SIEMENS

SINUMERIK 828D
SINUMERIK Integrate Run
MyScreens

Manuale di programmazione

Valido per

Software CNC Versione 4.95

10/2020
6FC5398-4EP40-0CA0

Introduzione 1
Avvertenze di sicurezza di base 2
Insieme delle funzioni 3
Operazioni iniziali 4
Nozioni di base 5
Finestre di dialogo 6
Variabili 7
Comandi di programmazione 8
Elementi grafici e logici 9
Settore operativo Custom 10
Selezione di finestre di dialogo 11
Esempi di maschere di ciclo 12
Progettazione nel Sidescreen 13
Progettazione in Display Manager 14
Liste di riferimento A
Suggerimenti utili B
Elementi animati C

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: 6FC5398-4EP40-0CA0
Ⓟ 07/2021 Con riserva di modifiche

Indice del contenuto

1 Introduzione 9

1.1 Informazioni su SINUMERIK 9

1.2 Informazioni su questa documentazione 9

1.3 Documentazione in Internet 10
1.3.1 Panoramica della documentazione SINUMERIK 828D 10
1.3.2 Panoramica della documentazione dei componenti operativi SINUMERIK 11

1.4 Feedback relativo alla documentazione tecnica 11

1.5 Documentazione mySupport 11

1.6 Service e Support 12

1.7 Informazioni importanti sul prodotto 14

2 Avvertenze di sicurezza di base 15

2.1 Avvertenze di sicurezza generali 15

2.2 Garanzia e responsabilità per gli esempi applicativi 15

2.3 Avvertenze di sicurezza 15

3 Insieme delle funzioni 17

3.1 Insieme delle funzioni di "Run MyScreens" 17

4 Operazioni iniziali 21

4.1 Introduzione 21

4.2 Progetto di esempio 21
4.2.1 Descrizione del job 21
4.2.2 Creazione del file di progettazione (esempio) 25
4.2.3 Salvataggio del file di progettazione nella directory OEM (esempio) 28
4.2.4 Creazione della guida in linea (esempio) 29
4.2.5 Integrazione della guida in linea e salvataggio dei file nella directory OEM (esempio) 32
4.2.6 Copia di "easyscreen.ini" nella directory OEM (esempio) 33
4.2.7 Notifica del file COM in "easyscreen.ini" (esempio) 33
4.2.8 Test del progetto (esempio) 33

5 Nozioni di base 35

5.1 Struttura del file di progettazione 35

5.2 Struttura dell'albero di comando 37

5.3 Definizioni e funzioni per i softkey di accesso 38
5.3.1 Definizione del softkey di accesso 38
5.3.2 Funzioni per softkey di accesso 39

5.4 Gestione degli errori (file di log) 41

5.5 Note su "easyscreen.ini" 42

5.6 Avvertenze per utenti che migrano a "Run MyScreens" ... 44
5.7 Sintassi di progettazione estesa ... 46
5.8 SmartOperation e comando MultiTouch ... 48
6 Finestre di dialogo ... 49
6.1 Struttura ed elementi di una finestra di dialogo ... 49
6.1.1 Definizione della finestra di dialogo ... 49
6.1.2 Definizione delle proprietà delle finestre di dialogo ... 51
6.1.3 Definizione degli elementi delle finestre di dialogo ... 59
6.1.4 Definizione di finestra di dialogo a più colonne ... 60
6.1.5 Finestre di dialogo per password ... 62
6.1.6 Utilizzo di immagini/grafica ... 64
6.2 Definizione delle barre di softkey ... 64
6.2.1 Modifica delle proprietà di softkey durante il tempo di esecuzione ... 67
6.2.2 Testo dipendente dalla lingua ... 70
6.3 Progettazione della guida in linea ... 73
6.3.1 Panoramica ... 73
6.3.2 Creazione di file HTML ... 74
6.3.3 Creazione di un registro della guida ... 77
6.3.4 Integrazione della Guida in linea in SINUMERIK Operate ... 79
6.3.5 Archiviazione dei file della guida ... 81
6.3.6 File della guida in formato PDF ... 81
7 Variabili ... 83
7.1 Definizione delle variabili ... 83
7.2 Esempi applicativi ... 84
7.3 Esempio 1: Assegnazione di tipo di variabile, testi, pagina di help, colori, Tooltips ... 85
7.4 Esempio 2: Assegnazione di tipo di variabile, valori limite, attributi, posizione del testo sintetico ... 86
7.5 Esempio 3: Assegnazione di tipo di variabile, preassegnazione, variabile di sistema o utente, posizione campo di input/output ... 87
7.6 Esempio 4: Campo di toggle e casella di riepilogo ... 87
7.7 Esempio 5: Visualizzazione immagini ... 88
7.8 Esempio 6: Barra di avanzamento ... 88
7.9 Esempio 7: Modalità di immissione della password (asterisco) ... 91
7.10 Parametri delle variabili ... 91
7.11 Particolarità sul tipo di variabile ... 98
7.12 Particolarità sul campo di toggle ... 100
7.13 Particolarità sulla preassegnazione ... 102
7.14 Particolarità sulla posizione del testo sintetico, posizione del campo di input/output ... 103
7.15 Utilizzo di stringhe ... 104
7.16 Variabile CURPOS ... 106

7.17 Variabile CURVER 106
7.18 Variabile ENTRY 107
7.19 Variabile ERR 107
7.20 Variabile FILE_ERR 108
7.21 Variabile FOC 109
7.22 Variabile S_ALEVEL 110
7.23 Variabile S_CHAN 111
7.24 Variabile S_CONTROL 112
7.25 Variabile S_LANG 112
7.26 Variabile S_NCCODEREADONLY 113
7.27 Variabile S_RESX e S_RESY 113

8 Comandi di programmazione 115

8.1 Operatori 115
8.1.1 Operatori matematici 115
8.1.2 Operatori a bit 118

8.2 Metodi 119
8.2.1 ACCESSLEVEL 120
8.2.2 CHANGE 120
8.2.3 CHANNEL 122
8.2.4 CONTROL 122
8.2.5 FOCUS 123
8.2.6 LANGUAGE 124
8.2.7 LOAD 124
8.2.8 UNLOAD 125
8.2.9 OUTPUT 125
8.2.10 PRESS 126
8.2.11 PRESS(ENTER) 127
8.2.12 PRESS(TOGGLE) 128
8.2.13 RESOLUTION 128
8.2.14 RESUME 129
8.2.15 SUSPEND 130
8.2.16 Esempio: Gestione delle versioni con metodi OUTPUT 130

8.3 Funzioni 132
8.3.1 Lettura e scrittura dei parametri di azionamento: RDOP, WDOP, MRDOP 133
8.3.2 Richiamo del sottoprogramma (CALL) 135
8.3.3 Definizione del blocco (//B) 136
8.3.4 Check Variable (CVAR) 137
8.3.5 Funzione file Copy Program (CP) 138
8.3.6 Funzione file Delete Program (DP) 139
8.3.7 Funzione file Exit Program (EP) 140
8.3.8 Funzione file Move Program (MP) 141
8.3.9 Funzione file Select Program (SP) 142
8.3.10 Accessi ai file: RDFILE, WRFILE, RDLINEFILE, WRLINEFILE 142
8.3.11 Dialog Line (DLGL) 145
8.3.12 DEBUG 146

8.3.13 Uscita dalla finestra di dialogo (EXIT) ... 147
8.3.14 Manipolazione dinamica delle liste dei campi di toggle o ListBox ... 148
8.3.15 Evaluate (EVAL) ... 151
8.3.16 Exit Loading Softkey (EXITLS) ... 152
8.3.17 Function (FCT) ... 152
8.3.18 Generate Code (GC) ... 154
8.3.19 Funzioni per password ... 157
8.3.20 Load Array (LA) ... 158
8.3.21 Load Block (LB) ... 159
8.3.22 Load Mask (LM) ... 160
8.3.23 Load Softkey (LS) ... 162
8.3.24 Load Grid (LG) ... 163
8.3.25 Selezione multipla SWITCH ... 164
8.3.26 Multiple Read NC PLC (MRNP) ... 165
8.3.27 P_LOGICAL_FILEPATH ... 167
8.3.28 P_REAL_FILEPATH ... 167
8.3.29 Servizi PI ... 168
8.3.30 Lettura (RNP) e scrittura (WNP) di variabili di sistema e variabili utente ... 168
8.3.31 RESIZE_VAR_IO e RESIZE_VAR_TXT ... 170
8.3.32 Register (REG) ... 171
8.3.33 RETURN ... 173
8.3.34 Decompilare ... 173
8.3.35 Decompilazione senza commento operativo ... 175
8.3.36 Search Forward, Search Backward (SF, SB) ... 178
8.3.37 SL_HMI_INSTALL_DIR ... 179
8.3.38 SL_TEMP_DIR ... 179
8.3.39 Funzioni STRING ... 180
8.3.40 Loop WHILE/UNTIL ... 186
8.3.41 Esecuzione ciclica degli script: START_TIMER, STOP_TIMER ... 188

9 Elementi grafici e logici ... 191

9.1 Linea, linea di separazione, rettangolo, cerchio ed ellisse ... 191
9.1.1 Definizione degli elementi grafici ... 191
9.1.2 Funzioni grafiche ... 194

9.2 Definizione di un array ... 196
9.2.1 Accesso al valore di un elemento array ... 197
9.2.2 Esempio: Accesso a un elemento array ... 199
9.2.3 Richiamo dello stato di un elemento array ... 201

