.setfocus"> control name` `</function>` **Attributo:** **name** - Nome della funzione **Valore:** **control name** - Nome del controllo **Esempio:** `<function name="control.setfocus" ">_T"listbox1"` `</function>`
Get cursor selection	La funzione fornisce l'indice del cursore per una Listbox. Il nome del controllo deve essere trasferito alla funzione come espressione di testo. **Sintassi:** `<function name="control.getcurssel" return="var">control name` `</function>` **Esempio:** `<let name>="index"></let>` `<function name="control.getcurssel" return="index">_T"listbox1"` `</function>`

Nome della funzione	Significato
Set cursor selection	La funzione imposta il cursore sulla riga corrispondente in caso di Listbox. Il nome del controllo deve essere trasferito alla funzione come espressione di testo. **Sintassi:** `<function name="control.setcurssel" > control name, index </function>` **Esempio:** `<let name>="index">2</let>` `<function name="control.setcurssel" ">_T"listbox1",index </function>`
Get Item	La funzione copia il contenuto della riga selezionata nella variabile specificata in caso di Listbox. Come variabile di riferimento va indicata una variabile String. Il nome del controllo deve essere trasferito alla funzione come espressione di testo. **Sintassi:** `<function name="control.getitem" return="var"> control name, index </function>` **Esempio:** `<let name>="index">2</let>` `<let name>="item" type="string"></let>` `<function name="control.getitem" return="item" ">_T"listbox1",index </function>`
Get Item Data	La funzione copia il valore di elemento assegnato dall'utente nella variabile specificata in caso di Listbox. In caso di controllo Edit la funzione copia il valore assegnato dall'utente (item_data) nella variabile specificata. Come variabile di riferimento va indicata una variabile Integer. Il nome del controllo deve essere trasferito alla funzione come espressione di testo. **Sintassi:** `<function name="control.getitemdata" return="var"> control name, index </function>` **Esempio:** `<let name>="index">2</let>` `<let name>="itemdata"></let>` `<function name="control.getitemdata" return="itemdata" ">_T"listbox1",index</function>`

Nome della funzione	Significato
Image Box Set	Questa funzione serve alla gestione dell'area visibile dei controlli **Imagebox** e **Graphicbox**. Come parametri di richiamo è necessario specificare il nome del controllo (Control), il comando di controllo e i relativi valori. **Sintassi:** `<function name="control.imageboxset">control name, command, command parameter</function>` **Parametri di richiamo:** • **x_offs** La funzione sposta la sezione d'immagine sulla posizione X specificata. Come ulteriore parametro è necessario specificare la coordinata dello spigolo sinistro. • **y_offs** La funzione sposta la sezione d'immagine sulla posizione Y specificata. Come ulteriore parametro è necessario specificare la coordinata dello spigolo superiore. • **xy_offs** La funzione sposta la sezione d'immagine sulla posizione X/Y specificata. Come ulteriore parametro è necessario specificare le coordinate dello spigolo sinistro e superiore. • **followCursor** La sezione immagine segue la posizione del cursore impostata. • **zoomplus** L'immagine viene ingrandita di un fattore 0,12. • **zoomminus** L'immagine viene ridotta di un fattore 0,12. • **autozoom** L'immagine viene messa automaticamente in scala rispetto al campo di visualizzazione. • **Update** Il controllo viene nuovamente disegnato. • **SetBkColor** Il comando imposta il colore di sfondo specificato. Come ulteriore parametro è necessario specificare la codifica del colore. • **SetCursorRect** La sezione specificata come rettangolo viene spostata nell'area visibile. • **SetAnimationState** (vale solo per il controllo Imagebox) – **start** - Il comando avvia un'animazione. – **stop** - Il comando arresta un'animazione.

Nome della funzione	Significato
Image Box Set Continuazione	**Esempio:** `...` `...` `<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"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>` `</init>` `</form>`
Image Box Get	Questa funzione serve per richiamare le proprietà del controllo Imagebox. Come parametri di richiamo è necessario specificare il comando di controllo e i relativi valori. **Sintassi:** `<function name="control.imageboxget">control name, command,` `command parameter</function>` **Parametri di richiamo:** **GetViewSize** Restituisce le dimensioni dell'area di visualizzazione Come ulteriore parametro è necessario specificare una variabile del tipo di struttura SIZE. **Esempio:** `<let name="view_size" type="size" />` `<function name="control.imageboxget">_T"topoview", _T"`**`GetViewSize`**`",` `view_size</function>`

Nome della funzione	Significato
Bitmap Dim	La funzione ricopia la dimensione di una mappa di bit (bitmap) in una variabile del tipo di struttura SIZE. Per la definizione del tipo è necessario integrare il file struct_def.xml nel progetto. **Per ulteriori informazioni** vedere il capitolo "Creazione del file struct_def.xml (Pagina 142)". **Sintassi:** `<function name="bitmap.dim" >name, variable type` `</function>` **Parametri:** **name** - Percorso file **variable type** - Nome di una variabile del tipo SIZE **Esempio:** `<let name="bmp_size" type="size" />` `<function name="bitmap.dim" >_T"test.bmp", bmp_size` `</function>`
Screen Resolution	La funzione ricopia la risoluzione dello schermo assoluta del sistema in una variabile del tipo di struttura SIZE. Per la definizione del tipo è necessario integrare il file struct_def.xml nel progetto. **Per ulteriori informazioni** vedere il capitolo "Creazione del file struct_def.xml (Pagina 142)". **Sintassi:** `<function name="hmi.screen_resolution" >varialble type </function>`
Get HMI Resolution	La funzione ricopia la risoluzione dello schermo utilizzata da SINUMERIK Operate in una variabile del tipo di struttura SIZE. Per la definizione del tipo è necessario integrare il file struct_def.xml nel progetto. **Per ulteriori informazioni** vedere il capitolo "Creazione del file struct_def.xml (Pagina 142)". **Sintassi:** `<function name="hmi.get_hmi_resolution" >variable type </function>`
Get Caption Height	La funzione fornisce l'altezza della riga del titolo in pixel. **Sintassi:** `<function name="hmi.get_caption_height" return="<return var>" />` **Attributi:** **return** - Variabile numero intero

Nome della funzione	Significato
Get Font Families	La funzione fornisce un elenco dei font installati. Nell'elenco, i nomi dei font sono separati da un carattere LF. Come valore restituito deve essere fornito il nome di una variabile String. **Sintassi:** `<function name="HMI.GET_FONT_FAMILIES"` `return="< String-Variable>" />` **Esempio:** `<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 funzione fornisce il valore assoluto del numero indicato. **Sintassi:** `<function name="abs" return="var"> value` `</function>`
SDEG	La funzione converte in gradi il valore specificato. **Sintassi:** `<function name="sdeg" return="var"> value` `</function>`
SRAD	La funzione converte in radianti il valore specificato. **Sintassi:** `<function name="srad" return="var"> value` `</function>`
SQRT	La funzione calcola la radice del valore indicato. **Sintassi:** `<function name="sqrt" return="var"> value` `</function>`
ROUND	La funzione arrotonda il numero trasferito alla quantità specificata di cifre dopo la virgola. Se non viene impostata una quantità di cifre dopo la virgola, la funzione arrotonda tenendo conto della prima cifra dopo la virgola. **Sintassi:** `<function name="round" return="var"> value, nDecimalPlaces` `</function>`

Nome della funzione	Significato
FLOOR	La funzione fornisce il valore intero massimo possibile che è minore o uguale al valore trasferito. **Sintassi:** `<function name="floor" return="var"> value </function>`
CEIL	La funzione fornisce il valore intero minimo possibile che è maggiore o uguale al valore trasferito. **Sintassi:** `<function name="ceil" return="var"> value </function>`
LOG	La funzione calcola il logaritmo del valore indicato. **Sintassi:** `<function name="log" return="var"> value </function>`
LOG10	La funzione calcola il logaritmo decadico del valore indicato. **Sintassi:** `<function name="log10" return="var"> value </function>`
POW	La funzione calcola il valore "a^b". **Sintassi:** `<function name="pow" return="var"> a, b </function>`
MIN	La funzione confronta i valori trasferiti e restituisce il valore minore. **Sintassi:** `<function name="min" return="var"> value1, value2 </function>`
MAX	La funzione confronta i valori trasferiti e restituisce il valore maggiore. **Sintassi:** `<function name="max" return="var"> value1, value2 </function>`
RANDOM	La funzione fornisce un numero pseudo-casuale. **Sintassi:** `<function name="random" return="var" </function>`

Nome della funzione	Significato
MMC	**Funzione**: Tramite il comando MMC si possono visualizzare dal programma pezzo in SINUMERIK Operate le finestre di dialogo definite dall'utente (maschere interattive). La struttura delle finestre di dialogo si definisce con una progettazione esclusivamente testuale (file XML nella directory costruttore); il software di sistema resta invariato. Mediante modifiche nel sistema base di Operate, i parametri risultano come segue: XML → CYCLES oppure POPUPDLG XML_ON → PICTURE_ON XML_OFF → PICTURE_OFF **Sintassi:** MMC ("settore operativo, comando, file di script XML, nome del menu, riservato, riservato, tempo di visualizzazione o variabile di conferma, riservato, modalità di conferma") **Significato:** • MMC Dal programma pezzo richiamare in modo interattivo la finestra di dialogo in SINUMERIK Operate. • CYCLES o POPUPDLG Identificativo per finestre di dialogo utente che devono essere avviate dal programma pezzo. • PICTURE_ON o PICTURE_OFF Comando: Selezione o deselezione di una finestra di dialogo • File XML Nome del file di script XML • Menu Nome del tag del menu che gestisce la finestra di dialogo da visualizzare • riservato riservato per SINUMERIK Operate • riservato riservato per SINUMERIK Operate • Tempo Tempo di visualizzazione della finestra di dialogo con la modalità di conferma "N" • riservato riservato per SINUMERIK Operate • Modalità di conferma "S" sincrono, conferma tramite softkey "OK" "N" asincrono, la finestra di dialogo viene chiusa dopo il tempo stabilito

Nome della funzione	Significato
MMC Continuazione	**Esempio di richiamo sincrono:** Mediante modifiche nel sistema base di Operate, i parametri risultano come segue: XML → CYCLES oppure POPUPDLG XML_ON → PICTURE_ON XML_OFF → PICTURE_OFF Istruzione NC `MMC("POPUPDLG or CYCLES",PICTURE_ON,mmc_cmd.xml,cmd1,,,,,","S")` `File: 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>` **Esempio di richiamo asincrono (nessuna conferma prevista):** Mediante modifiche nel sistema base di Operate, i parametri risultano come segue: XML → CYCLES oppure POPUPDLG XML_ON → PICTURE_ON XML_OFF → PICTURE_OFF Istruzione NC `MMC("POPUPDLG or CYCLES",PICTURE_ON,mmc_cmd.xml,cmd1,,,10,,","N")` `File: 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>`

Nome della funzione	Significato
MMC Continuazione	**Esempio di estrazione di parti di script da un programma pezzo** Mediante modifiche nel sistema base di Operate, i parametri risultano come segue: XML → CYCLES oppure POPUPDLG XML_ON → PICTURE_ON XML_OFF → PICTURE_OFF Istruzione NC `MMC("POPUPDLG, PICTURE_ON, XMLDIAL_EMB.XML,main","S")` `File: 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>`

Nome della funzione	Significato
MMC Continuazione	Esempio di programma

```
;<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>
```

Nome della funzione	Significato
	; 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 funzione controlla in contenuto di una stringa per stabilire se può essere valutato come numero. Gli spazi vuoti e le tabulazioni vengono ignorati. Altri caratteri considerati validi sono il segno e il punto decimale. La funzione restituisce il valore 1 se le condizioni sono soddisfatte. **Parametri:** **string** - stringa di caratteri **Sintassi:** <function name="isnumeric" return="result"><string> </function>
ROOT	La funzione estrae la radice n di un numero. **Parametri:** **value** - Numero **n** - Esponente di radice **Sintassi:** <function name="ROOT" return="result"><value>, <n></function>

3.12 Comandi Multitouch

3.12.1 Funzione Multitouch

Panoramica

L'introduzione del display Multitouch comporta la necessità di adattare il comando alla funzionalità estesa del display. I softkey con posizione fissa, ad esempio, sono sostituiti oppure integrati con i pulsanti con posizionamento libero. La varietà di configurazioni possibili per i pulsanti elimina la necessità di utilizzare per la programmazione un solo controllo, il cui aspetto è dettato esclusivamente dalle specifiche di design del software operativo. I controlli a commutazione (toggle), che dispongono di due soli stati, possono ad es. essere sostituiti da interruttori che rappresentano ogni stato con un'icona e che reagiscono alle azioni di tocco (tap) e di scorrimento (flick).

Gli elementi grafici visualizzati possono essere progettati in modo da reagire all'azione di scorrimento (pan). La **imagebox** del controllo è stata ampliata in questo senso.

Nota

I grafici emessi nel tag Paint non reagiscono alle azioni delle dita.

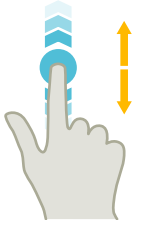
Oltre ai controlli forniti dal parser, è possibile elaborare le azioni delle dita nello script per permettere, ad esempio, di offrire nuove strategie di navigazione nelle finestre di dialogo. Le azioni delle dita possono essere utilizzate per ingrandire/ridurre i contenuti delle finestre di dialogo.

Il parser XML di Easy supporta le seguenti azioni delle dita:

Azioni delle dita

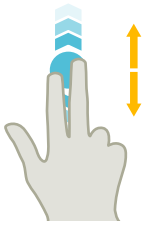
Tocco (tap)

- Selezione di finestre
- Selezione di oggetti (ad es. blocco NC)
- Attivazione di campi di immissione
  - Immissione o sovrascrittura di valori
  - Tocco ripetuto per la modifica di valori



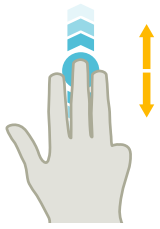
Scorrimento verticale con 1 dito (flick)

- Scorrimento nelle liste (ad es. programmi, utensili, punti zero)
- Scorrimento nei file (ad es. programma NC)



Scorrimento verticale con 2 dita (flick)

- Scorrimento laterale nelle liste (ad es. spostamento origine)
- Scorrimento laterale nei file (ad es. programmi NC)



Scorrimento verticale con 3 dita (flick)

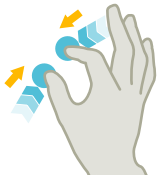
- Scorrimento all'inizio o alla fine di liste
- Scorrimento all'inizio o alla fine di file

Scorrimento orizzontale con 1 dito (flick)

- Scorrimento in liste con più colonne

Ingrandimento (spread)

- Ingrandimento di contenuti grafici (ad es. simulazione, vista per la costruzione di stampi)



Riduzione (pinch)

- Riduzione di contenuti grafici (ad es. simulazione, vista per la costruzione di stampi)

Spostamento con 1 dito (pan)

- Spostamento di contenuti grafici (ad es. simulazione, vista per la costruzione di stampi)
- Spostamento di contenuti di liste

Per la gestione delle azioni delle dita vengono utilizzati il tag **gesture_event** e la variabile **$gestureinfo**, che consentono le azioni del programma per le azioni delle dita decodificate.

3.12.2 Programmazione delle azioni delle dita

Tag gesture_event

Nota

Questo tag viene eseguito soltanto se il pannello operatore supporta il comando multitouch.

Quando il software operativo riconosce un'azione delle dita, il parser fornisce le informazioni relative alle azioni nella variabile **$gestureinfo** ed esegue il tag **gesture_event**. La variabile possiede il tipo di dati **GestureInfoStruct** e contiene i seguenti attributi:

Attributo	Valore	Significato
type	3	Azione delle dita Sposta
	4	Azione delle dita Riduci
	5	Azione delle dita Tocco
flag	1	Fattore di scala modificato
	2	Angolo di rotazione modificato
state	1	Avviato
	2	Aggiornato
	3	Terminato
item_data		Identificazione del controllo attivo
	-1	L'azione non è assegnata a un controllo
	!= -1	L'azione è stata eseguita in un controllo a cui è stato assegnato questo valore al momento della creazione
point		Posizione dell'azione
	point.x	Posizione X dell'azione
	point.y	Posizione Y dell'azione
delta		Differenza tra l'inizio dell'azione e l'evento corrente
	<Differenza di scala>	Con fattore di scala modificato
	<Differenza angolare>	Con angolo di rotazione modificato

Gli attributi sono predefiniti nel file **struct_def.xml**.

Per ulteriori informazioni vedere il capitolo "Creazione del file struct_def.xml (Pagina 142)".

3.12.3 Gestione delle azioni delle dita per i grafici

Variabile Control della imagebox

Per il controllo delle azioni degli elementi grafici esistono le seguenti tre estensioni per la variabile Control **Imagebox:**

Attributo	Significato/comportamento
rotationangle	Il valore dell'attributo indica l'angolo con il quale deve essere rappresentata la grafica.
setrotationmode	Il valore dell'attributo **true** consente l'elaborazione dell'azione Pinch
setzoommode	Il valore dell'attributo **true** permette l'ingrandimento/la riduzione e lo spostamento di un elemento grafico nell'area di visualizzazione. Per impostazione predefinita questo attributo è impostato a **false**.

Sintassi

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

Esempio di rotazione del bitmap

Rotazione del bitmap di 30,5 gradi in senso orario:

```
<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>
```

Esempio di ingrandimento del bitmap

Per consentire l'ingrandimento del 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>
```

Funzione di controllo imageboxset

È possibile impostare le proprietà tramite la funzione **control.imageboxset** .

Sono disponibili i seguenti comandi:

Comando	**Significato/comportamento**
SetRotationAngle	La grafica viene ruotata dell'angolo indicato in **lparam1**.
SetRotationMode	Il valore di **lparam1** definisce la valutazione dell'azione Pinch rispetto alla rotazione. Valore uguale a 1 - l'azione viene elaborata dal controllo
SetZoomMode	Se il valore di **lparam1** è uguale a 1, avviene la valutazione dell'azione Pinch per ingrandire/ridurre o spostare l'elemento grafico.

3.12.4 Elaborazione delle azioni delle dita

Tag message

Se il fattore di scala è superiore a 1.0, l'immagine nell'area di rappresentazione viene spostata. Con un fattore di 1.0, questa azione delle dita può essere utilizzata ad es. per sfogliare un elenco di immagini. In questo caso, il parser invia una notifica al form che deve essere valutato nel tag **message**.

Il parametro Message **$message_par1** contiene il valore Item_Data del controllo. Il parametro Message **$message_par2** segnala la direzione del movimento dell'azione delle dita.

$message_par2 può assumere i seguenti valori:

- -1 rotazione verso sinistra
- 1 rotazione verso destra
- -2 rotazione verso l'alto
- 2 rotazione verso il basso

Esempio

```
…
…
<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 rotazione verso sinistra
1 rotazione verso destra
-2 rotazione verso l'alto
2 rotazione verso il basso
 -->
<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 Progettazione personalizzata dei pulsanti

I pulsanti possono essere integrati in un form come pulsanti di azione o pulsanti di selezione.

3.13.1 Push-Button

Tag property

Il push-button è un elemento di comando che può essere impiegato come tasto o come interruttore (tasto con autoritenuta). Mediante le definizioni degli attributi il pulsante può essere progettato nello stile "Softkey Look & Feel" o nello stile specifico dell'utente. I rispettivi attributi vanno definiti con il tag **property**.

La modifica del colore di primo piano e di sfondo è possibile mediante gli attributi **color_fg**, **color_bk**, **color_fg_pressed** e **color_bk_pressed**.

Gli stati **premuto/innestato** o **rilasciato** sono rappresentati dai valori uno o zero. Per la valutazione della modifica dello stato di un tasto è necessario indicare una funzione handler. L'indicazione della funzione handler avviene con l'attributo **function** nel tag di controllo.

Lo stato di un interruttore può essere determinato dalla lettura della variabile Control o di una variabile di riferimento assegnata.

Nel comando Touch, l'effetto delle azioni delle dita "premuto" e "rilasciato" è visibile solo al termine dell'azione "rilasciato".

Sintassi

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

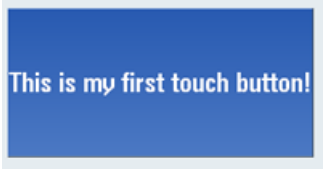
Oppure, con la funzione Handler

```
<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>
```

Oppure

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

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

disattivato



premuto, innestato

Esempio

```
<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 Funzioni del push-button

3.13.2.1 Sub-tag per il push-button

Tag caption

Con il tag **caption** il programmatore definisce il testo del pulsante da visualizzare per lo stato non premuto. Per impostazione predefinita, la rappresentazione del testo è centrata. L'allineamento del testo può essere modificato con l'attributo **alignment**. Si possono specificare i seguenti valori degli attributi:

Valore attributo	Allineamento
left	a sinistra
right	a destra
top	in alto
bottom	in basso
center	centrato

Se per lo stato premuto si deve visualizzare un altro testo, si deve programmare una seconda istruzione Caption con l'attributo **pressed** e il valore **true**.

Sintassi

Rappresentazione centrata

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

Rappresentazione allineata a sinistra

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

Rappresentazione del tasto premuto

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

</control>
```

3.13.2.2 Proprietà del Push-Button

Con il tag **property** vengono assegnate al controllo delle proprietà aggiuntive.

Definizione delle proprietà dei tasti

L'attributo **checkable** consente di utilizzare il pulsante come interruttore. A questo il valore dell'attributo va impostato a **true**.

Sintassi

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

Blocco del tasto

L'attributo **disabled** controlla l'operabilità del pulsante. Se il valore è **true**, le operazioni di comando non vengono elaborate.

Le modifiche dello stato richiamate da una variabile di riferimento assegnata provocano l'aggiornamento del pulsante.

Sintassi

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

Esempio

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

Assegnazione delle icone

Il design predefinito si può personalizzare specificando icone o colori dei pulsanti propri. Per gli stati "non premuto", "premuto" e "bloccato" è possibile assegnare al pulsante un'icona specifica. Se non viene effettuata l'assegnazione per gli stati "premuto" o "bloccato", il controllo mostra l'icona per lo stato "non premuto".

Altri attributi gestiscono la disposizione, la scalatura e la trasparenza della rispettiva icona.

Attributo	Significato/comportamento
Picture	Icona in primo piano Nome dell'icona per lo stato "non premuto"
PicturePressed	Icona in primo piano Nome dell'icona per lo stato "premuto"
PictureDisabled	Icona in primo piano Nome dell'icona per lo stato "bloccato"

Attributo	Significato/comportamento
BackgroundPicture	Icona in secondo piano Nome dell'icona per lo stato "non premuto"
BackgroundPicturePressed	Icona in secondo piano Nome dell'icona per lo stato "premuto"
BackgroundPictureDisabled	Icona in secondo piano Nome dell'icona per lo stato "bloccato"

Attributo alignment

Nel tag è possibile specificare ulteriori attributi, i cui valori di riferiscono all'**alignment** delle assegnazioni delle icone eseguite:

Valore attributo	Allineamento
left	a sinistra
right	a destra
top	in alto
bottom	in basso
center	centrato
stretch	Icona di sfondo: Se si utilizza il valore **stretch**, il parser scala l'icona sull'angolo destro del pulsante. Per creare un profilo qualsiasi del pulsante, assegnare al colore trasparente l'attributo **transparent**.

3.13.2.3 Variabili Control per il push-button

In un momento successivo è possibile modificare le variabili del pulsante assegnando nuovi valori nelle variabili dell'attributo specificate. L'assegnazione avviene in un'istruzione di operazione, immettendo il nome del controllo seguito dal nome della variabile Control. I due nomi devono essere separati da un punto.

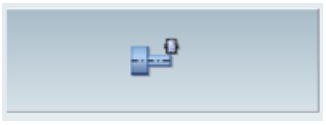
Sintassi

<Control-Name>.<Control-Variable>

Variabile Control	Significato/comportamento
backgroundpicture	Tipo di variabile: String Nome dell'icona di sfondo
backgroundpicturedisabled	Tipo di variabile: String Nome dell'icona di sfondo per lo stato bloccato
backgroundpicturepressed	Tipo di variabile: String Nome dell'icona di sfondo per lo stato premuto
picture	Tipo di variabile: String Nome dell'icona di primo piano
picturedisabled	Tipo di variabile: String Nome dell'icona di primo piano per lo stato bloccato
picturepressed	Tipo di variabile: String Nome dell'icona di primo piano per lo stato premuto
picture_textalignedtopicture	Tipo di variabile: Bool Valore: 1 = true 0 = false Il testo del pulsante viene disposto sull'icona
picturedisabled_textalignedtopicture	Tipo di variabile: Bool Valore: 1 = true 0 = false Il testo del pulsante viene disposto sull'icona "stato bloccato"
picturepressed_textalignedtopicture	Tipo di variabile: Bool Valore: 1 = true 0 = false Il testo del pulsante viene disposto sull'icona "stato premuto"

Variabile Control	Significato/comportamento
backgroundpicture_keepaspectratio	Tipo di variabile: Bool Valore: 1 = true 0 = false Il rapporto tra altezza e larghezza dell'icona di sfondo viene mantenuto o ignorato
backgroundpicturedisabled_keepaspectratio	Tipo di variabile: Bool Valore: 1 = true 0 = false Il rapporto tra altezza e larghezza dell'icona di sfondo "stato bloccato" viene mantenuto o ignorato
backgroundpicturepressed_keepaspectratio	Tipo di variabile: Bool Valore: 1 = true 0 = false Il rapporto tra altezza e larghezza dell'icona di sfondo "stato premuto" viene mantenuto o ignorato
picture_keepaspectratio	Tipo di variabile: Bool Valore: 1 = true 0 = false Il rapporto tra altezza e larghezza dell'icona viene mantenuto o ignorato
picturedisabled_keepaspectratio	Tipo di variabile: Bool Valore: 1 = true 0 = false Il rapporto tra altezza e larghezza dell'icona "stato bloccato" viene mantenuto o ignorato
picturepressed_keepaspectratio	Tipo di variabile: Bool Valore: 1 = true 0 = false Il rapporto tra altezza e larghezza dell'icona "stato premuto" viene mantenuto o ignorato
backgroundpicture_scaled	Tipo di variabile: Bool Valore: 1 = true 0 = false L'icona di sfondo viene adattata alle dimensioni del pulsante

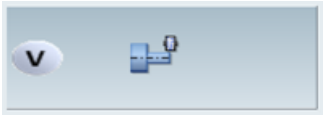
Variabile Control	Significato/comportamento
backgroundpicturedisabled_scaled	Tipo di variabile: Bool Valore: 1 = true 0 = false L'icona di sfondo "stato bloccato" viene adattata alle dimensioni del pulsante
backgroundpicturepressed_scaled	Tipo di variabile: Bool Valore: 1 = true 0 = false L'icona di sfondo "stato premuto" viene adattata alle dimensioni del pulsante
picture_scaled	Tipo di variabile: Bool Valore: 1 = true 0 = false L'icona viene adattata alle dimensioni del pulsante
picturedisabled_scaled	Tipo di variabile: Bool Valore: 1 = true 0 = false L'icona "stato bloccato" viene adattata alle dimensioni del pulsante
picturepressed_scaled	Tipo di variabile: Bool L'icona "stato premuto" viene adattata alle dimensioni del pulsante
backpicture_alignment	Tipo di variabile: String Vedere l'attributo alignment
backgroundpicturedisabled_alignment	
backgroundpicturepressed_alignment	
picture_alignment	
picturedisabled_alignment	
picturepressed_alignment	
caption	Tipo di variabile: String Testo del pulsante "stato non premuto"
captionpressed	Tipo di variabile: String Testo del pulsante "stato premuto"
caption_alignment	Tipo di variabile: String Vedere l'attributo alignment
captionpressed_alignment	
captiondisabled_alignment	
disabled	Tipo di variabile: Bool Valore: 1 = true - Push-Button bloccato 0 = false - Push-Button utilizzabile



Picture

PicturePressed

PictureDisabled



BackgroundPicture + Picture

Attributo TextAlignedToPicture

Se il valore dell'attributo è **true**, l'orientamento del testo è riferito all'icona.