9.3 Descrizione tabelle (grid) ... 201
9.3.1 Definizione di una tabella (grid) ... 203
9.3.2 Definizione delle colonne ... 204
9.3.3 Controllo della selezione nella tabella (grid) ... 205

9.4 Custom Widget ... 206
9.4.1 Definizione dei Custom Widget ... 206
9.4.2 Struttura della libreria Custom Widget ... 206
9.4.3 Struttura dell'interfaccia Custom Widget ... 207
9.4.4 Interazione fra il Custom Widget e la finestra di dialogo - scambio dati automatico ... 209
9.4.5 Interazione fra Custom Widget e finestra di dialogo - scambio dati manuale ... 210
9.4.5.1 Lettura e scrittura di proprietà ... 210
9.4.5.2 Esecuzione di un metodo del Custom Widget ... 212
9.4.5.3 Reazione a un segnale Custom Widget ... 215

9.5 SlEsGraphCustomWidget ... 217
9.5.1 SlEsGraphCustomWidget ... 217
9.5.2 Hinweise zur Performance ... 219
9.5.3 Properties lesen und schreiben ... 220
9.5.4 Properties ... 220
9.5.5 Funktionen ... 231
9.5.6 Signale ... 246
9.6 SIEsTouchButton ... 248
9.6.1 SIEsTouchButton ... 248
9.6.2 Properties lesen und schreiben ... 249
9.6.3 Properties ... 250
9.6.4 Funktionen ... 266
9.6.5 Signale ... 268
9.6.6 Posizionamento e allineamento di immagine e testo ... 271
9.6.7 Testi dipendenti dalla lingua ... 273
10 Settore operativo Custom ... 275
10.1 Come attivare il settore operativo "Custom" ... 275
10.2 Come progettare il softkey per "Custom" ... 275
10.3 Come progettare il settore operativo "Custom" ... 276
10.4 Esempio di programmazione per il settore "Custom" ... 278
11 Selezione di finestre di dialogo ... 283
11.1 Selezione di finestre di dialogo tramite softkey PLC ... 283
11.1.1 Richiamo delle maschere dei cicli Run MyScreens nell'editor di programma ... 284
11.2 Selezione di finestre di dialogo tramite hardkey PLC ... 286
11.3 Selezione di finestre di dialogo tramite NC ... 288
12 Esempi di maschere di ciclo ... 289
12.1 Esempi di maschere di ciclo ... 289
13 Progettazione nel Sidescreen ... 291
13.1 Visualizzazione di una progettazione Run MyScreens in una Sidescreen Page ... 292
13.2 Visualizzazione di più progettazioni Run MyScreens in una pagina Sidescreen ... 294
13.3 Aggiunta di progettazioni Run MyScreens in una Sidescreen Page ... 296
13.4 Aggiunta di progettazioni Run MyScreens a un elemento Sidescreen ... 297
13.5 Avvertenze sull'esecuzione delle progettazioni Run MyScreens nel Display-Manager ... 299
14 Progettazione in Display Manager ... 301
14.1 Visualizzazione delle progettazioni Run MyScreens nel Sidescreen ... 301
14.2 Integrazione in SISideScreenDialog ... 301
14.3 Integrazione in SIWidgetsApp ... 302
14.4 Avvertenze sull'esecuzione delle progettazioni Run MyScreens in Display Manager ... 303

A Liste di riferimento ... 305

A.1 Elenchi dei softkey di accesso ... 305
A.1.1 Elenco dei softkey di accesso per Tornitura ... 305
A.1.2 Elenco dei softkey di accesso per Fresatura ... 306

A.2 Elenco dei softkey predefiniti ... 308

A.3 Lista dei livelli di accesso ... 308

A.4 Elenco dei colori ... 309

A.5 Lista degli identificativi delle lingue nel nome file ... 310

A.6 Elenco delle variabili di sistema accessibili ... 311

A.7 Comportamento all'apertura della finestra di dialogo (attributo CB) ... 311

B Suggerimenti utili ... 313

B.1 Suggerimenti di carattere generale ... 313

B.2 Suggerimenti per il debugging ... 314

B.3 Suggerimenti per il metodo CHANGE ... 314

B.4 Suggerimenti per i loop DO-LOOP ... 315

C Elementi animati ... 317

C.1 Premessa ... 317

C.2 Modellazione ... 318
C.2.1 Presupposti ... 318
C.2.2 Regole per la modellazione ... 318
C.2.3 Importazione di grafici (modelli) ... 321
C.2.4 Modelli per la modellazione ... 323

C.3 Comandi XML ... 324
C.3.1 Panoramica ... 324
C.3.2 Struttura del file di descrizione scene ... 325
C.3.3 Specularità e rotazioni ... 328
C.3.4 Tipo di vista ... 328

C.4 Conversione in file hmi ... 329

C.5 Visualizzazione in Create MyHMI /3GL ... 329
C.5.1 X3D Viewer ... 329
C.5.2 Classe SlX3dViewerWidget ... 329
C.5.3 Metodi pubblici ... 330
C.5.4 Slot pubblici ... 330
C.5.5 Librerie ... 331
C.5.6 Esempio di implementazione ... 331
C.5.7 Dati macchina ... 331
C.5.8 Note relative all'applicazione ... 332

C.6 Visualizzazione in Run MyScreens ... 332

Indice analitico ... 335

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 destinata a programmatori e progettisti.

Vantaggi

Con l'ausilio del Manuale di programmazione i destinatari hanno la possibilità di progettare, scrivere e testare programmi e interfacce software e di eliminare gli eventuali errori.

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.

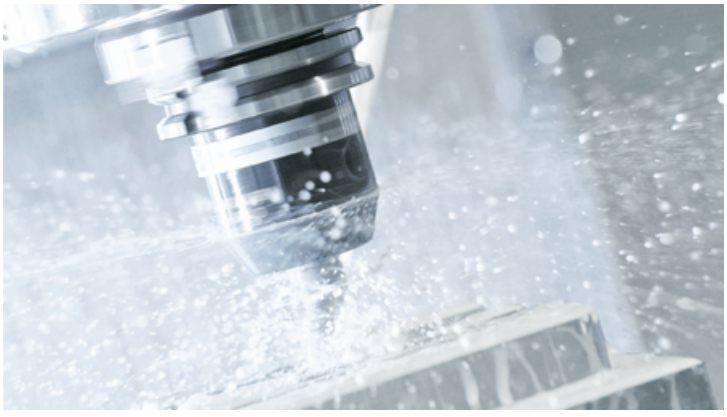
Pagine web di terze parti

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 828D

Una documentazione completa sulle funzioni di SINUMERIK 828D a partire dalla versione 4.8 SP4 è riportata in Panoramica della documentazione 828D (https://support.industry.siemens.com/cs/ww/en/view/109766724).



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: Progettazione
- Costruttore/Service: Messa in servizio
- Costruttore/Service: Funzioni
- Costruttore/Service: Safety Integrated
- SINUMERIK Integrate/MindApp
- Informazioni / Training

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).

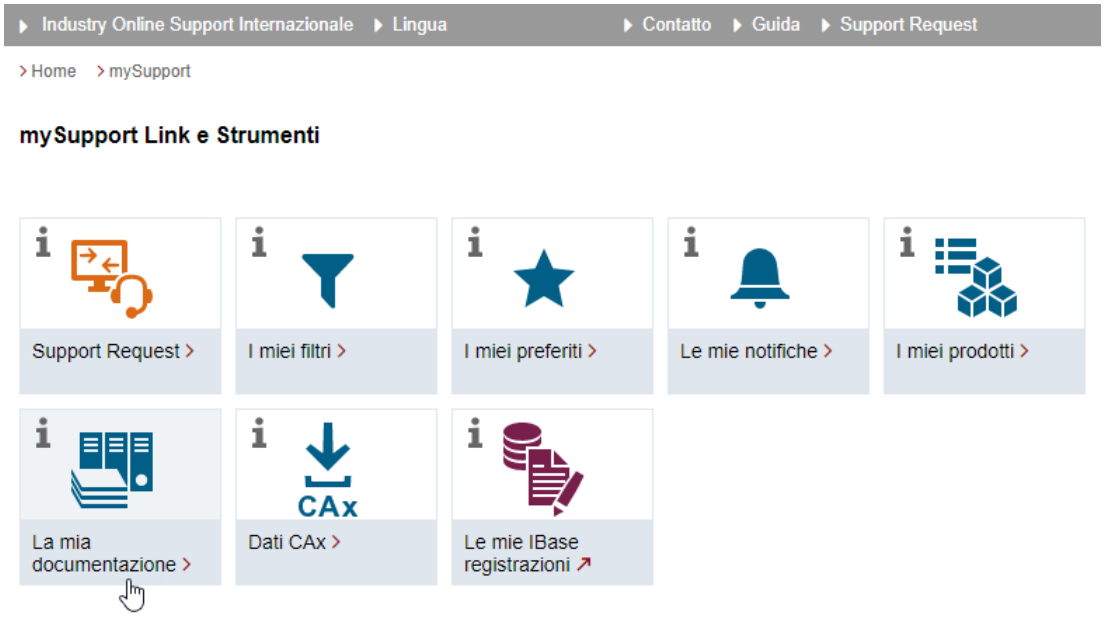
1.4 Feedback relativo alla documentazione tecnica

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

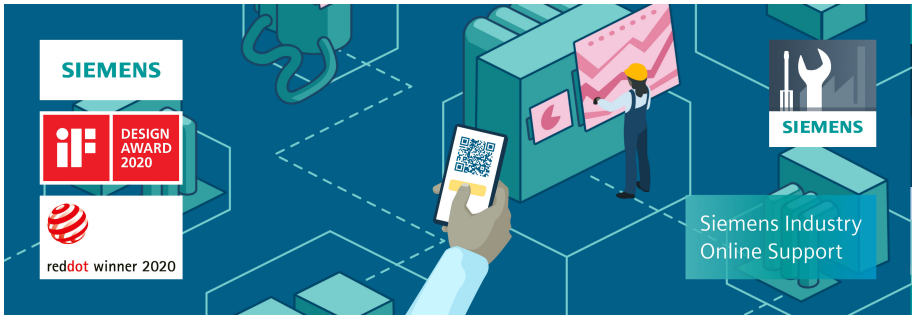
I numeri telefonici della consulenza tecnica specifica dei vari Paesi si trovano in Internet al seguente indirizzo (https://support.industry.siemens.com/cs/ww/it/sc/4868) nella sezione "Contatti".