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

Attributo scaled

Se il valore dell'attributo è **true**, le dimensioni della figura vengono adattate alle dimensioni del pulsante avviene in modo che per l'elemento grafico sia disponibile al massimo il 50 % dell'altezza o della larghezza del pulsante.

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

Attributo stretch (attivo solo per le icone di sfondo)

Se il valore dell'attributo è **true**, le dimensioni dell'immagine vengono adattate alle dimensioni del pulsante.

```
<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 Switch on/off

Un controllo Switch è un elemento grafico che segnala uno stato di due stati possibili mediante un'icona. Questo controllo è gestibile tramite tocco o tramite mouse.

Lo stato assunto può essere salvato in una variabile di riferimento o il cambiamento di stato può essere definito in una funzione assegnata al controllo. L'assegnazione avviene con l'attributo **function** nel tag di controllo.

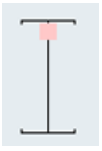
L'attributo **alignment** consente di disporre il pulsante in verticale o in orizzontale.

Questo controllo non ha un design standard. Al controllo non sono assegnate icone e gli unici elementi rappresentati sono la Bounding-Box e la posizione dell'interruttore (indicata da un punto).

Sintassi

```
<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>
```

Attributo alignment



Interruttore verticale: valore **vr**

Interruttore orizzontale: valore **hr**

3.13.4 Funzioni dello switch

3.13.4.1 Proprietà dello switch

Tag property

Con il tag **property** possono essere assegnate al controllo le seguenti posizioni dell'interruttore:

Attributo	Significato/comportamento
left_position	Alla variabile Control viene assegnato il valore dell'attributo se l'interruttore viene spostato verso sinistra.
right_position	Alla variabile Control viene assegnato il valore dell'attributo se l'interruttore viene spostato verso destra.
upper_position	Alla variabile Control viene assegnato il valore dell'attributo se l'interruttore viene spostato verso l'alto.
lower_position	Alla variabile Control viene assegnato il valore dell'attributo se l'interruttore viene spostato verso il basso.
disabled	L'attributo controlla l'operabilità dell'interruttore. Se il valore è **true**, le operazioni di comando non vengono valutate. Le modifiche dello stato richiamate da una variabile di riferimento assegnata provocano l'aggiornamento dello stato dell'interruttore.

Il design definitivo dell'interruttore va definito specificando ulteriori attributi. Questi attributi devono essere definiti insieme agli attributi **left_position**, **right_position**, **upper_position** o **lower_position**.

Per creare un profilo qualsiasi dell'interruttore, assegnare al colore trasparente l'attributo **transparent**.

Attributo	Significato/comportamento
picture	Icona sullo sfondo Nome dell'icona per lo stato "non premuto"
pictureDisabled	Icona in primo piano Nome dell'icona per lo stato "bloccato"

Attributo	Significato/comportamento
transparent	Il valore dell'attributo contiene il valore del colore trasparente. Questo valore del colore viene utilizzato per tutte le icone assegnate allo switch.
caption	Il valore dell'attributo contiene il testo relativo alla posizione dell'interruttore. Questo testo viene rappresentato sull'elemento ed è limitato dalle dimensioni dell'interruttore.

3.13.4.2 Variabili Control per lo switch

In un momento successivo è possibile modificare le proprietà dell'interruttore assegnando nuovi valori nelle variabili Control specificate. L'assegnazione avviene in un'istruzione di operazione, immettendo il nome del controllo seguito dal nome della variabile Control. I due nomi devono essere separati da un punto.

Sintassi

<Control-Name>.<Control-Variable>

Variabile Control	Significato/comportamento
Left_position	Tipo di variabile: Int Alla variabile Control viene assegnato il valore dell'attributo se l'interruttore viene spostato verso sinistra.
Right_position	Tipo di variabile: Int Alla variabile Control viene assegnato il valore dell'attributo se l'interruttore viene spostato verso destra.
Lower_position	Tipo di variabile: Int Alla variabile Control viene assegnato il valore dell'attributo se l'interruttore viene spostato verso il basso.
Upper_position	Tipo di variabile: Int Alla variabile Control viene assegnato il valore dell'attributo se l'interruttore viene spostato verso l'alto.
Picture_left_position	Tipo di variabile: String Nome dell'icona per la posizione a sinistra
Picture_right_position	Tipo di variabile: String Nome dell'icona per la posizione a destra
Picture_upper_position	Tipo di variabile: String Nome dell'icona per la posizione in alto
Picture_lower_position	Tipo di variabile: String Nome dell'icona per la posizione in basso
Picturedisabled_left_position	Tipo di variabile: String Nome dell'icona per la posizione a sinistra "stato bloccato"
Picturedisabled_right_position	Tipo di variabile: StringName dell'icona per la posizione a destra, "stato bloccato"
Picturedisabled_upper_position	Tipo di variabile: String Nome dell'icona per la posizione in alto "stato bloccato"
Picturedisabled_lower_position	Tipo di variabile: String Nome dell'icona per la posizione in basso "stato bloccato"

Variabile Control	Significato/comportamento
Caption_left_position	Tipo di variabile: String Testo nella posizione sinistra
Caption_right_position	Tipo di variabile: String Testo nella posizione destra
Caption_upper_position	Tipo di variabile: String Testo nella posizione in alto
Caption_lower_position	Tipo di variabile: String Testo nella posizione in basso
disabled	Tipo di variabile: Bool Valore: 1 = true - interruttore bloccato 0 = false - interruttore utilizzabile

Rappresentazione orizzontale

```
<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>
```

Rappresentazione verticale

```
<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>
```

Oppure, assegnazione con la funzione Handler

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

Esempio

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>
```

Assegnazione delle proprietà tramite la variabile 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 Radio-Button

I radio-button consentono di selezionare un'opzione fra più opzioni disponibili. Per ogni opzione deve essere creato un pulsante. I radio-button appartenenti a un form, un Groupbox o una Scrollarea sono correlati, per cui può essere selezionato un solo pulsante alla volta. Gli stati dei pulsanti vengono trasferiti alle variabili Control.

Sintassi

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

3.13.6 Checkbox

Le checkbox consentono di selezionare più opzioni. Per ogni opzione deve essere creato un pulsante. Lo stato della checkbox viene trasferito alla variabile Control.

Sintassi

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

3.13.7 Groupbox

Una Groupbox riunisce visivamente un gruppo di controlli in un riquadro con un titolo. Serve a raggruppare i controlli correlati dal punto di vista logico, che devono interagire soltanto all'interno del gruppo. Le coordinate dei controlli incorporati si riferiscono all'angolo superiore sinistro sotto la riga del titolo.

Tutti i controlli appartenenti al gruppo vengono integrati nel tag Control come sotto-controlli.

Quando si assegnano i nomi dei controlli integrati e il valore **Item_Data**, tenere presente che questi devono essere univoci in relazione a tutti i controlli appartenenti al form.

Il titolo della Groupbox viene definito con il tag **caption**.

Sintassi

```
<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 Scroll-Area

Una Scroll-Area viene utilizzata per visualizzare i controlli all'interno di un'area definita. Le coordinate dei controlli incorporati si riferiscono all'angolo superiore sinistro sotto la Scroll-Area. Se i controlli vengono creati al di fuori del campo visibile, è consentito spostare il campo visibile. Tutti i controlli appartenenti alla Scroll-Area devono essere incorporati come Sub-Control nel tag Control.

Quando si assegnano i nomi dei controlli integrati e il valore **Item_Data**, tenere presente che questi devono essere univoci in relazione a tutti i controlli appartenenti al form.

Sintassi

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

Comandi a sfioramento

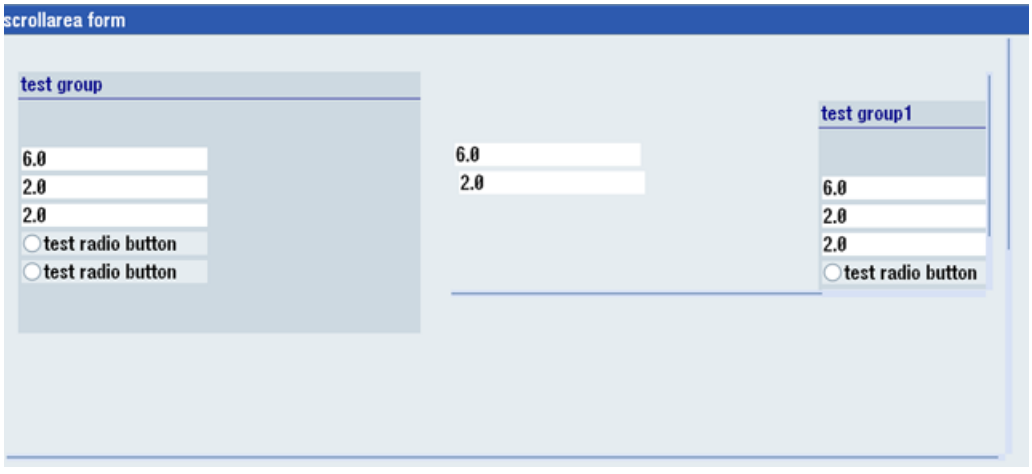
Se si attiva un controllo che al momento non è del tutto visibile, il campo visibile viene spostato finché il controllo può essere visualizzato completamente.

Azione Tab

Questa azione delle dita viene inoltrato allo script se non può essere assegnato a un controllo integrato.

Esempio

Aree di scorrimento annidate



```
<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 Applicazione Sidescreen

3.14.1 Easy XML nel sidescreen

Il linguaggio script Easy XML può anche essere utilizzato per la creazione di finestre di dialogo nell'area del sidescreen. Per visualizzare le finestre di dialogo sono disponibili i componenti del sidescreen **Page** e **Widget**. L'integrazione avviene tramite il file **slsidescreen.ini**. I componenti del sidescreen vengono avviati automaticamente con l'avvio del software operativo e terminati alla chiusura del programma. Ciò significa che il parser carica gli script assegnati una sola volta, alla prima attivazione del componente del sidescreen.

Le dimensioni dell'area del sidescreen sono definite dai parametri di layout nel file **slsidescreen_<Risoluzione schermo>.ini**. Per consentire una progettazione indipendente, la larghezza utilizzabile è pari a 276 pixel per gli script Easy XML. L'altezza utilizzabile di una finestra di dialogo di Easy XML dipende dai componenti del sidescreen utilizzati. Per una pagina sono disponibili 768 pixel. Nel caso di un widget, l'altezza è definita dalla gestione del sidescreen e può essere richiesta con la funzione **GETHMIResolution**.

Il parser degli script attende un menu per l'attivazione di un form o per la diramazione ad altri form. Il menu principale deve avere il nome **main**. Nel sidescreen non sono supportati tag dei softkey, per cui la navigazione in altri form deve avvenire attraverso gli elementi di comando del form stesso.

Il numero di pagine del sidescreen o dei widget del sidescreen non è limitato dal parser.

3.14.2 Integrazione di finestre di dialogo del sidescreen

Per l'integrazione di pagine o widget si utilizza il file **slsidescreen.ini**.

Le pagine devono essere create nella sezione **[Sidescreen]**. Per ogni pagina si deve indicare la parola chiave **PAGE** seguita dal numero di pagina progressivo e dagli attributi necessari. L'attributo **name** definisce il nome oggetto della pagina. L'attributo **implementation** indica la libreria di implementazione e il nome della classe. I due attributi devono essere separati da una virgola. Le pagine del tipo **SlSideScreenPage** possono integrare elementi del sidescreen. Questo avviene mediante le voci dette **ELEMENT**. La regola per creare una riga dell'elemento **ELEMENTxxx** corrisponde alla regola per la creazione di una pagina. Se la pagina deve contenere soltanto finestre di dialogo progettate con Easy XML, si deve indicare **slagmforms.SlEESideScreenPage** come valore dell'attributo **implementation**.

I widget possono essere aggiunti in un elemento del sidescreen nella sezione **[Element_<name>]**.

La regola per creare una riga del widget **WIDGETxxx** corrisponde alla regola per la creazione di una pagina.

Per attivare un widget del sidescreen Easy XML, si deve indicare **slagmforms.SlEESideScreenWidget** come valore dell'attributo **implementation**.

Sintassi

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

...
...

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

L'identificativo del widget viene utilizzato di norma per la formazione del nome del modulo Main.

Esempio

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

Nome del modulo Main: sidescreen_proginfluence.xml

Altre proprietà

Ulteriori proprietà possono essere assegnate alle pagine o ai widget immettendo una sezione con il nome **PAGE_<pagename>**, **ELEMENT_<elmentname>** oppure **WIDGET_<widgetname>** nel file **slsidescreen.ini**.

A una pagina, un elemento o un widget possono essere assegnate le seguenti proprietà:

Attributo	Significato/comportamento
TextId	Identificativo indipendente dalla lingua del testo
TextFile	Nome del file di testo

Attributo	Significato/comportamento
TextContext	Contesto del testo EASY_XML
Icon	Nome dell'icona

Proprietà	Significato/comportamento
maskPath	Questa proprietà definisce il nome del modulo Main da utilizzare
maskName	Questa proprietà definisce il nome del menu Main da utilizzare
focusable	Questa proprietà definisce se si può impostare lo stato di immissione attivo sull'elemento

Esempio

```
[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 Gestione della lingua e dei testi

Per la gestione dei testi indipendenti dalla lingua, ad ogni componente si può assegnare un file di testo. Il nome del file di testo è composto dal nome del componente, dalla versione della lingua e dall'estensione ".ts".

Come contesto si deve utilizzare **EASY_XML**.

Sintassi

<name>_<language>.ts

Esempio

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

Nome del file di testo: sidescreen_proginfluence**_eng.ts**

Se non viene indicato alcun file di testo, il parser ricerca nel file di testo standard **oem_xml_screens_xxx.ts** l'identificativo del testo.

3.14.4 Componenti del sidescreen

3.14.4.1 Elemento del sidescreen

Se una pagina è interconnessa con l'implementazione standard, a questa pagina possono essere assegnati degli elementi. A questo scopo si deve creare una sezione **[Page_<page_name>]** ed elencare tutti gli elementi appartenenti alla pagina.

Esempio

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

[Page_<page_name>]

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

Ogni elemento richiede una sezione **[Element_<element name>]**, nella quale sono indicati tutte le proprietà e i relativi widget.

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

3.14.4.2 Widget del sidescreen

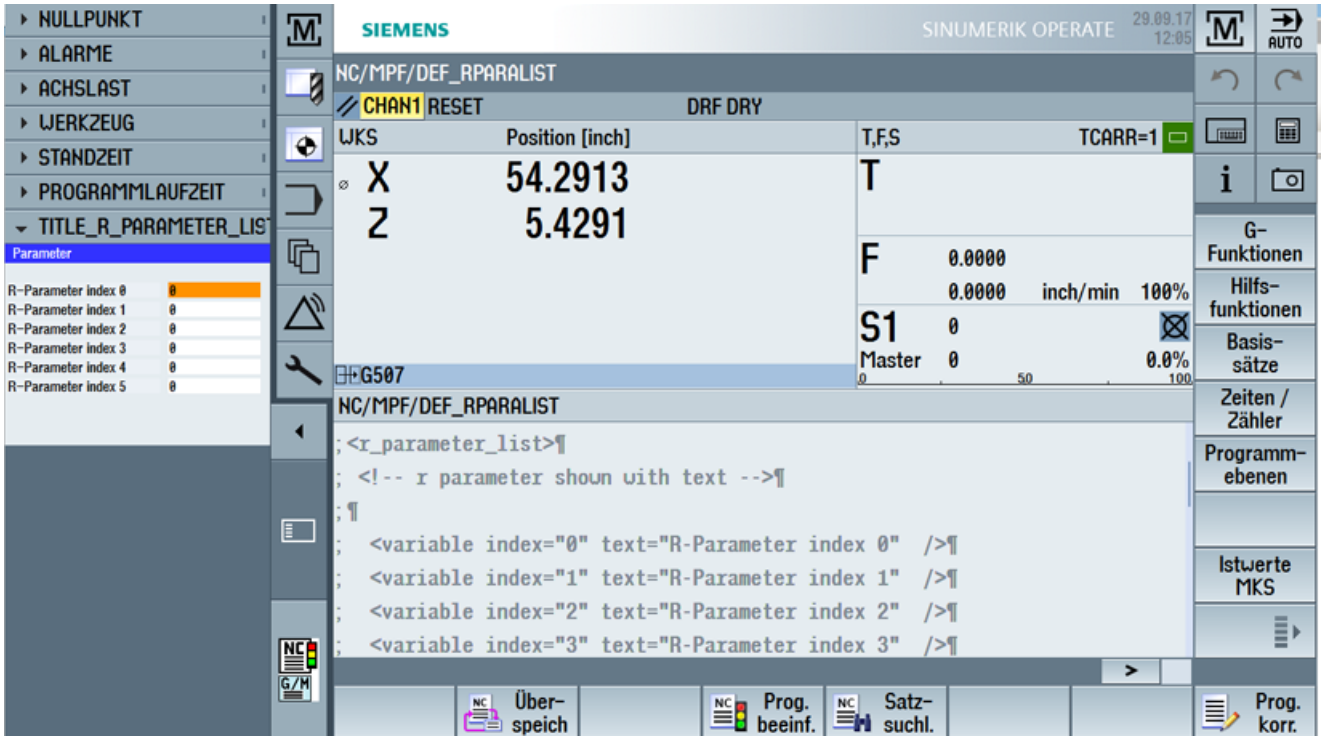
Ad un widget viene assegnata, tramite la gestione sidescreen, un'area nella quale possono essere rappresentati i form progettati. Per impostazione predefinita, un widget non può essere attivo per l'immissione, quindi non è possibile modificarne i campi. Questo comportamento è modificabile con l'attributo **focusable**. Quando si apre il widget, il parser apre il form appartenente al menu Main. All'interno del form si può accedere ad altri menu attraverso un'istruzione di navigazione.

Esempio

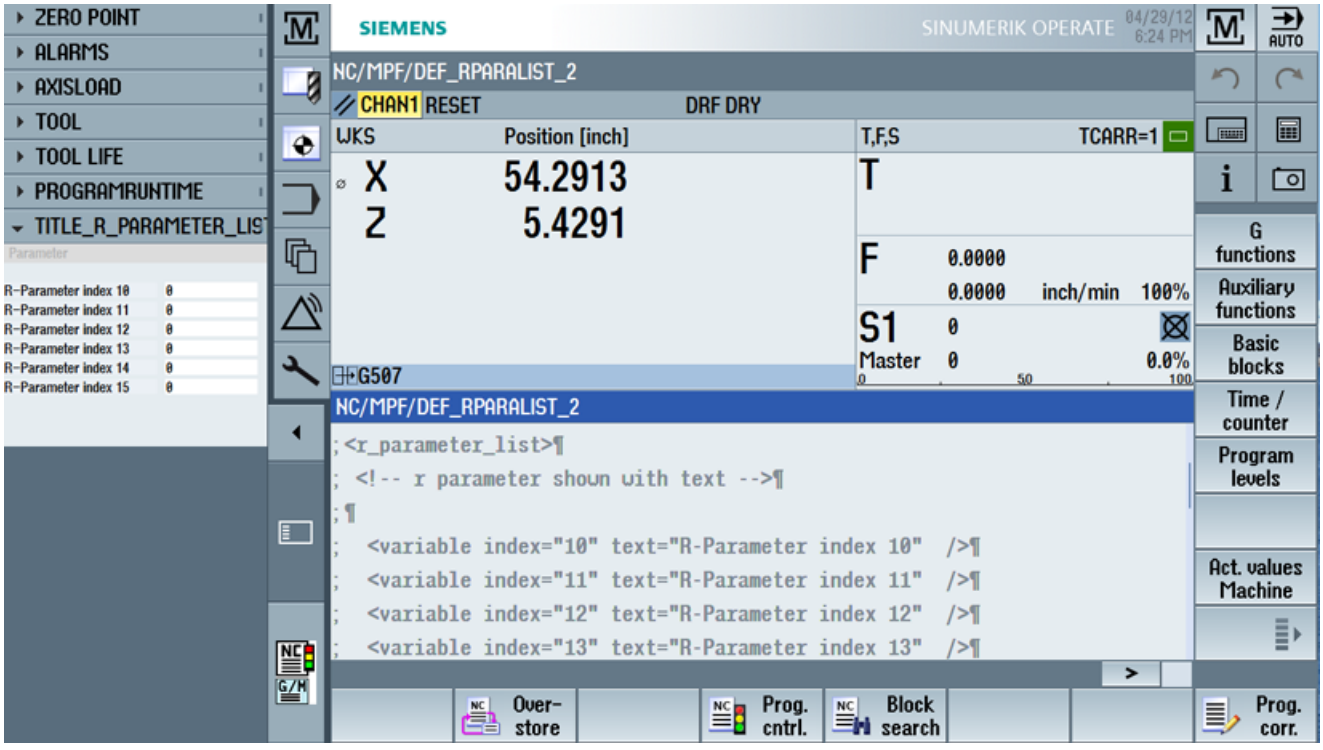
Lo script Easy XML ricerca nel programma pezzo attivo la descrizione del contenuto di dati del widget. Nell'esempio concreto, si attende la descrizione di una lista di parametri R che devono essere visualizzati. Ad ogni parametro può essere assegnato un testo descrittivo.

Esempio per toolbox

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



Se viene selezionato un altro programma pezzo, viene automaticamente elaborato il widget.


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 Pagina del sidescreen

Una pagina, che viene avviata tramite l'implementazione di **slagmforms.SIEESideScreenPage**, fornisce un form per l'intera area del sidescreen. Non contiene una riga del titolo, per cui non è possibile progettare un titolo con l'istruzione Caption. Per impostazione predefinita, una pagina non può essere attiva per l'immissione, quindi non è possibile modificarne i campi. Questo comportamento è modificabile con l'attributo **focusable**. Quando si apre la pagina, il parser apre il form appartenente al menu Main. All'interno del form si può accedere ad altri menu attraverso un'istruzione di navigazione.

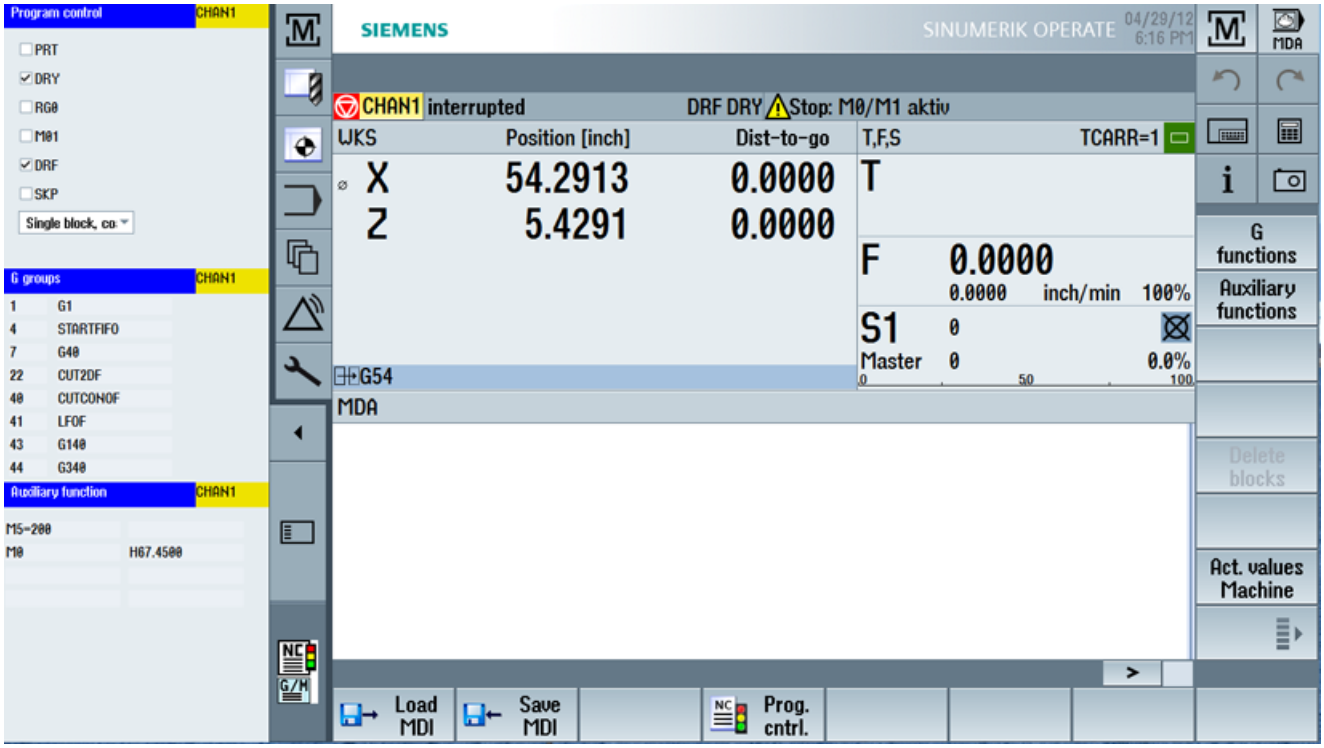
Esempio

La pagina del sidescreen viene visualizzata come pagina che contiene esclusivamente un form Easy XML.

In questo esempio, la visualizzazione delle informazioni aggiuntive avviene nell'area del sidescreen.

Esempio per 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
```

File di testo: sidescreen_proginfluence_deu.ts

3.15 Applicazione Display Manager

3.15.1 Easy XML in Display Manager

La funzione Easy XML si può incorporare nella configurazione Display Manager tramite la finestra di dialogo **SISideScreenPage**. A seconda dello spazio disponibile questa finestra di dialogo può visualizzare una o più pagine Sidescreen. Quando vi sono più pagine Sidescreen, queste vengono visualizzate sotto forma di colonne affiancate. Lo spazio (larghezza) disponibile viene distribuito uniformemente per le singole colonne. Il numero delle pagine da visualizzare e il loro contenuto vengono configurati. Ogni pagina ha un proprio file di configurazione. I nomi di questi file di configurazione vengono specificati in fase di progettazione della finestra di dialogo nel file **systemconfiguration.ini**, nel parametro della riga di comando **cmdline**.