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

Pericolo di morte in caso di mancata osservanza delle avvertenze di sicurezza e dei rischi residui

In caso di mancata osservanza delle avvertenze di sicurezza e dei rischi residui indicati nella relativa documentazione hardware possono verificarsi degli incidenti che possono causare gravi lesioni o la morte.

- Rispettare le avvertenze di sicurezza contenute nella documentazione hardware.
- Nella valutazione dei rischi occorre tenere conto dei rischi residui.

⚠ 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 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.

2.3 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.

Insieme delle funzioni 3

3.1 Insieme delle funzioni di "Run MyScreens"

Panoramica

"Run MyScreens" consente di realizzare interfacce operative che rappresentano ampliamenti di funzioni specifici del costruttore della macchina o dell'utente oppure che realizzano un layout specifico dell'utente.

Con interfacce operative create dall'utente è possibile generare, tra l'altro, i richiami dei cicli. La definizione delle finestre di dialogo può avvenire direttamente nel controllo.

La funzione "Run MyScreens" viene realizzata tramite un interprete e file di progettazione che contengono la descrizione delle interfacce operative.

"Run MyScreens" viene configurato attraverso file ASCII: questi file di progettazione contengono la descrizione dell'interfaccia operativa. La sintassi per la realizzazione dei file è descritta nei capitoli seguenti.

Funzioni base

La funzione "Run MyScreens" permette al costruttore della macchina di progettare le proprie finestre di dialogo. Già nelle funzioni base è possibile progettare 5 finestre di dialogo nella struttura di menu di comando o per finestre di dialogo di cicli specifici del cliente.

Opzione software

Per ampliare il numero delle finestre di dialogo è necessaria una delle seguenti opzioni software:

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

Condizioni secondarie

Vanno rispettate le seguenti condizioni:

- Il passaggio da una finestra di dialogo all'altra è possibile solo all'interno di un settore operativo.
- Le variabili utente non possono avere lo stesso nome di quelle di sistema o PLC (vedere anche il Manuale delle liste Variabili di sistema/PGAsl/)
- Le finestre di dialogo attivate dal PLC costituiscono un proprio settore operativo (simile alle schermate dei cicli di misura).

Tool

- Editor con funzionalità UTF8 (ad es. l'editor integrato dell'interfaccia operativa o Blocco note)
- Per la creazione di grafici/immagini è richiesto un programma di grafica.

Nomi di file e codifica

Nota

Se sulla NCU si utilizza HMI Operate, tutti i nomi file vengono salvati con lettere minuscole (com, png, txt) sulla scheda CF. Si tratta di un requisito del sistema operativo Linux.

Sulla PCU i nomi file si possono scrivere indifferentemente con lettere maiuscole e minuscole. Si raccomanda tuttavia di utilizzare anche qui le lettere minuscole, ad es. per un eventuale passaggio futuro a Linux.

Nota

Per il salvataggio dei file di progettazione e della lingua assicurarsi che nell'editor utilizzato la codifica sia impostata su UTF-8.

Percorsi di archiviazione

Per l'archiviazione dei file di progettazione, dei file della Guida, ecc. osservare la seguente convenzione:

[cartella di sistema Siemens]	
Linux:	/card/siemens/sinumerik/hmi
Windows 7:	C:\ProgramData\Siemens\MotionControl\siemens
[cartella di sistema oem]	
Linux:	/card/oem/sinumerik/hmi
Windows 7:	C:\ProgramData\Siemens\MotionControl\oem
[cartella di sistema user]	
Linux:	/card/user/sinumerik/hmi

	Windows 7:	C:\ProgramData\Siemens\MotionControl\user
[cartella di sistema addon]		
	Linux:	/card/addon/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\addon
Luoghi di archiviazione		
	Progettazione "Run MyScreens":	[cartella di sistema oem]/proj
	File di configurazione:	[cartella di sistema oem]/cfg
	File della lingua:	[cartella di sistema oem]/lng
	File di immagini:	[cartella di sistema oem]/Ico/ico[risoluzione] Esempio: [cartella di sistema oem]/Ico/ico640
	Guida in linea:	[cartella di sistema oem]/hlp/[lingua] Esempio: [cartella di sistema oem]/hlp/eng

Impiego

È possibile realizzare le seguenti funzioni:

- Visualizzazione di finestre di dialogo e messa a disposizione di:
  - softkey
  - variabili
  - testo e testo di help
  - grafici e figure di help
- Richiamo delle finestre di dialogo mediante:
  - azionando il softkey (accesso)
  - selezione da PLC/NC
- 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
- Avvio di azioni mediante:
  - visualizzazione di finestre di dialogo
  - immissione di valori (variabili)
  - azionamento di softkey
  - uscita dalle finestre di dialogo
  - modifica del valore di una variabile di sistema o utente

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

Operazioni iniziali

4

4.1 Introduzione

L'esempio seguente illustra quali sono le operazioni necessarie per inserire le proprie finestre di dialogo nell'interfaccia operativa SINUMERIK Operate con Run MyScreens. L'utente apprende inoltre come creare proprie finestre di dialogo, inserire pagine e richiami di help, definire softkey e navigare tra le finestre di dialogo.

4.2 Progetto di esempio

4.2.1 Descrizione del job

Nel progetto di esempio verranno create le finestre di dialogo descritte di seguito, inclusa una guida in linea contestuale.

Finestra di dialogo 1

Nella prima finestra di dialogo vengono visualizzati i parametri R scrivibili e (0 e 1) e i nomi degli assi di geometria scrivibili (campi di immissione). Per entrambi i parametri R vengono integrate le corrispondenti pagine di help. Per gli assi di geometria viene integrata una guida contestuale. Inoltre la finestra di dialogo contiene esempi di linee di separazione (orizzontali e verticali), campi toggle, campi I/O che integrazione la selezione delle unità e barre di avanzamento (con e senza evidenziazione con colore).

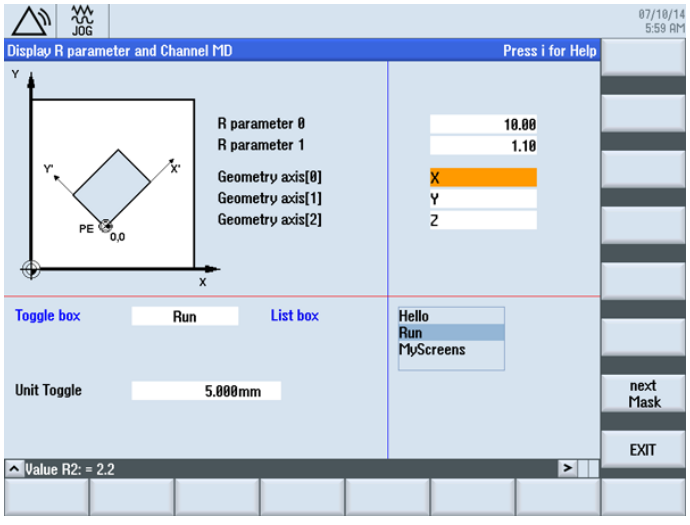


Figura 4-1 Parametri R con pagine di help

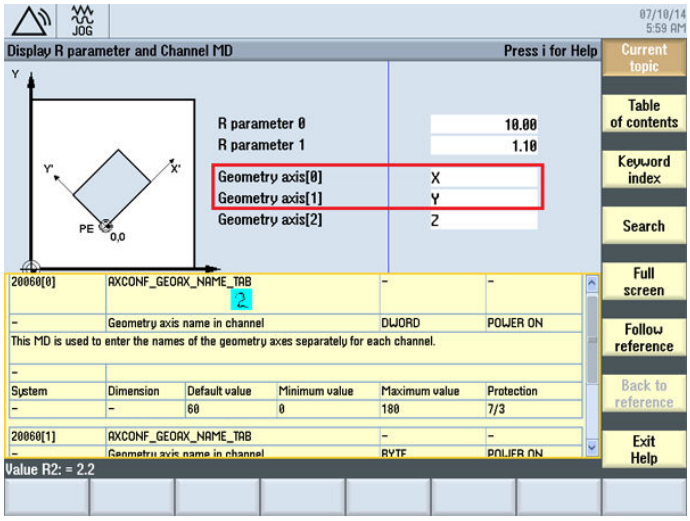


Figura 4-2 Assi di geometria con guida in linea contestuale

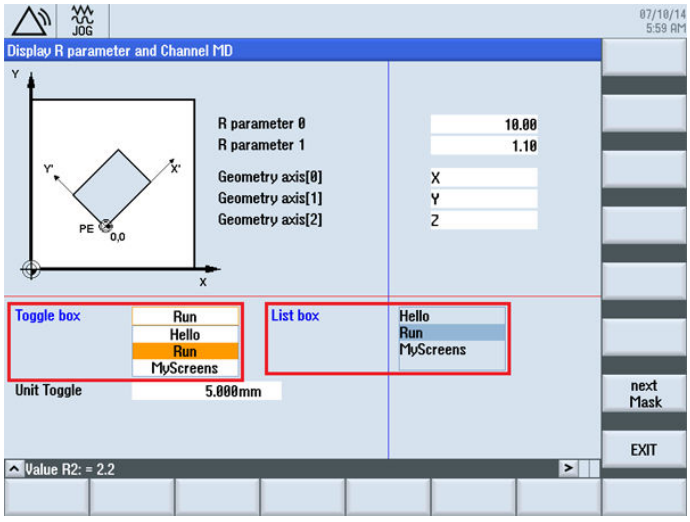


Figura 4-3 Campi toggle: lista e casella

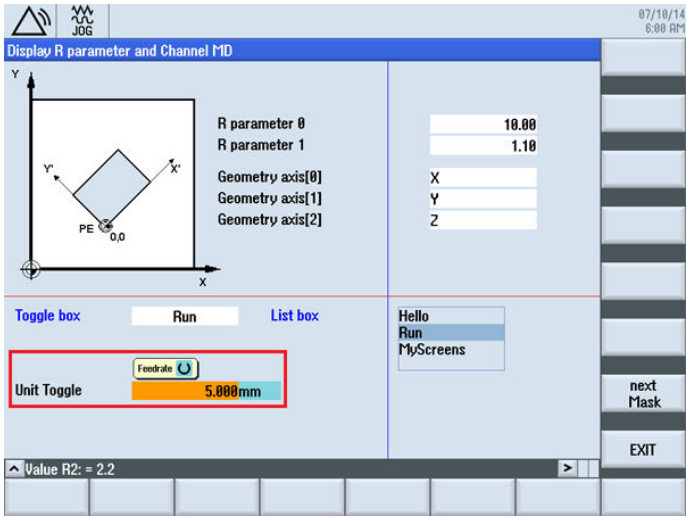


Figura 4-4 Campo I/O con selezione di unità integrata

Finestra di dialogo 2

Nella seconda finestra di dialogo sono visualizzati i valori SCM e SCP. Inoltre la finestra di dialogo contiene esempi di un indicatore di avanzamento con e senza evidenziazione con colore.

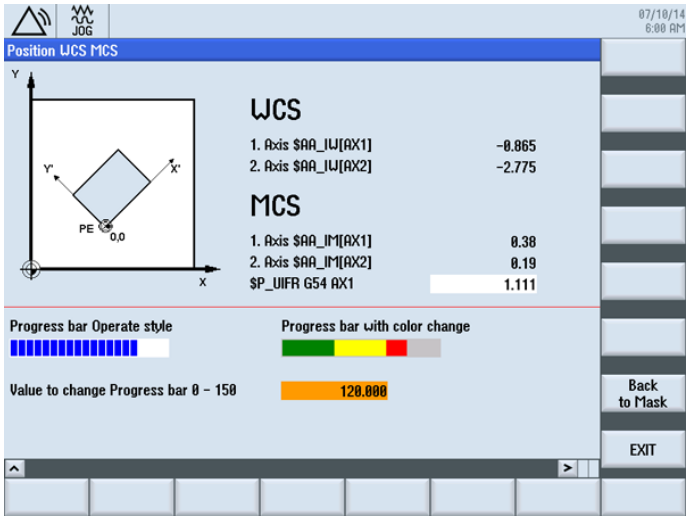


Figura 4-5 Barre di avanzamento con (a destra) e senza (a sinistra) evidenziazione con colore

Navigazione

Il richiamo della prima finestra di dialogo avviene tramite il softkey di accesso "START" nel settore operativo Diagnostica. Si utilizza il softkey orizzontale SK7.

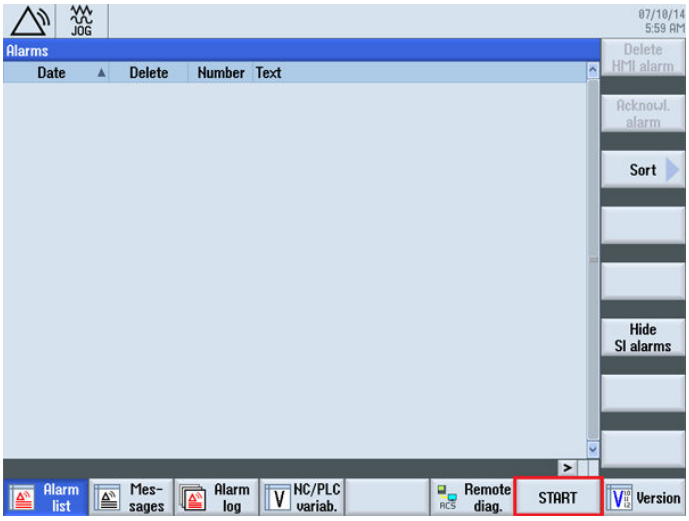


Figura 4-6 Softkey di accesso „START" nel settore operativo Macchina, modo operativo AUTO

A partire dalla prima finestra di dialogo è possibile richiamare, con il softkey "Next Mask", la seconda finestra di dialogo. Con il softkey "EXIT" si ritorna nella pagina base del settore operativo (vedere la figura precedente).

Anche dalla seconda finestra di dialogo si può tornare alla pagina base del settore operativo tramite il softkey "EXIT" (vedere la figura precedente). Tramite il softkey "Back to Mask" è possibile fare ritorno alla prima finestra di dialogo.

Procedura

Nei capitoli seguenti vengono spiegate le operazioni necessarie:

1. Creazione del file di progettazione (file COM) (Pagina 25)
2. Salvataggio del file di progettazione nella directory OEM (Pagina 28)
3. Creazione della guida in linea (Pagina 29)
4. Integrazione della guida in linea e salvataggio dei file nella directory OEM (Pagina 32)
5. Copia di "easyscreen.ini" nella directory OEM (Pagina 33)
6. Notifica del file COM in "easyscreen.ini" (Pagina 33)
7. Test del progetto (Pagina 33)

4.2.2 Creazione del file di progettazione (esempio)

Contenuto del file di progettazione

Creare il file di progettazione "diag.com" per le due finestre di dialogo in un editor che supporta la codifica UTF8.

```
; Codice iniziale softkey di accesso
//S(START)
   ; Softkey di accesso solo testo
   HS7=("START")
   ; Softkey di accesso per testo in funzione della lingua e png
   ;HS7=([$80792,"\\sk_ok.png"])

   ; Metodo Press
   PRESS(HS7)
      ; Funzione LM o LS
      LM("MASK1")
      ; LM con indicazione di un file com
      LM("MASK1","TEST.COM")
   END_PRESS
; Codice finale softkey di accesso
//END

; Definizione finestra di dialogo 1 con titolo e figura
//M(MASK1/"Display R parameter and Channel MD"/"mz961_01.png")
   ; Definizione delle variabili
   DEF VAR1 = (R2///,"R parameter 0"///"$R[0]"/200,50,150/400,50,100,)
   ; Con pagina di help
   DEF VAR2 = (R2///,"R parameter 1"//"mz961_02.png"/"$R[1]"/200,70,150/400,70,100)
   ; Con Guida in linea
   DEF ACHS_NAM1 = (S///"Press i for Help","Geometry axis[0]"////200,100,150/400,100,100//"sinume-
rik_md_1.html","20060[0]")
   ; Con Guida in linea
   DEF ACHS_NAM2 = (S///"Press i for Help","Geometry axis[1]"////200,120,150/400,120,100//"sinume-
rik_md_1.html","20060[1]")
   DEF ACHS_NAM3 = (S///,"Geometry axis[2]"////200,140,150/400,140,100)
   ; Definizione campi di toggle e toggler di unità
   DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run"/,"Toggle box"////10,230,100/120,230,100/,,6)

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

```
; Definizione dei softkey nella finestra di dialogo
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("next Mask")
VS8=("EXIT")

; Definizione blocco LOAD
LOAD
   ; Lettura valore con RNP
   ACHS_NAM1 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]")
   ACHS_NAM2 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[1]")
   ACHS_NAM3 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[2]")
   ; Emissione di una riga di dialogo
   DLGL("Value R2: = " << RNP("$R[2]"))
   ; Scrittura di un valore con WNP
   WNP("$R[3]",VAR0)

  ; Linea di separazione verticale
  V_SEPARATOR(360,1,6,1)
  ; Linea di separazione orizzontale
  H_SEPARATOR(220,1,7,1)
END_LOAD

; Metodo Press
PRESS(VS7)
   ; Caricamento di un'altra finestra di dialogo
   LM("MASK2")
; Codice finale metodo Press
END_PRESS

; Metodo Press
PRESS(VS8)
```

```
      ; Chiusura della finestra di dialogo
      EXIT
   ; Codice finale metodo Press
   END_PRESS
; Codice finale finestra di dialogo 1
//END

; Definizione finestra di dialogo 2 con titolo e figura
//M(MASK2/"Position WCS MCS"/"mz961_01.png")
   ; Definizione delle variabili
   DEF TEXT1 = (I///,"WCS"/WR0,fs2///230,30,120/,,1)
   DEF VAR1 = (R3///,"1. Axis $AA_IW[AX1]"/WR1//"$AA_IW[AX1]"/230,70,150/400,70,100)
   DEF VAR2 = (R3///,"2. Axis $AA_IW[AX2]"/WR1//"$AA_IW[AX2]"/230,90,150/400,90,100)