Dopo l'identificativo del parametro **-sidescreen1** si trova il nome del file di configurazione per la prima colonna, dopo l'identificativo del parametro **-sidescreen2** il nome del file di configurazione per la seconda colonna, ecc.

Tutte le applicazioni vengono avviate automaticamente all'avvio del sistema da Display Manager e terminate alla chiusura del sistema. Questo significa che le modifiche apportate a uno script hanno effetto solo dopo aver riavviato l'HMI.

3.15.2 Integrazione della Customer Area nel menu Display Manager

La condizione per poter richiamare il parser Easy XML nella Customer Area è che vi sia un'area collegata con la finestra di dialogo Easy XML. Questo avviene di default con la definizione d'area **AREA111** nel file **systemconfiguration.ini**.

Sintassi

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

Nello stato di fornitura quest'area non è visibile. L'attivazione avviene tramite il file **slamconfig.ini** in cui il flag di visibilità **Visible** va modificato da **false** a **true**.

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

Esempi

L'integrazione della Customer Area in un menu di Display Manager avviene tramite i file INI di Display Manager che dipendono dalla risoluzione.

Nel file corrispondente si deve creare sotto lo switch **Operate buttons** una voce di menu che implichi il rimando all'area **CustomXML**:

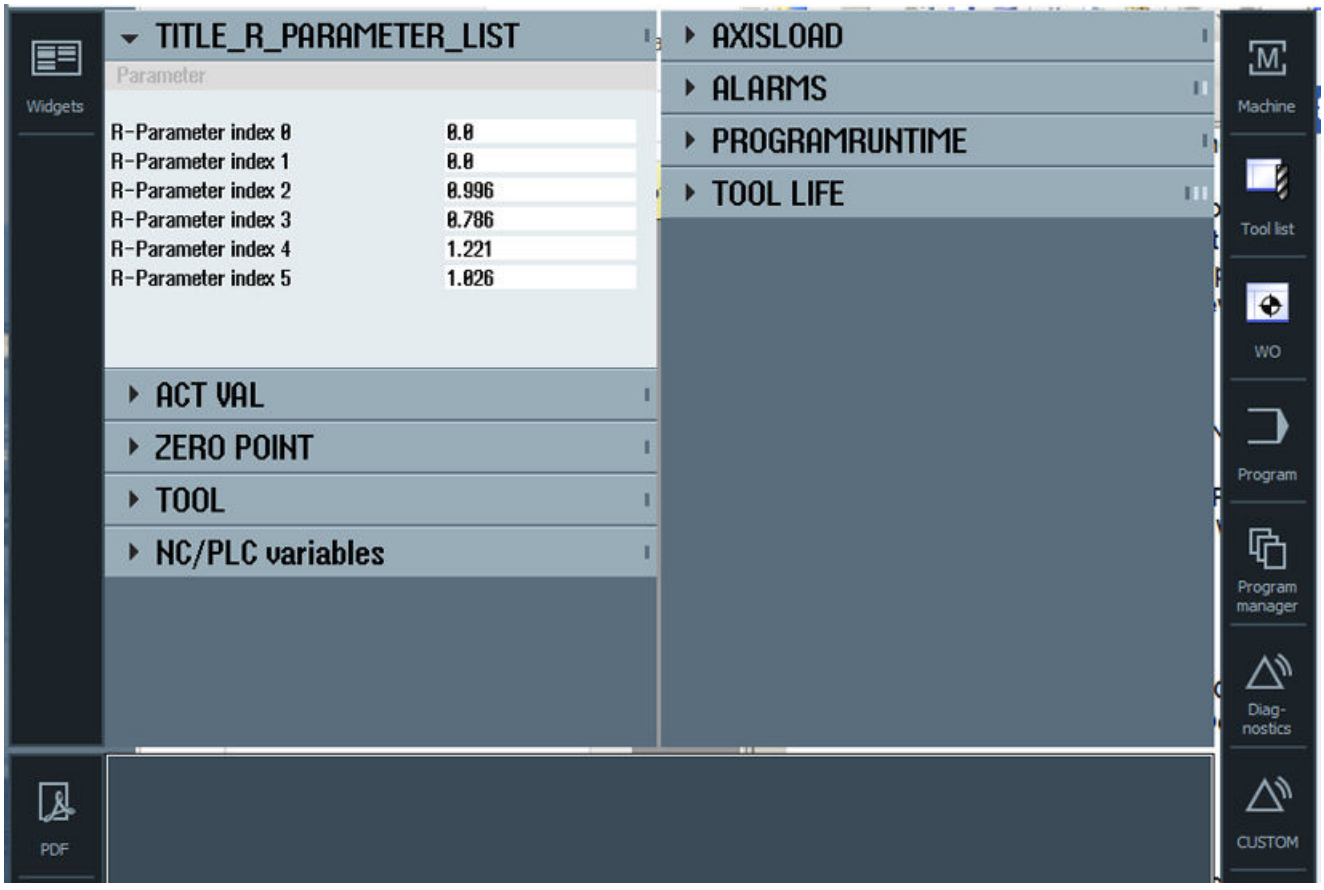
```
MENUITEM107= name:=mieasyXML, menuItemStyle:=misMenu,
onClicked:="hidePopup(); sendCmd(OPERATE, CustomXML);
showApp(defaultFrame, OPERATE)", image:=<bild.png>,
textID:= <textid>
```

Anche questo elemento di menu si assegna a un menu in 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

I widget in Display Manager si creano e gestiscono come i widget Sidescreen. La descrizione dei "widget Sidescreen" avviene tuttavia nei file di configurazione di Display Manager, ad es. nei file INI **sldmwidgets*.ini**. Il riferimento ai file avviene tramite **systemconfiguration.ini**.



Un esempio con il file **rparameter_widget.xml** si trova nel capitolo "Widget del sidescreen (Pagina 238)".

Esempio

systemconfiguration.ini

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

Il nome della finestra di dialogo viene assegnato a un frame:

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

I frame definiti in un display:

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

sldmwidgets1.ini

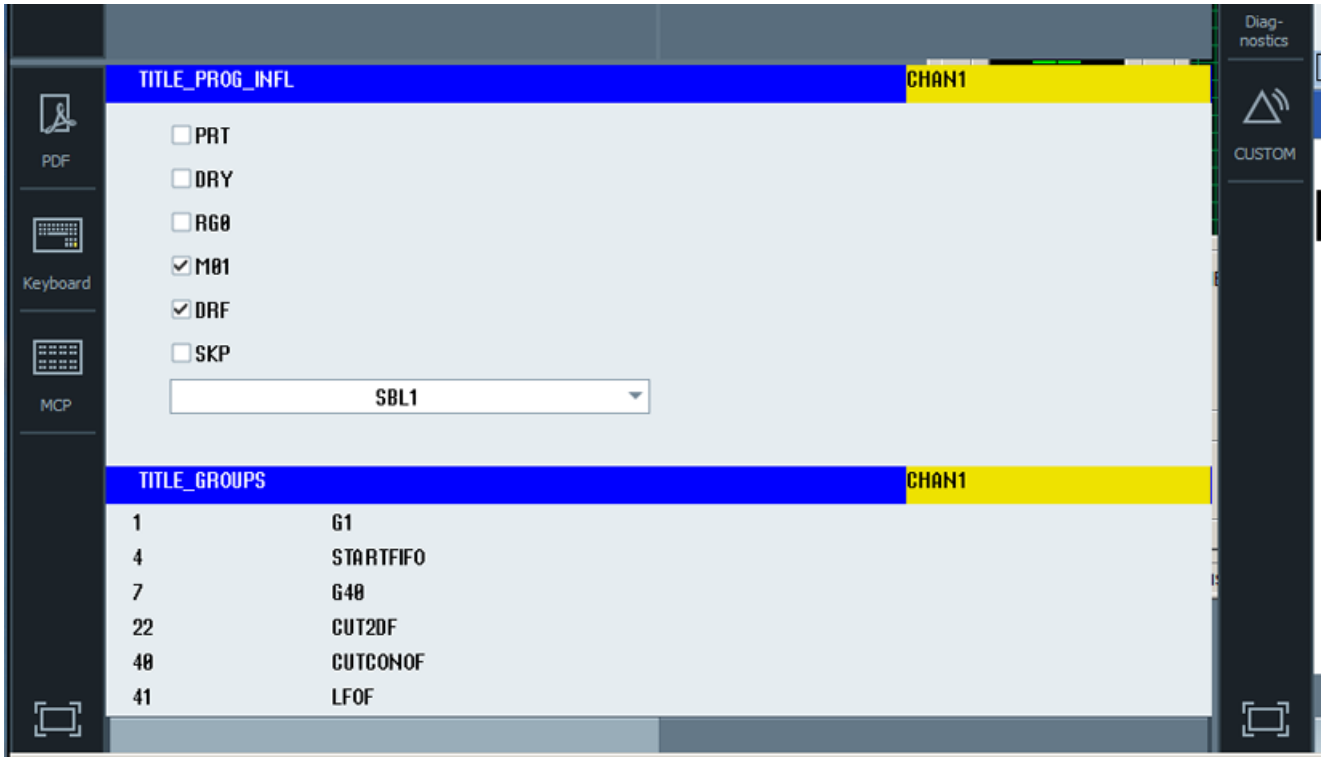
La struttura e il significato degli switch e degli elementi sono descritti nel capitolo "Applicazione Sidescreen (Pagina 234)".

```
[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

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

A **WIDGET001** viene assegnato lo script da elaborare (senza estensione file) e l'implementazione **slagmforms.SlEESideScreenWidget**.

3.15.4 Pagina Sidescreen di Display Manager

La pagine Sidescreen in Display Manager si creano e gestiscono come le pagine Sidescreen. La descrizione delle "pagine Sidescreen" avviene tuttavia nei file INI **sldmxxxpage*.ini**. Il riferimento a questi file avviene tramite il file **systemconfiguration.ini**.



Un esempio con il file **sidescreen_proginfluence.xml** si trova nel capitolo "Pagina del sidescreen (Pagina 240)".

Esempio

sldmmcppage.ini

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

Il nome della finestra di dialogo viene assegnato a un frame:

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

I frame definiti in 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
```

A **PAGE002** viene assegnato lo script da elaborare (senza estensione file) e l'implementazione **slagmforms.SlEESideScreenPage**.

3.16 Selezione di finestre di dialogo tramite hardkey PLC

Campo d'impiego

Dal PLC è possibile avviare le seguenti funzioni nel software operativo:

- Selezione del settore operativo
- Selezione di determinati scenari all'interno di settori operativi
- Esecuzione di funzioni assegnate ai softkey

Hardkey

Tutti i tasti (anche quelli del PLC) verranno di seguito denominati hardkey. È possibile definire fino a un massimo di 254 hardkey, distribuiti come segue:

Numero tasto	Impiego
Tasti 1 – 9	Tasti del frontale del pannello operatore
Tasti 10 – 49	riservati
Tasti 50 – 254 Tasti 50 – 81	Tasti del PLC: riservati per OEM
Tasto 255	Preimpostato con informazioni di controllo

Gli hardkey 1 - 9 sono preimpostati come segue:

Denominazione tasto		Azione / effetto
HK1	MACHINE	Selezione del settore operativo "Macchina", ultima finestra di dialogo
HK2	PROGRAM	Selezione del settore operativo "Programma", ultima finestra di dialogo o ultimo programma
HK3	OFFSET	Selezione del settore operativo "Parametri", ultima finestra di dialogo
HK4	PROGRAM MANAGER	Selezione settore operativo "Programma", schermata di base "Program Manager" nessuna funzione
HK5	ALARM	Selezione settore operativo "Diagnostica", finestra di dialogo "Lista allarmi"
HK6	CUSTOM	Selezione del settore operativo "Custom"

Progettazione

La progettazione avviene nel file di configurazione systemconfiguration.ini nella sezione [keyconfiguration]. Ogni riga definisce un cosiddetto "evento hardkey". Per evento hardkey si intende la n-esima attivazione di un determinato hardkey. Ad esempio la seconda e la terza attivazione di un determinato hardkey può provocare reazioni diverse.

Le voci del file di configurazione systemconfiguration.ini possono essere impostate con dati specifici dell'utente. A questo scopo sono disponibili la [cartella di sistema user]/cfg e la [cartella di sistema oem]/cfg.

Le righe per la progettazione degli eventi hardkey hanno la seguente struttura:

```
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: numero dell'hardkey, campo di valori: 1 – 254

n: numero dell'evento – corrisponde all'attivazione n dell'hardkey, campo di valori: 0 – 9

Presupposto

Il programma utente del PLC deve soddisfare il seguente requisito:

Viene sempre elaborato un solo hardkey. Per questo motivo è possibile impostare una nuova richiesta solo quando il software operativo ha confermato la richiesta precedente. Se il programma utente del PLC ricava l'hardkey da un tasto della pulsantiera di macchina, è necessario garantire una sufficiente memorizzazione intermedia dei tasti, affinché non vengano persi dei comandi impartiti da tastiera quando si digitano velocemente i tasti.

Interfaccia PLC

Nell'interfaccia PLC è previsto un settore per la selezione di un hardkey. Il settore si trova nel DB19.DBB10. Qui il PLC può impostare direttamente un valore del tasto compreso tra 50 e 254.

La conferma da parte del software operativo avviene in due fasi. Questa procedura è necessaria affinché lo stesso codice di tasto specificato due volte di seguito possa essere riconosciuto correttamente come due eventi separati dal software operativo. Nella prima fase l'informazione di controllo 255 viene scritta nel byte DB19.DBB10. Con questa attivazione virtuale del tasto è possibile riconoscere in modo univoco qualsiasi sequenza di tasti del PLC. L'informazione di controllo non ha alcun significato per il programma PLC e non deve essere modificata. Nella seconda fase avviene la conferma vera e propria al PLC e DB19.DBB10 viene cancellato. A partire da ora il programma utente del PLC può impostare un nuovo hardkey. Parallelamente viene elaborata la richiesta dell'hardkey attuale nel software operativo.

Esempio

File di configurazione:

```
; configuration of OP hardkeys (KEY1-KEY9) and
; Hardkey PLC (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
```

Gli identificatori di area e dialogo si ricavano da systemconfiguration.ini nella [cartella di sistema siemens]/cfg.

Se si utilizzano solo i dati della finestra **SlEECustomDialog** , il parser attende il file **xmldial.xml** nella directory dell'applicazione, nonché un tag del menu con il nome **main**. Altri nomi di file o nomi di menu Main possono essere notificati aggiungendo la chiave **cmdline**. Il parametro **-mainModule** definisce la descrizione XML da caricare. In questo script è previsto il menu Main dal parser. Il parametro **-entry** definisce il nome del menu Main. Se manca questo parametro, il parser utilizza il nome **main** per avviare la funzione. Il parametro **-conf** con il valore **slagmdialog.hmi** deve essere sempre incluso.

Un esempio:

```
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"
```

Interfaccia operativa

Le seguenti voci del file systemconfiguration.ini consentono l'utilizzo di Easy XML per la funzione descritta sopra.

```
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 Assegnazione dei softkey riservati alle finestre di dialogo utente

In tutti i settori operativi standard un softkey è riservato per ampliare la funzionalità esistente. Per attivare e associare un softkey OEM a un progetto utente esistono nel sistema i rispettivi file nella seguente directory:

/siemens/sinumerik/hmi/template/cfg

Questi file comportano delle descrizioni XML specifiche del settore che vengono interpretati all'attivazione del software operativo per essere integrati nella struttura di menu del settore operativo.

Per integrare una propria applicazione si deve copiare e modificare il file corrispondente al settore operativo dalla directory dei modelli alla seguente directory:

/oem/sinumerik/hmi/template/cfg

3.17.1 Definizione del testo dei softkey indipendente dalla lingua

Il tag TEXTFILE contiene il nome file del file di testo che deve essere associato al settore OEM. Il nome file va specificato senza estensione e suffisso linguistico.

Sintassi

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

Esempio

oem_xml_screens_deu.ts

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

Il testo del softkey da visualizzare è gestito dal tag **property** nel tag **softkey**. Il valore OEM deve essere sostituito dall'ID di testo desiderato. La proprietà **translationContext** fa riferimento a un'area nel file di testo cui è associato il testo. Nel caso di easyXML si deve specificare il contesto EASY_XML.

Sintassi

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

Esempio

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

3.17.2 Assegnazione dell'area Easy XML

Nel tag **softkey** si definisce inoltre l'obiettivo della navigazione. A questo scopo occorre immettere nel tag **area** l'obiettivo della navigazione **CustomXML** nell'attributo **name**. Gli attributi **dialog** e **screen** si devono cancellare, perché questi parametri sono definiti automaticamente.

Sintassi

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

Esempio

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

Quando si usa l'obiettivo di navigazione **area**, il parser attende il file xmldial.xml come modulo iniziale e il menu **main** come punto di accesso per la gestione dei menu. Per poter offrire più applicazioni in parallelo si deve impiegare la funzione **switchToDialog**. A questo scopo occorre sostituire il tag NAVIGATION con il tag FUNCTION. Come argomenti di richiamo si devono trasmettere a questa funzione il settore **CustomXML**, la finestra di dialogo **SlEECustomDialog** e il nome del modulo iniziale e del menu principale.

Per uscire dal settore operativo easyXML e tornare a quello standard si usa il tag **switchToArea**.

Sintassi

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

Progetto di esempio

sldiagnose_oem.xml

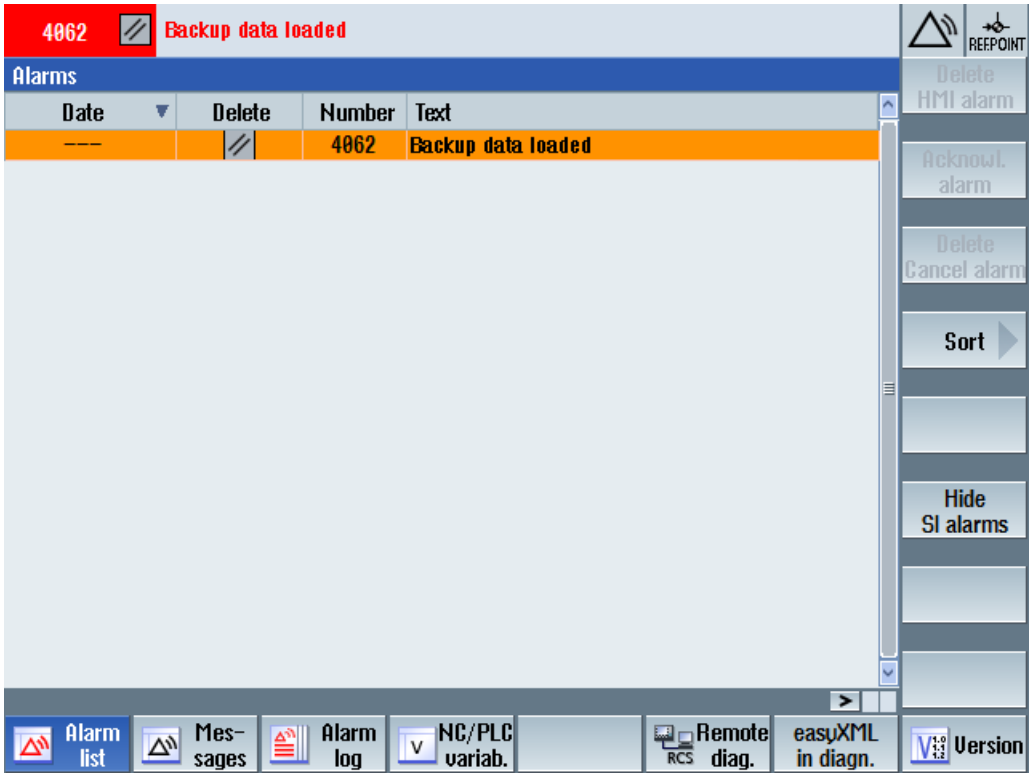
```
<!DOCTYPE DIALOG>
<DIALOG>
<TEXTFILE>oem_xml_screens</TEXTFILE>
<!-- ========================================================================
-->
<!-- OEM softkey number 7 on first horizontal main menu in area diagnosis --
>
<!-- ========================================================================
-->
<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>
```

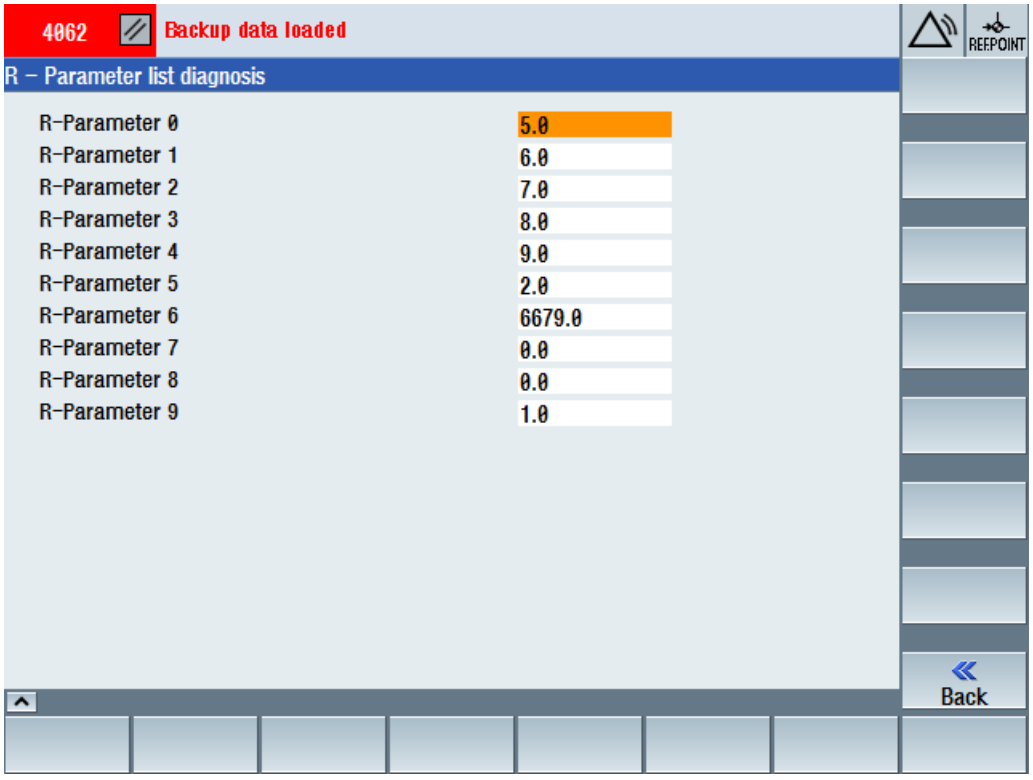
oppure

```
<!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 Descrizione dei tag

Tag softkey

Al di fuori del tag **softkey** si può impostare lo stato del softkey con l'attributo **state** impiegando l'attributo POSITION. Come valore di attributo si indica il numero del softkey per il quale deve essere impostato lo stato.

Esempio

```
<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>
...
...
```

Tag typedef

Nota

La preassegnazione degli elementi negli esempi va rimossa.

Nella definizione del tipo una variabile si dichiara con il tag **element**. Gli attributi del tag corrispondono agli attributi dell'istruzione **let**. Gli elementi si possono preassegnare quando si creano le variabili.

Esempio

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>
```

Tag let

Con l'attributo **dim** si deve specificare il numero di elementi del campo. In caso di campo bidimensionale la seconda dimensione viene specificata dopo la prima dimensione, separata da una virgola. L'accesso a un elemento del campo avviene tramite l'indice del campo, che va definito tra parentesi angolari dopo il nome della variabile. Le dimensioni dell'array si possono definire in modo diretto o tramite il contenuto di una variabile.

Esempio

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

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

oppure

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

oppure

```
<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 Creazione di testi indipendenti dalla lingua

Tutti i testi utilizzati nelle finestre di dialogo sono gestibili in modo indipendente dalla lingua impiegando degli identificatori di testo che al runtime vengono sostituiti da un testo nella lingua selezionata. Questo testo va salvato in un file di testo in formato UTF-8 insieme all'identificatore testuale per ogni lingua specificata.

Di default il file oem_xml_screens_<con estensione>.ts è associato alla funzione Easy XML come file di testo. Come contesto testuale si deve specificare l'identificatore EASY_XML.

Per contrassegnare l'identificatore testuale occorre anteporgli nello script due simboli del dollaro.

Struttura

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>
```

Esempio

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

3.19 File di testo specifici del progetto

3.19.1 Creazione di file di testo specifici del progetto

Per gestire progetti separati si possono impiegare per ciascun progetto, ogni form e ogni menu dei file di testo separati. La notifica avviene con l'attributo TEXTFILE nei tag corrispondenti.