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

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

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

  ; Definizione dei softkey nella finestra di dialogo
  HS1=("")
  HS2=("")
  HS3=("")
  HS4=("")
  HS5=("")
  HS6=("")
  HS7=("")
  HS8=("")
  VS1=("")
  VS2=("")
  VS3=("")
  VS4=("")
  VS5=("")
  VS6=("")
  VS7=("Back to Mask")
  VS8=("EXIT")
```

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

   ; Metodo Change
   CHANGE(VAR5)
      ; Funzione PI_START
      PI_START("/NC,201,_N_SETUFR")
   ; Codice finale metodo Change
   END_CHANGE

   ; Metodo Press
   PRESS(RECALL)
      ; Ritorno alla maschera di partenza
      LM("MASK1")
   ; Codice finale metodo Press
   END_PRESS

   ; Metodo Press
   PRESS(VS7)
      ; Ritorno alla maschera di partenza
      LM("MASK1")
   ; Codice finale metodo Press
   END_PRESS

   ; Metodo Press
   PRESS(VS8)
      ; Chiusura della finestra di dialogo per applicazione standard
      EXIT
   ; Codice finale metodo Press
   END_PRESS
; Codice finale finestra di dialogo
//END
```

4.2.3 Salvataggio del file di progettazione nella directory OEM (esempio)

Percorso di archiviazione

Salvare il file di progettazione "diag.com" nel seguente percorso:

[cartella di sistema oem]/proj

4.2.4 Creazione della guida in linea (esempio)

Creare il file HTML "sinumerik_md_1.html". L'esempio in "Contenuto della Guida in linea" mostra il richiamo di help contestuale tramite **name="20060[0]"** e **name="20060[1]"**.

Note:

Le avvertenze per la creazione di file HTML sono contenute nel capitolo Creazione di file HTML (Pagina 74).

Le avvertenze per l'integrazione della guida in linea sono riportate nel capitolo Integrazione della guida in linea e salvataggio dei file nella directory OEM (esempio) (Pagina 32).

Contenuto della guida in linea

L'esempio non è commentato.

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

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

4.2.5 Integrazione della guida in linea e salvataggio dei file nella directory OEM (esempio)

Integrazione della guida in linea

La sintassi per l'integrazione della guida in linea sono analoghe a quella in SINUMERIK Operate:

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

Nota

Il nome del file HTML deve essere scritto obbligatoriamente in lettere minuscole in LINUX.

Percorso di archiviazione

Salvare il file HTML "sinumerik_md_1.html" per la guida in italiano nel seguente percorso:

[cartella di sistema oem]/hlp/deu

Per altre lingue occorre creare le necessarie cartelle corrispondenti (ad es.: chs, eng, esp, fra, ita ...).

Un elenco delle sigle delle lingue è riportato nell'appendice "Liste di riferimento".

Ulteriori informazioni

Informazioni dettagliate sulla creazione di guide in linea specifiche dell'OEM sono contenute nel capitolo Progettazione della guida in linea (Pagina 73).

4.2.6 Copia di "easyscreen.ini" nella directory OEM (esempio)

Percorso di archiviazione

Copiare il file "easyscreen.ini" dalla directory

[cartella di sistema Siemens]/cfg

nella directory

[cartella di sistema oem]/cfg.

4.2.7 Notifica del file COM in "easyscreen.ini" (esempio)

Adattamento nel file "easyscreen.ini"

Effettuare la seguente modifica nel file "easyscreen.ini" nella cartella OEM. In questo modo il file di progettazione "diag.com" è stato dichiarato.

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

4.2.8 Test del progetto (esempio)

Test del richiamo della finestra di dialogo

Passare al settore operativo Diagnostica. Fare clic sul softkey orizzontale "START".

Quando "Run MyScreens" rileva degli errori nell'interpretazione dei file di progettazione, questi errori vengono memorizzati nel file di testo "easyscreen_log.txt". Per maggiori informazioni vedere il capitolo Gestione degli errori (file di log) (Pagina 41).

Test di pagine di help contestuali e Guida in linea

Effettuare il test delle pagine di help contestuali e della Guida in linea. In caso di errori controllare i percorsi di memorizzazione.

Nozioni di base

5

5.1 Struttura del file di progettazione

Introduzione

La definizione di nuove interfacce operative viene salvata nei file di progettazione. Questi file sono interpretati automaticamente ed il risultato è visualizzato sullo schermo. I file di progettazione non sono presenti allo stato di fornitura, devono quindi essere ancora creati.

Per la creazione dei file di progettazione e della lingua si utilizza un editor ASCII con funzionalità UTF-8 (ad es. l'editor integrato dell'interfaccia operativa o Blocco note). La descrizione può essere ulteriormente chiarita tramite commenti. Come carattere di commento viene aggiunto un ";" davanti ad ogni spiegazione.

Nota

Per il salvataggio dei file di progettazione e della lingua assicurarsi che nell'editor utilizzato la codifica sia impostata su UTF-8.

Le parole chiave (anche in un file COM salvato in formato UTF-8) devono essere costituite solo da caratteri contenuti nel set di caratteri ASCII. In caso contrario l'interpretazione e quindi la visualizzazione delle maschere/finestre di dialogo possono presentare errori.

Progettazione

Ogni settore operativo HMI dispone di softkey di accesso fissi attraverso i quali è possibile passare alle nuove finestre di dialogo create.

Se viene richiamata la funzione "Carica maschera" (ted. Lade Maske, LM) oppure "Carica barra dei softkey" (ted. Lade Softkey-Leiste, LS) in un file di progettazione, è possibile immettere un nuovo nome file in cui sia contenuto l'oggetto richiamato. In questo modo è possibile suddividere la progettazione, ad es. tutte le funzioni di un livello di comando in un proprio file di progettazione.

Struttura del file di progettazione

Un file di progettazione è costituito dai seguenti elementi:

1. Descrizione dei softkey di accesso
2. Definizione delle finestre di dialogo
3. Definizione delle variabili
4. Descrizione dei metodi
5. Definizione delle barre di softkey

Nota

Sequenza

La sequenza fornita nel file di progettazione deve essere obbligatoriamente rispettata.

Esempio:

```
//S (START)                  ; Definizione dei softkey di accesso (opzionale)
....
//END
//M (.....)                  ; Definizione della finestra di dialogo
DEF .....                    ; Definizione delle variabili
LOAD                         ; Descrizione dei blocchi
...
END_LOAD
UNLOAD
...
END_UNLOAD
...
//END
//S (...)                    ; Definizione di una barra dei softkey
//END
```

Percorso di archiviazione dei file di progettazione

I file di progettazione vengono archiviati nel percorso [cartella di sistema user]/proj e corrispondentemente anche nei percorsi [cartella di sistema addon] e [cartella di sistema oem].

Conversione di testi da altre applicazioni HMI

Procedura per convertire un file di testo con codifica codepage secondo la codifica testuale UTF-8:

1. Aprire il file di testo da un PG/PC in un editor di testo.
2. Durante il salvataggio impostare la codifica UTF-8

Il meccanismo di lettura tramite codifica codepage continua ad essere supportata.

5.2 Struttura dell'albero di comando

Principio dell'albero di comando

Più finestre di dialogo collegate tra loro costituiscono un albero di comando. 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.

Dietro a ciascun softkey di accesso può essere creato un albero di comando:

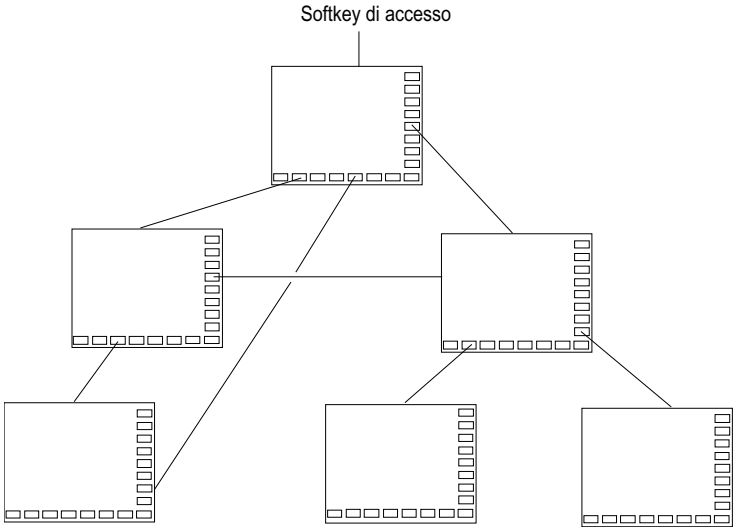


Figura 5-1 Albero di comando

Softkey di accesso

In un file di progettazione indicato in "easyscreen.ini" si definiscono uno o più softkey (softkey di accesso) che fungono da punto iniziale per i propri processi operativi.

Alla definizione di un softkey è collegato il caricamento di una propria finestra di dialogo oppure un'altra barra dei softkey, con cui è possibile effettuare le azioni successive.

Premendo il softkey di accesso viene caricata la finestra di dialogo associata. Vengono anche attivati i softkey appartenenti alla finestra di dialogo. Le variabili sono emesse nelle posizioni standard se non è stata progettata alcuna posizione speciale.

Ritorno all'applicazione standard

È possibile uscire dalla finestra di dialogo modificata e tornare al settore operativo standard.

Il tasto <RECALL>, se non è stato progettato per un'altra azione, permette di uscire dalle nuove interfacce operative realizzate.

Nota

Richiamo delle finestre di dialogo nel programma utente PLC

Oltre che tramite softkey, la selezione delle finestre di dialogo è possibile anche dal PLC: per lo scambio di segnali PLC → HMI esiste un segnale di interfaccia in DB19.DBB10.