Come valore di attributo si deve indicare il nome del file di testo senza suffisso linguistico ed estensione file. Ai testi si potrà accedere fintanto che la form o il menu non vengono chiusi. Se si carica un file di testo con il tag **DialogGui**, si potrà accedere al suo contenuto finché non si esce dal progetto. Se si utilizza più volte un contesto testuale, la ricerca degli identificatori testuali inizia nell'ultimo file caricato. Per ogni file di testo si può utilizzare un contesto testuale.

Esempio

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>
```

Attivazione

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

oppure

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

oppure

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

Finestra di dialogo di esempio

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 di dialogo

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 Indicazione del contesto testuale

In un file di testo è possibile raggruppare i testi in base a nomi di settori autodefiniti. All'interno del tag **context** l'identificativo del settore è dato dal tag **name**.

Sintassi

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

Esempio

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

Se si lavora con un proprio contesto si deve indicare con l'attributo TEXTCONTEXT il nome di quest'ultimo insieme al nome del file di testo per un progetto, una form o un menu. Ai testi archiviati nell'ambito del contesto si può accedere finché non viene impostato un nuovo contesto. Questo significa che un contesto definito nel progetto viene rimpiazzato da un contesto impostato in una forma o in un menu.

Progetto di esempio

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 Indicazione della lista dei file di testo

Se si vogliono utilizzare più file di testo per un progetto, si possono trasmettere al parser i nomi dei file di testo necessari in una lista tramite l'attributo **textfilelist**.

La notazione dei nomi dei file di testo avviene riga per riga e deve terminare con un ritorno a capo. Come nome del file di testo è previsto il nome del file senza estensione linguistica né estensione file.

Esempio

Il file `form2_screens_deu.ts` deve essere specificato nella lista con `form2_screens`.

textfilelist.ini

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

Attivazione

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

3.20 Ripristino della sessione

Con la chiusura di un progetto Easy XML si abilitano tutte le variabili locali e si scaricano le istruzioni di script. Se si desidera che i dati delle variabili vengano mantenuti anche dopo lo spegnimento del controllore, occorre dotare queste variabili dell'attributo **permanent**. I dati che devono essere mantenuti fino allo spegnimento del controllore devono essere contrassegnati con l'attributo **poweroff** .

Se si riseleziona l'applicazione Easy XML, nel menu Main è possibile impostare quale menu o quale form deve essere attivato dopo aver selezionato nuovamente l'applicazione. Il programmatore è tenuto a garantire che tutti i dati necessari siano disponibili.

Se si imposta l'attributo **restoresession** nel tag **DialogGUI**, il parser organizza la riselezione dell'ultimo menu aperto e dell'ultimo form aperto. Il programmatore è tenuto a fornire i dati necessari.

Creazione di finestre di dialogo di messa in servizio 4

4.1 Prospetto delle funzioni

Scopo

I dispositivi aggiuntivi possono essere messi in servizio, attivati, disattivati o testati in modo semplice con la funzione "Easy Extend". I dispositivi disponibili e i relativi stati sono visualizzati sul controllore in una lista. Il sistema può gestire al massimo 64 dispositivi.

L'attivazione o la disattivazione di un dispositivo avviene tramite softkey.

La funzione "Easy Extend" è disponibile nel settore operativo "Parametri".

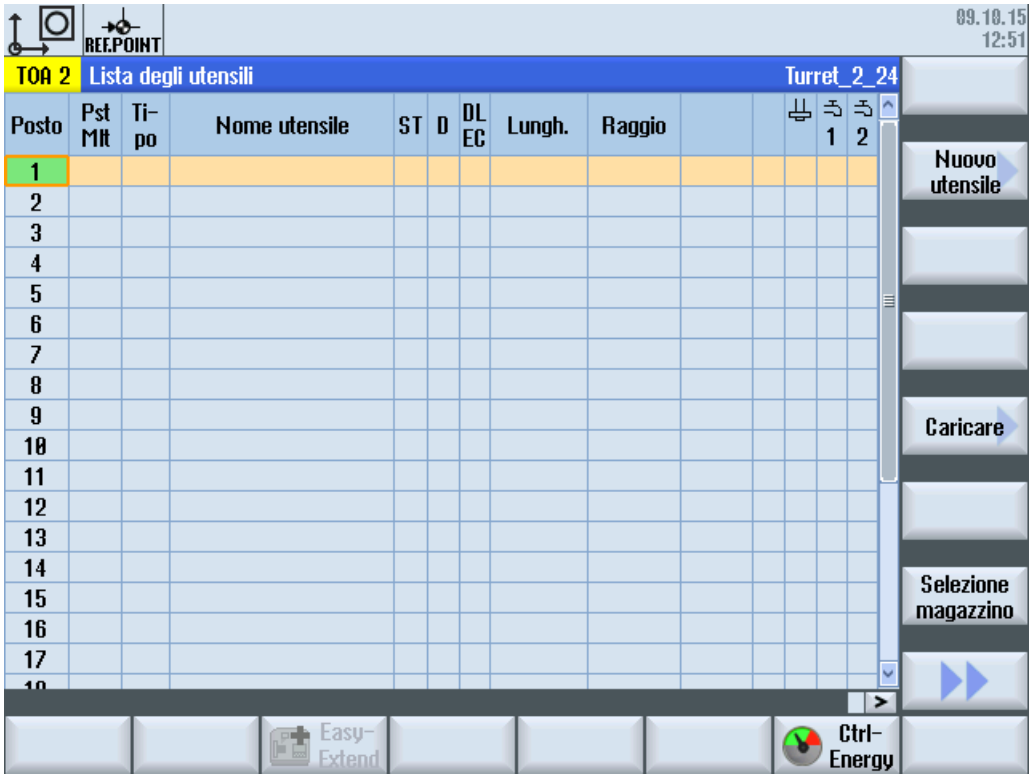


Figura 4-1 Pagina base "Parametri"

Progettazione

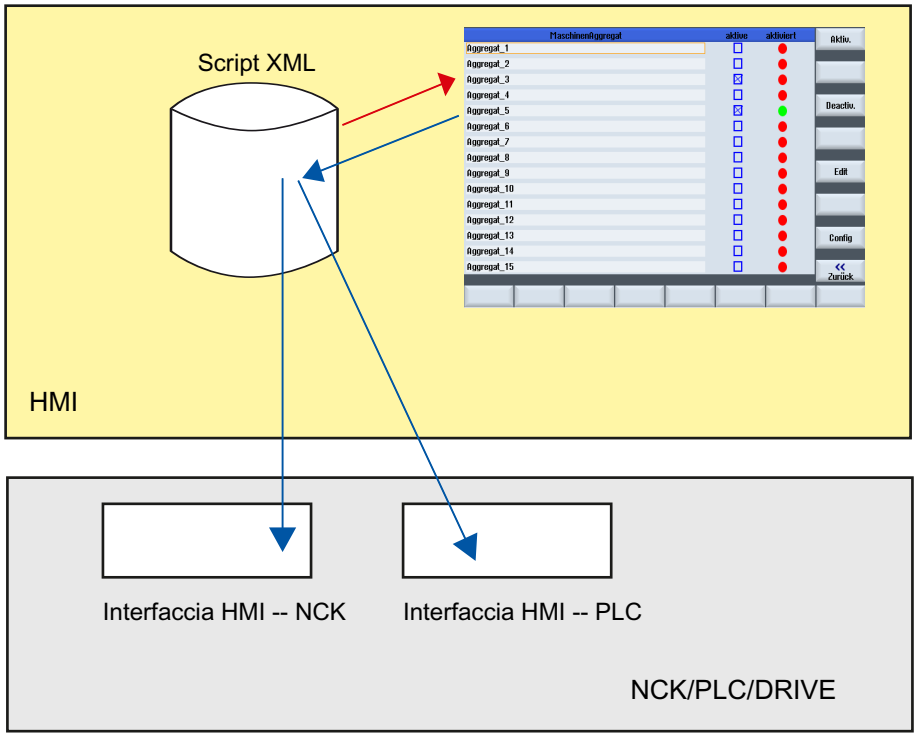


Figura 4-2 Funzionamento di "Easy Extend"

Per poter usare la funzione "Easy Extend", il costruttore della macchina deve progettare le seguenti funzioni:

- **Interfaccia PLC ↔ HMI**
  La gestione dei dispositivi aggiuntivi avviene tramite l'interfaccia tra l'HMI e il PLC.
- **Elaborazione degli script**
  Il costruttore della macchina memorizza in uno script di istruzioni le sequenze da eseguire per l'installazione, l'attivazione, la disattivazione e il test di un dispositivo.
- **Finestra di dialogo Parametri (opzionale)**
  La finestra di dialogo Parametri consente di visualizzare le informazioni sui dispositivi che sono state memorizzate in un file di script.

Archiviazione dei file

L'archiviazione dei file facenti parte di "Easy Extend" avviene sulla scheda CompactFlash di sistema nella directory "dvm" (costruttore della macchina).

File	Nome	Directory di destinazione
File di testo	oem_aggregate_xxx.ts	`\oem\sinumerik\hmi\lng`
File di script	agm.xml	`\oem\sinumerik\hmi\dvm`
File di archivio	A piacere	`\oem\sinumerik\hmi\dvm\archives`
Programma applicativo PLC	A piacere	PLC

4.2 Progettazione nel programma applicativo PLC

Caricamento delle configurazioni

Le configurazioni create vengono trasferite insieme al file di script e di testo nella directory del costruttore del controllore. Inoltre deve essere caricato il programma applicativo PLC corrispondente.

Programmazione dei dispositivi

La comunicazione tra il componente operativo e il PLC avviene nel programma utente PLC mediante un blocco dati definito dal programmatore, nel quale 128 parole sono riservate alla gestione dei dispositivi. Per la descrizione dell'identificatore del blocco dati vedere il capitolo "PLC_INTERFACE (Pagina 281)".

Il blocco dati deve essere generato e caricato come blocco utente. Nello script "agm.xml" il rispettivo blocco deve essere notificato con il tag **plc_interface** della funzione EasyExtend.

Nota

Compatibilità

Si consiglia di definire il blocco dati DB9905 come interfaccia PLC, affinché gli script siano compatibili anche con SINUMERIK 828D.

Esempio:

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

Per la creazione del blocco si possono desumere i formati dei dati e gli identificatori simbolici dalla tabella seguente:

Formato dati / identificatore simbolico	Descrizione
DBX0.0 Enable_1 BOOL bit OFF OFF HMI → PLC	Il dispositivo è stato messo in servizio
DBX0.1 Activate_1 BOOL bit OFF OFF HMI → PLC	Il dispositivo deve essere attivato
DBX0.2 Deactivate_1 BOOL bit OFF OFF HMI → PLC	Il dispositivo deve essere disattivato
DBB1 Res_1 BYTE senza segno 0 0	Riservato per un utilizzo futuro
DBX2.0 IsActive_1 BOOL bit OFF OFF PLC → HMI	Il dispositivo è attivo
DBX2.1 Error_1 BOOL bit OFF OFF PLC → HMI	Il dispositivo è guasto
DBB3 DeviceId_1 BYTE senza segno 0 0	Numero di dispositivo univoco

L'assegnazione delle parole PLC avviene a partire dal dispositivo 1:

Blocco dati	Nome del dispositivo
DB9905.DBB0	Dispositivo 1
DB9905.DBB4	Dispositivo 2
...	...

Blocco dati	Nome del dispositivo
DB9905.DBB192	Dispositivo 49
DB9905.DBB196	Dispositivo 50

Per ogni dispositivo vengono usati quattro byte con il significato seguente:

Byte	Bit	Descrizione	
0	0	== 1	Il dispositivo è stato messo in servizio (risposta dell'HMI)
	1	== 1	Il dispositivo deve essere attivato (richiesta dell'HMI)
	2	== 1	Il dispositivo deve essere disattivato (richiesta dell'HMI)
	3-7	riservato	
1	0-7	riservato	
2	0	== 1	Il dispositivo è attivo (risposta del PLC)
	1	== 1	Il dispositivo è guasto
	2-7	riservato	
3	0-7	Identificativo univoco del dispositivo	

Archiviazione dei file

L'archiviazione dei file avviene sulla scheda CompactFlash di sistema nella directory "oem" (MANUFACTURER) e "oem_i" (INDIVIDUAL).

Nota

Il nome del file può contenere solo lettere minuscole.

File	Nome	Directory di destinazione
File di testo	oem_aggregate_xxx.ts	/oem/sinumerik/hmi/lng/ /oem_i/sinumerik/hmi/lng/
File di script	agm.xml	/oem/sinumerik/hmi/dvm /oem_i/sinumerik/hmi/dvm
File di archivio	A piacere	/oem/sinumerik/hmi/dvm/archives /oem_i/sinumerik/hmi/dvm/archives
Programma applicativo PLC	A piacere	PLC

Procedura generale

Per preparare i dati necessari il costruttore della macchina deve effettuare le operazioni seguenti:

1. Creazione di un programma applicativo PLC che tiene conto dei dispositivi sul lato PLC.
2. Messa in servizio della "macchina standard" con successivo salvataggio dei dati in un archivio di messa in servizio di serie.
3. Montaggio dei dispositivi, messa in servizio e successiva lettura dei dati come archivio di messa in servizio di serie differenziale.

Nota

Modifica della configurazione di macchina

Se i dati macchina dell'azionamento devono essere modificati, occorre prima correggerli nel controllore. Questa procedura va ripetuta per tutti i dispositivi e tutte le situazioni.

Aggiunta di assi

Se la macchina viene ampliata con l'aggiunta di assi macchina, occorre rispettare una sequenza fissa nella struttura dei Drive Object (DO) in quanto l'archivio di messa in servizio di serie contiene la situazione della macchina di riferimento del costruttore della macchina e non può essere applicato in caso di sequenza modificata.

Per i "componenti del controllore" si consiglia di selezionare le impostazioni seguenti:

- Dati NC
- Dati PLC
- Dati di azionamento
  - Formato ACX (binario)

4.3 Rappresentazione sull'interfaccia operativa

Finestre di dialogo sull'interfaccia operativa

Per la funzione "Easy Extend" sono disponibili le seguenti finestre di dialogo:

- Il controllore offre una **finestra di dialogo progettabile** in cui sono visualizzati i dispositivi disponibili.
- Se non è ancora stata effettuata la prima messa in servizio, il controllore apre la **finestra di dialogo di messa in servizio**.

Se per l'apparecchiatura è programmata una procedura di messa in servizio (istruzione XML: "START_UP") e il dispositivo non è ancora stato messo in servizio, il controllore avvia la procedura di messa in servizio.

A questo scopo prima avviene un backup completo dei dati prima che venga letto l'archivio di messa in servizio di serie memorizzato nel file di script.

In caso di errore l'addetto alla messa in servizio può decidere se ripetere la messa in servizio o correggere manualmente gli eventuali errori.

- La funzione "Interruzione" consente di interrompere anticipatamente la messa in servizio. Quindi il controllore ripristina i file di messa in servizio salvati in precedenza.

Se dopo la conclusione corretta della messa in servizio è necessario spegnere la macchina, con l'istruzione XML "POWER_OFF" si può programmare che un messaggio corrispondente venga emesso sul controllore.

4.4 Creazione di testi dipendenti dalla lingua

Struttura del file di testo

I file xml con testi dipendenti dalla lingua devono essere creati in formato UTF8:

Esempi

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>Primo dispositivo</translation>
    </message>
    <source>DEVICE_TWO</source>
    <translation>Secondo dispositivo</translation>
    </message>
</context>
</TS>
```