5.3 Definizioni e funzioni per i softkey di accesso

5.3.1 Definizione del softkey di accesso

Softkey indipendente dalla finestra di dialogo

I softkey di accesso sono softkey indipendenti dalle finestre di dialogo, i quali non possono essere richiamati a partire da una finestra di dialogo, bensì sono stati progettati **antecedentemente** alla prima nuova finestra di dialogo. Per arrivare a una finestra di dialogo di accesso o a una barra dei softkey di accesso, è necessario definire il softkey di accesso.

Programmazione

Il blocco di definizione per un softkey di accesso è strutturato nel seguente modo:

```
//S(Start)                  ; Codice iniziale softkey di accesso
HS1=(...)                   ; Definizione del softkey di accesso: SK 1 orizzontale
PRESS(HS1)                  ; Metodo
  LM...                     ; Funzione LM o LS
END_PRESS                   ; Fine metodo
//END                       ; Codice finale softkey di accesso
```

Posizioni consentite per i softkey di accesso

Nei settori operativi sono consentite le seguenti posizioni per i softkey di accesso di "Run MyScreens":

Settore operativo	Posizione
Macchina	HSK6
Parametri	HSK7
Programma	HSK6 e HSK15 Cicli di misura: HSK13 e HSK14
Program Manager	HSK2-8 e HSK12-16, se non occupati con unità.

Settore operativo	Posizione
Diagnostica	HSK7
Messa in servizio	HSK7

I softkey di accesso vengono progettati in file speciali. I nomi di questi file vengono dichiarati nel file "easyscreen.ini. Generalmente hanno un nome specifico del settore operativo (ad es. "startup.com" per il settore Messa in servizio). La sola eccezione è costituita dal settore operativo Macchina. Nel settore operativo Macchina esistono più file specifici del modo operativo ("ma_jog.com", "ma_auto.com").

La barra dei softkey con i softkey di accesso si chiama "Start". L'uso delle progettazioni esistenti dei softkey di accesso continua ad essere possibile. La funzionalità di fusione ("Merge") dei softkey di accesso con i softkey dell'applicazione HMI (settore operativo) nel menu dei softkey di accesso non è supportata.

Fino al primo richiamo di una finestra di dialogo, quindi fino al momento da cui è disponibile la funzionalità completa (ad es. l'esecuzione di blocchi PRESS), è solo possibile sostituire completamente un menu o una barra softkey con un elemento analogo.

Modello per la configurazione del softkey di accesso

Una descrizione dettagliata di tutte le posizioni consentite per i softkey di accesso e della loro progettazione si trova nel file "easyscreen.ini" nella seguente directory:

[cartella di sistema Siemens]/cfg

Questo file funge da modello per la propria configurazione del softkey di accesso.

Vedere anche

Elenchi dei softkey di accesso

5.3.2 Funzioni per softkey di accesso

Funzioni per softkey indipendenti dalla finestra di dialogo

Con i softkey di accesso è possibile attivare soltanto particolari funzioni.

Sono consentite le seguenti funzioni:

- Con la **Funzione LM** è possibile caricare un'altra finestra di dialogo: **LM**("Identificatore"["File"])
- Con la **funzione LS** è possibile visualizzare un'altra barra dei softkey: **LS**("Identificatore"[, "File"][, Merge])
- Con la **Funzione "EXIT"** è possibile uscire dalle nuove interfacce operative create e tornare all'applicazione standard.
- Con la **Funzione "EXITLS"** è possibile uscire dall'interfaccia operativa corrente e caricare una barra dei softkey definita.

Metodo PRESS

All'interno del blocco di definizione viene definito il softkey e nel metodo PRESS viene assegnata la funzione "LM" oppure "LS".

Se la definizione del softkey di accesso viene identificata come commento (punto e virgola ; all'inizio della riga) oppure il file di progettazione viene eliminato, il softkey di accesso è privo di funzione.

```
//S(Start)                          ; Codice iniziale
HS6=("1a maschera")                 ; Assegnazione della dicitura "1ª maschera" al SK6
PRESS(HS6)                          ; Metodo PRESS per SK 6 orizzontale
  LM("Maschera1")                   ; Caricamento della funzione Maschera1, tenendo
                                    conto che Maschera1 deve essere definita nello
                                    stesso file.
END_PRESS                           ; Fine del metodo PRESS
HS7=("2a maschera")                 ; Assegnazione della dicitura "2ª maschera" al SK
                                    7 orizzontale
PRESS(HS7)                          ; Metodo PRESS per SK 7 orizzontale
  LM("Maschera2")                   ; Caricamento della funzione Maschera2, tenendo
                                    conto che Maschera2 deve essere definita nello
                                    stesso file.
END_PRESS                           ; Fine del metodo PRESS
//END                               ; Codice finale del blocco di accesso
```

Esempio

```
HS1 = ("nuova barra dei softkey")
HS2 = ("nessuna funzione")
PRESS (HS1)
  LS("Barra1")                        ; Caricamento della nuova barra dei softkey
END_PRESS
PRESS (HS2)                           ; Metodo PRESS vuoto
END_PRESS
```

Progettazioni diverse dei softkey di accesso

Le progettazioni diverse dei softkey di accesso vengono riunite. In un primo tempo viene estratto da "easyscreen.ini" il nome del file da interpretare. Nelle directory seguenti vengono ricercati i file con l'estensione *.com:

- [cartella di sistema user]/proj
- [cartella di sistema oem]/proj
- [cartella di sistema addon]/proj
- [cartella di sistema Siemens]/proj

Le progettazioni per i softkey di accesso contenute vengono ora riunite in un'unica progettazione, ossia i singoli softkey vengono comparati. Se esistono due o più progettazioni per un softkey, nella versione "merge" viene sempre applicato quello con il valore maggiore.

Le barre dei softkey o le finestre di dialogo eventualmente presenti vengono ignorate. Se un softkey contiene un comando senza indicazione del file, ad es. LM("test"), dato che la barra dei softkey o la finestra di dialogo è contenuta nello stesso file, il corrispondente nome file viene integrato nella versione "merge" interna, in modo da non richiedere alcun adattamento. Al termine la progettazione "merge" ottenuta viene visualizzata.

5.4 Gestione degli errori (file di log)

Panoramica

Quando "Run MyScreens" rileva degli errori nell'interpretazione dei file di progettazione, questi errori vengono memorizzati nel file di testo "easyscreen_log.txt". Sono registrati solo gli errori della finestra di dialogo attualmente attivata. Le registrazioni di errore relative a finestre di dialogo attivate in precedenza vengono cancellate.

Il file contiene le seguenti informazioni:

- L'azione durante la quale si è verificato un errore.
- Il numero di riga e colonna del primo carattere errato.
- L'intera riga errata del file di progettazione.
- Le immissioni effettuate mediante la funzione DEBUG.

Nota

Ogni voce è preceduta da un time stamp tra parentesi quadre. Ciò può essere di aiuto, ad es., per le progettazioni con criticità temporale.

Archiviazione del file "easyscreen-log.txt"

Il file "easyscreen_log.txt" è archiviato nella seguente directory:

[cartella di sistema user]/log

Sintassi

L'Interpretazione della sintassi ha inizio solo quando il softkey di accesso è stato definito ed è stata progettata una finestra di dialogo con codice iniziale e finale e una riga di definizione.

```
//S(Start)
HS6=("1ª maschera")
PRESS(HS6)
  LM("Maske1")
```

```
END_PRESS
//END

//M(Maske1)
  DEF Var1=(R)
  DEF VAR2 = (R)
  LOAD
  VAR1 = VAR2 + 1              ; Segnalazione di errore nel file di log, in quanto
                               VAR2 non ha alcun valore
...
//END

; la forma corretta sarebbe
ad es.:
//M(Maske1)
   DEF Var1=(R)
   DEF VAR2 = (R)

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

5.5 Note su "easyscreen.ini"

A partire da SINUMERIK Operate V4.7 il file "easycreen.ini" è esteso con le registrazioni descritte in questo capitolo. Il file "easyscreen.ini" si trova nel percorso [cartella di sistema siemens]/cfg.

Posizione di partenza della pagina di help

Voce in "easyscreen.ini":

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

Nota:

La posizione di partenza delle pagine di help viene posizionata, indipendentemente dalla risoluzione, sulla posizione del pixel progettata (standard=true).

Comportamento di estensione, altezza della riga e distanza tra le righe standard

Registrazione in easyscreen.ini:

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

Note:

- La voce "SymmetricalAspectRatio" determina se vanno osservati gli stessi fattori di estensione per l'adattamento di una progettazione ad una determinata risoluzione dello schermo in direzione X e Y.
  - "false" (standard): alle risoluzioni widescreen, i campi e i grafici vengono compressi in direzione Y (estensioni asimmetriche riferite a 640x480). Ad esempio, un quadrato progettato nella risoluzione 640x480 viene compresso verticalmente in un pannello widescreen e così rappresentato come rettangolo.
  - "true": in direzione X e Y si opera con lo stesso fattore di estensione; i campi e i grafici mantengono le loro proporzioni progettate in origine, riferite a 640x480. Ad esempio, un quadrato progettato nella risoluzione 640x480 continua ad essere rappresentato in un pannello widescreen come quadrato.
- Con le voci "DefaultLineHeight" e "DefaultLineSpacing" è possibile stabilire l'altezza standard della riga (standard: 18 pixel) e interlinea standard (standard: 3 pixel) riferite a 640x480. Esse si attivano sempre solo se nella progettazione per la posizione del testo sintetico o del campo I/O non è specificata una posizione Y o un'altezza.

Cicli nella pagina base Macchina

- Accesso tramite Macchina in modo operativo "JOG" via HS6 con la possibilità di generare un richiamo di ciclo nella pagina base macchine e sezione [JOBSHOPINTEGRATION]:
  ```
  [JOBSHOPINTEGRATION]
  Integration = true
  ```
  oppure tramite blocco di avvio nella progettazione
  ```
  LM("Maschera1",,1)
  PRESS(VS8)
     GC("MOVE_RIDE") ; il richiamo del ciclo viene generato
     EXIT
  END_PRESS
  ```
- Mantenimento dei menu Operate con Sezione [Integration]:
  Se nel dialogo sono contenuti solo softkey verticali, la barra standard dei softkey incluso il softkey di accesso resta attiva durante la visualizzazione del dialogo Se il dialogo contiene sia softkey orizzontali che verticali, la barra orizzontale del dialogo sostituisce la barra orizzontale con il softkey di accesso.
  ```
  [Integration]
  OperateMenusEnabled = true
  ```

Posizione della maschera in funzione della risoluzione: Form Panels

Voce in "easyscreen.ini":

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

Note:

La posizione della maschera può essere definita come segue:

- La posizione della maschera può essere definita in "pixel riferiti a 640x480".
  Esempio:
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**0,219,335,174**`)`
- La posizione della maschera può essere accoppiata alla posizione dipendente dalla risoluzione di un FormPanel da un layout di schermo standard Operate, ad es. "FormPanel4" ("funzioni ausiliarie") dal layout di schermo "slmastandardscreenlayout.SlMaStandardScreenLayout" dell'area "Macchina"
  `//M(MyMask/"MyCaption"/"\`
  `\myhelp.png"/`**"slmastandardscreenlayout.SlMaStandardScreenLayout.FormPanel4"**
  `)`
  .
  In questo modo si possono agevolmente sovrapporre le forme standard di Operate o posizionare esattamente le maschere Easyscreen.
  Esempio:
  `//M(Mask/"Mask"//`**"slstandardscreenlayout.SlStandardScreenLayout.LowerForm"**`)`
  È tuttavia possibile utilizzare qualsiasi altro layout di schermo.
- La posizione della maschera può essere accoppiata nel file "easyscreen.ini" a definizioni autodefinite, dipendenti dalla risoluzione, dei FormPanel.
  Esempio:
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**"MyPanel"**`)`

5.6 Avvertenze per utenti che migrano a "Run MyScreens"

Nota

In caso di utilizzo di HMI Operate nella NCU, si noti che sulla scheda CF tutti i nomi file vengono salvati con lettere minuscole (com, png, txt).

Nota

Per il salvataggio dei file di progettazione e della lingua assicurarsi che nell'editor utilizzato la codifica sia impostata su UTF-8.

File di immagine

Salvare i file di immagine sempre in formato PNG "*.png".

I dati devono essere archiviati, ad es. per adattamenti OEM, in

[directory di sistema oem]/ico/[risoluzione].

Per maggiori informazioni vedere il capitolo Utilizzo di immagini/grafica (Pagina 64).

Adattamento del file di progettazione

Effettuare le seguenti verifiche sui file di progettazione:

- Confrontare i softkey di accesso con i softkey consentiti attualmente ed eventualmente modificarli.
- Rinominare i file delle immagini inseriti secondo quanto specificato nel paragrafo "File di immagini".

I dati vengono archiviati, ad es. per gli adattamenti OEM, nella seguente cartella:

[cartella di sistema oem]/proj

[cartella di sistema user]/proj A

[cartella di sistema addon]/proj

Adattamento dei file di help

Tutti i file di help devono essere memorizzati in formato UTF-8. Controllare i file esistenti e memorizzarli nuovamente con un editor adatto.

I file HTML vengono memorizzati nella seguente cartella, ad es. per il tedesco:

[cartella di sistema oem]/hlp/deu

[cartella di sistema user]/hlp/deu

[cartella di sistema addon]/hlp/deu

Le directory per altre lingue devono essere create in base ai codici delle lingue.

Verifica della licenza di "Run MyScreens"

Verificare che il numero di finestre di dialogo inserite non superi il quantitativo di base massimo di 5.

Per ampliare il numero delle finestre di dialogo è necessaria una delle seguenti opzioni software:

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

5.7 Sintassi di progettazione estesa

A partire da SINUMERIK Operate V4.7 è disponibile una sintassi semplificata per la definizione di maschere, variabili, softkey e per le colonne di tabella. Con questa sintassi alternativa migliora la leggibilità e facilità di manutenzione. Le proprietà e gli attributi possono essere specificati in qualsiasi sequenza, le voci vuote sono superflue. Rispetto alla sintassi valida finora, l'elencazione delle proprietà e degli attributi è racchiusa, anziché come finora tra le parentesi tonde "**(**" e "**)**", entro le parentesi graffe "**{**" e "**}**".

Le proprietà e gli attributi sono specificati come segue:
`{<nome> = <valore>, <nome> = <valore>, …}`

La sintassi precedente resta compatibile.

Sintassi estesa per la definizione delle maschere

//M {<nome maschera> [,**HD**=<intestazione>] [,**HLP**=<grafica>] [,**X**=<posizione X>] [,**Y**=<posizione Y>] [,**W**=<larghezza>] [,**H**=<altezza>] [,**VAR**=<>variabile di sistema o variabile utente] [,**HLP_X**=<posizione X, pagina di help>] [,**HLP_Y**=<posizione Y, pagina di help>] [,**CM**=<allineamento colonne>] [,**CB**=comportamento all'apertura delle finestre di dialogo] [,**XG**=<interpretazione pagina di help come grafica X3d>] [,**PANEL**=<nome del FormPanel combinato>][,**MC**=<colore di sfondo della maschera>][,**AL**=<allineamento dell'intestazione maschera>] [,**LANGFILELIST**=<lista dei file della lingua specifici per maschera]}

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

Sintassi estesa per la definizione delle variabili