4.5 Esempio applicativo per una parte di potenza

Attivazione di un Drive Object

Il Drive Object da attivare è già stato messo in servizio e nuovamente disattivato dal costruttore della macchina per commercializzare l'asse o gli assi in opzione.

Per attivare l'asse procedere come segue:

- Attivare l'asse di azionamento tramite p0105.
- Abilitare il secondo asse nei dati macchina del canale.
- Eseguire un backup dei dati macchina dell'azionamento tramite p0971.
- Attendere finché i dati non sono stati scritti.
- Riavviare l'NCK e azionamenti.

Programmazione:

```
<DEVICE>
  <list_id>1</list_id>
  <name> "Attivazione azionamento" </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>
```

Attivazione di un dispositivo comandato da PLC

Il dispositivo viene interrogato tramite il byte di uscita 10 e segnala al PLC lo stato di pronto al funzionamento con il byte di ingresso 9.

Per l'attivazione si imposta il byte di uscita con una codifica fissa. Successivamente il loop While attende lo stato di pronto al funzionamento del dispositivo.

Programmazione:

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

4.6 Lingua di script

Nota

Tutti gli elementi di script descritti nella funzione Creazione finestre di dialogo utente (Pagina 27) costituiscono la base della funzione "Easy Extend". Per gestire gruppi aggiuntivi sono stati definiti ulteriori elementi di script.

Parti di programma dello script

Lo script si compone delle seguenti parti:

- Identificatore "Easy Extend"
- Frame per la definizione delle azioni che possono essere eseguite per un dispositivo
- Identificatore per il dispositivo
- Identificatore per la messa in servizio del dispositivo
- Identificatore per l'attivazione del dispositivo
- Identificatore per la disattivazione del dispositivo
- Identificatore per il test del dispositivo
- Identificatore per la finestra di dialogo Parametri

I singoli tag sono descritti nei capitoli seguenti.

Descrizione

Identificatore <tag>	Significato
AGM	Identificatore per la funzione "Assistente MIS"
DEVICE	Identificatore per la descrizione del dispositivo. **Attributi:** • **option_bit** Per la gestione delle opzioni, al dispositivo viene assegnato un numero di bit fisso.
NAME	L'identificatore definisce il nome del dispositivo che deve essere visualizzato nella finestra di dialogo. Se viene usato un riferimento di testo, la finestra di dialogo definisce il testo memorizzato per l'identificatore.
START_UP	L'identificatore contiene la descrizione delle operazioni necessarie per la messa in servizio del dispositivo.

Identificatore <tag>	Significato
SET_ACTIVE	L'identificatore contiene la descrizione delle operazioni necessarie per l'attivazione del dispositivo. **Attributi:** • **timeout** L'attributo consente di specificare un tempo di timeout in secondi. Se lo script non è ancora terminato dopo questo tempo, il sistema interrompe l'elaborazione.
SET_INACTIVE	L'identificatore contiene la descrizione delle operazioni necessarie per la disattivazione del dispositivo. **Attributi:** • **timeout** L'attributo consente di specificare un tempo di timeout in secondi. Se lo script non è ancora terminato dopo questo tempo, il sistema interrompe l'elaborazione.
TEST	L'identificatore contiene le istruzioni con le quali si può controllare la funzionalità di un dispositivo. **Attributi:** • **timeout** L'attributo consente di specificare un tempo di timeout in secondi. Se lo script non è ancora terminato dopo questo tempo, il sistema interrompe l'elaborazione.
UID	Identificatore numerico univoco del dispositivo nell'interfaccia PLC ↔ HMI.
VERSION	Identificatore per una versione

Conferma negativa dell'esecuzione della funzione

Con la variabile "$actionresult" fornita automaticamente il sistema offre la possibilità di comunicare un risultato di esecuzione negativo al parser XML. Se il valore viene impostato a zero, il parser interrompe l'elaborazione della funzione.

Esempio

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>                              Identificatore per "Assistente MIS"
<DEVICE>
  <NAME> Device 1 </NAME>          Identificatore per il dispositivo
  <START_UP>                       Identificatore per la messa in servizio
  …                                del dispositivo
  </START_UP>
  <SET_ACTIVE>                     Identificatore per l'attivazione del
  …                                dispositivo
  </SET_ACTIVE>
  <SET_INACTIVE>                   Identificatore per la disattivazione del
  …                                dispositivo
  </SET_INACTIVE>
```

```
    <TEST>                          Identificatore per il test del dispositivo
    …
    </TEST>
  </DEVICE>
  …
  </AGM>
```

4.6.1 CONTROL_RESET

Descrizione

Questo identificatore consente di riavviare uno o più componenti del controllore. L'elaborazione dello script prosegue solo dopo che il controllore ha ripreso il funzionamento ciclico.

Programmazione

Identificatore:	**CONTROL_RESET**	
Sintassi:	<CONTROL_RESET resetnc="TRUE" />	
Attributi:	resetnc="true"	Il componente NC viene riavviato.
	resetdrive="true"	I componenti dell'azionamento vengono riavviati.

4.6.2 FILE

Descrizione

L'identificatore consente di importare o creare archivi standard.

- Importazione di un archivio:
  Per importare un archivio occorre immettere il nome file dell'archivio.
- Creazione di un archivio:
  Se l'attributo create= "true" è fornito, la funzione crea un archivio standard (*.arc) con il nome specificato e lo salva nella directory .../dvm/archives ab.

Programmazione

Identificatore:	**FILE**	
Sintassi:	<file name ="<nome archivio>" />	
	<file name ="<nome archivio>" create="true" class="<classi di dati>" group="<intervallo>" />	
Attributi:	name	Identificatore per il nome del file

create	Un archivio di messa in servizio viene creato con il nome specificato e salvato nella directory …/dvm/archives/.
group	Specifica i gruppi di dati che devono essere contenuti nell'archivio. Se devono essere salvati più gruppi di dati, i gruppi vanno specificati separandoli con uno spazio. Nell'archivio possono essere contenuti i seguenti gruppi di dati: • NC • PLC • HMI • DRIVES

Esempio

```
<!-- Creazione di un archivio di classe dati -->
<file name="user.arc" create="true"
group="nc plc hmi" />

<!—Importazione dell'archivio nel controllore->
<file name="user.arc" />
```

4.6.3 OPTION_MD

Descrizione

L'identificatore consente di ridefinire un dato macchina opzionale. Nella configurazione di fornitura il sistema utilizza i dati macchina compresi tra MD14510 $MN_USER_DATA_INT[0] e $MN_USER_DATA_INT[3].

Se il programma applicativo PLC utilizza le opzioni, le corrispondenti parole dati vanno fornite in un blocco dati o GUD.

Il dato è organizzato a bit. A partire dal bit 0 si ha un'assegnazione fissa dei bit alla sequenza di elencazione dei dispositivi: il bit 0 è assegnato al dispositivo 1, il bit 1 al dispositivo 2, ecc. Se devono essere gestiti più di 16 dispositivi, gli identificatori di indirizzo vengono assegnati ai gruppi di dispositivi 1-3 tramite l'indice di intervallo.

Nota

Conversione dell'intervallo di valori

L'intervallo di valori del dato macchina MD14510 $MN_USER_DATA_INT[i] è compreso tra -32768 e +32767. Per poter abilitare i dispositivi tramite bit dalla finestra di dialogo dei dati macchina, occorre convertire la combinazione di bit in formato decimale.

Programmazione

Identificatore:	**OPTION_MD**	
Sintassi:	Intervallo 0:	
	<option_md name = "Identificatore di indirizzo del dato" />	
	OPPURE:	
	<option_md name = "Identificatore di indirizzo del dato" index= "0"/>	
	Intervallo da 1 a 3:	
	<option_md name = "Identificatore di indirizzo del dato" index= "Indice di intervallo"/>	
Attributi:	name	Identificatore per gli indirizzi, ad es. $MN_USER_DATA_INT[0]
	index	Identificatore per l'indice di intervallo:
		0 (preimpostazione): dispositivo da 1 a 16
		1: dispositivo da 17 a 32
		2: dispositivo da 33 a 48
		3: dispositivo da 49 a 64

4.6.4 PLC_INTERFACE

Descrizione

L'identificatore consente di ridefinire l'interfaccia PLC ↔ interfaccia HMI. Il sistema prevede 128 parole indirizzabili.

Nota

L'identificatore non è preimpostato e deve essere definito dal programmatore stesso.

Programmazione

Identificatore:	**PLC_INTERFACE**	
Sintassi:	<plc_interface name = "Identificatore di indirizzo del dato" />	
Attributi:	name	Identificatore per gli indirizzi, ad es. "plc/mb170"

Esempio: `plc/mb170`

4.6.5 POWER_OFF

Descrizione

Identificatore per un messaggio che invita l'utente a spegnere la macchina. Il testo del messaggio è memorizzato nel sistema in modo permanente.

Programmazione

Identificatore:	**POWER_OFF**
Sintassi:	<power_off />
Attributi:	--

4.6.6 WAITING

Descrizione

Dopo un reset dell'NC o dell'azionamento il sistema attende il riavvio del componente.

Programmazione

Identificatore:	**WAITING**	
Sintassi:	<WAITING WAITINGFORNC ="TRUE" />	
Attributi:	waitingfornc="true"	Il sistema attende il riavvio dell'NC.
	waitingfordrive="true"	Il sistema attende il riavvio dell'azionamento.

4.6.7 Identificatori XML per la finestra di dialogo

Finestra di dialogo per la parametrizzazione

Per impostare o emettere parametri aggiuntivi durante il runtime, si può impostare una finestra di dialogo per ogni dispositivo. Questa viene visualizzata premendo il softkey "Parametri aggiuntivi".

Per creare la finestra di dialogo sono disponibili tutti gli elementi di script descritti nel capitolo Creazione finestre di dialogo utente (Pagina 27).

Esempio

```
<?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>">          Identificatore per una finestra di
                                       dialogo utente
    <INIT>
      <CONTROL name = "edit1" .../>    Identificatore per un campo di
    </INIT>                            immissione
    <PAINT>                            Identificatore per la
      <TEXT>hello world !</TEXT>       visualizzazione di testo o immagini
    </PAINT>
  </FORM>


</DEVICE>
…
</AGM>
```

4.6.8 SOFTKEY_OK, SOFTKEY_CANCEL

Descrizione

L'identificatore SOFTKEY_OK sovrascrive il comportamento standard alla chiusura di una finestra di dialogo tramite il softkey "OK". L'identificatore SOFTKEY_CANCEL sovrascrive il comportamento standard alla chiusura di una finestra di dialogo tramite il softkey "CANCEL".

All'interno di questo identificatore possono essere eseguite le seguenti funzioni:

- Manipolazioni dati
- Elaborazione condizionata
- Elaborazione di loop

Programmazione

Identificatore:	**SOFTKEY_OK**
Sintassi:	<SOFTKEY_OK> ... </SOFTKEY_OK>
Identificatore:	**SOFTKEY_CANCEL**
Sintassi:	<SOFTKEY_CANCEL> ... </SOFTKEY_CANCEL>

Indice analitico

I

O

P

R

S

T

X