DEF <nome di variabile> = {[**TYP**=<tipo>] [,**MIN**=<valore minimo>] [,**MAX**=<valore massimo>] [,**TGL**=<valori di toggle>] [,**VAL**=<preassegnazione>] [,**LT**=<testo completo>] [,**ST**=<testo sintetico>] [,**GT**=<testo grafico>] [,**UT**=<testo unità>] [,**TT**=<testo descrizione comandi>] [,**TG**=<opzione toggle>] [,**WR**=<modalità di immissione>] [,**AC**=<livello di accesso>] [,**AL**=<allineamento testo>] [,**FS**=<dimensione carattere>] [,**LI**=<trattamento valore limite>] [,**UR**=<frequenza di aggiornamento>] [,**CB**=<comportamento all'apertura della finestra di dialogo>] [,**HLP**=<pagina di help>] [,**VAR**=<variabile di sistema o variabile utente>] >] [,**TXT_X**=<posizione X testo sintetico>] [,**TXT_Y**=<posizione Y testo sintetico>] [,**TXT_W**=<larghezza testo sintetico>] [,**TXT_H**=<altezza testo sintetico>] [,**X**=<posizione X campo I/O>] [,**Y**=<posizione Y campo I/O>] [,**W**=<larghezza campo I/O>] [,**H**=<altezza campo I/O>] [,**UT_DX**=<distanza tra campo I/O e campo unità] [,**UT_W**=<larghezza campo unità>] [,**BC**=<colore di sfondo campo I/O>] [,**FC**=<colore di primo piano campo I/O>] [,**BC_ST**=<colore di sfondo testo sintetico>] [,**FC_ST**=<colore di primo piano testo sintetico>] [,**BC_GT**=<colore di sfondo testo grafico>] [,**FC_GT**=<colore di primo piano testo grafico>] [,**BC_UT**=<colore di sfondo testo unità>] [,**FC_UT**=<colore di primo piano testo unità>] [,**SC1**=<colore di segnale 1 per progressbar>] [,**SC2**=<colore di segnale 2 per progressbar>] [,**SVAL1**=<valore di soglia 1 per progressbar>] [,**SVAL2**=<valore di soglia 2 per progressbar>] [,**DT**=<tipo di visualizzazione>] [,**DO**=<allineamento di visualizzazione>] [,**OHLP**=<Guida in linea>][,**LINK_TGL**=<nome della variabile di toggle collegata>]}

Esempi:

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

Sintassi estesa per la definizione dei softkey

SK = {[**ST**=<dicitura>] [,**AC**=<livello di accesso>] [,**SE**=<stato>]}

Esempi:

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

Sintassi estesa per la definizione delle colonne di tabella

{[**TYP**=<tipo>] [,**MIN**=<valore minimo>] [,**MAX**=<valore massimo>] [,**LT**=<testo completo>] [,**ST**=<testo sintetico>] [,**WR**=<modalità di immissione>] [,**AC**=<livello di accesso>] [,**AL**=<allineamento testo>] [,**FS**=<dimensione carattere>] [,**LI**=<trattamento valore limite>] [,**UR**=<frequenza di aggiornamento>] [,**HLP**=<pagina di help>] [,**VAR**=<variabile di sistema o utente>] >] [,**W**=<larghezza colonna>] [,**OF1**=<offset1>] [,**OF2**=<offset2>] [,**OF3**=<offset3>]}

Esempio:

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

5.8 SmartOperation e comando MultiTouch

Per l'adattamento specifico a SmartOperation e per il comando MultiTouch è possibile effettuare le seguenti impostazioni:

- Adattamento automatico dell'altezza e della larghezza del campo alle dimensioni minime utilizzabili del campo e dell'interlinea (attributo di maschera MA).
- Ridimensionamento ottimizzato dei campi rispettando il rapporto di pixel nel caso di risoluzioni superiori (attributo di maschera PA).
- Impostazione dell'altezza del campo e dell'interlinea proporzionale al font (attributo di maschera FA).
- Libero scorrimento dei campi affinché la tastiera virtuale non copra il campo di immissione (attributo di maschera KM).

Per maggiori dettagli, fare riferimento al capitolo Definizione delle proprietà delle finestre di dialogo (Pagina 51), sezione "Programmazione".

Finestre di dialogo

6

6.1 Struttura ed elementi di una finestra di dialogo

6.1.1 Definizione della finestra di dialogo

Definizione

Una finestra di dialogo è parte di un'interfaccia operativa, costituita da riga di intestazione, elementi della finestra di dialogo e/o grafica, riga di output per le segnalazioni nonché 8 softkey orizzontali e 8 verticali.

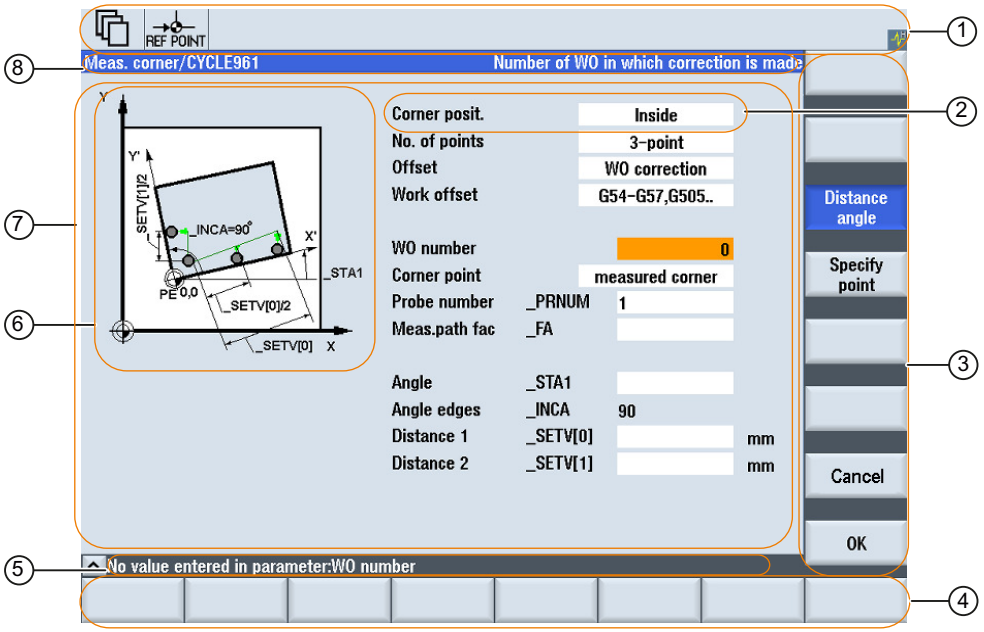
Gli elementi della finestra di dialogo sono:

- Variabili
  - Valori limite/campo toggle
  - Preassegnazione delle variabili
- Pagina di help
- Testi
- Attributi
- Variabile di sistema o utente
- Posizione del testo sintetico
- Posizione del campo di input/output
- Colori

Proprietà di una finestra di dialogo:

- Intestazione
- Grafica
- Dimensione
- Variabile di sistema o utente

- Posizione grafica
- Attributi



① Indicazione dello stato della macchina ("Intestazione")
② Elemento della finestra di dialogo
③ 8 softkey verticali
④ 8 softkey orizzontali
⑤ Segnalazioni di diagnostica emesse
⑥ Grafica
⑦ Finestra di dialogo
⑧ Riga di intestazione della finestra di dialogo, con titolo e testo completo

Figura 6-1 Struttura della finestra di dialogo

Panoramica

In linea di principio la definizione di una finestra di dialogo (blocco di definizione) è strutturata come segue:

Blocco di definizione	Commento	Rimando al capitolo
`//M...`	;Codice iniziale finestra di dialogo	
`DEF Var1=...` `...`	;Variabili	Variabili (Pagina 83)
`HS1=(...)` `...`	;Softkey	Definizione delle barre di softkey (Pagina 64)

`PRESS(HS1)`
`  LM...`
`END_PRESS`
	;Codice iniziale metodo
	;Azioni
	;Codice finale metodo
		Metodi (Pagina 119)

`//END`	;Codice finale finestra di dialogo	

All'interno dei blocchi di definizione della finestra di dialogo vengono definiti in primo luogo diverse variabili, visibili di volta in volta nella finestra di dialogo come elemento della finestra di dialogo stessa, nonché software orizzontali e verticali. Successivamente nei metodi vengono progettate diverse azioni.

Nota

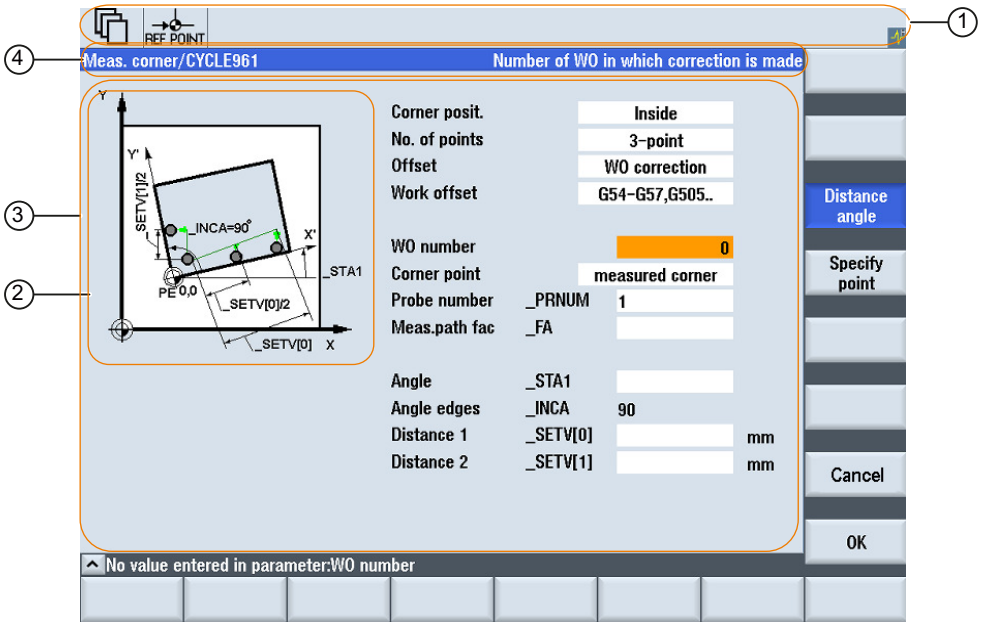
In caso di cambio di finestra di dialogo, nel settore operativo "Macchina" va rispettato il seguente ordine:

A condizione che le dimensioni della maschera successiva siano inferiori a quelle della precedente o che la posizione della maschera successiva sia diversa da quella della precedente, il cambio di finestra di dialogo funziona solo se prima di tornare alla prima maschera la maschera successiva è conclusa e quindi viene caricata la nuova maschera.

6.1.2 Definizione delle proprietà delle finestre di dialogo

Descrizione

Attraverso il codice iniziale della finestra di dialogo vengono definite contemporaneamente le proprietà della finestra di dialogo.



① Indicazione dello stato della macchina ("Intestazione")
② Grafica
③ Finestra di dialogo
④ Riga di intestazione della finestra di dialogo, con titolo e testo completo

Figura 6-2 Proprietà di una finestra di dialogo

Programmazione

Sintassi:	**//M**(Identificatore/[Titolo]/[Grafica]/[Dimensione]/[Variabile di sistema o utente]/[Posizione grafica]/[Attributi])/Lista dei file della lingua specifici della maschera) Vedere anche il capitolo Sintassi di progettazione estesa (Pagina 46).
Descrizione:	Definizione della finestra di dialogo

Parametri:	Identificatore			Nome della finestra di dialogo
	Intestazione			Titolo della finestra di dialogo come testo o richiamo di un testo (ad es. $85011) da un file di testo dipendente dalla lingua
	Grafica			File di grafica con percorso tra doppie virgolette
	Dimensione			Posizione e dimensioni della finestra di dialogo in pixel (distanza da sinistra, distanza dall'alto, larghezza, altezza), riferite all'angolo in alto a sinistra dello schermo. I dati sono separati dalla virgola.
	Variabile di sistema o utente			Variabile di sistema o utente a cui viene assegnata la posizione del cursore corrente. La posizione del cursore può essere indicata all'NC o PLC attraverso la variabile di sistema o utente. La prima variabile ha l'indice 1. La sequenza corrisponde alla sequenza di progettazione delle variabili.
	Posizione grafica			Posizione della grafica (distanza da sinistra, distanza dall'alto) in pixel riferita all'angolo in alto a sinistra della finestra di dialogo. I dati sono separati dalla virgola.
	Attributi			Gli attributi indicati vengono separati dalla virgola. Possibili attributi:
		CM		Column Mode: Allineamento colonna
			CM0	Preimpostazione: la divisione di colonna viene effettuata separatamente per ogni riga.
			CM1	La divisione di colonna della riga con il maggior numero di colonne si applica a tutte le righe.
		CB		CHANGE Block: Comportamento all'apertura della finestra di dialogo: gli attributi cb, indicati in una definizione di variabile, hanno per la suddetta variabile la precedenza sull'indicazione forfettaria nella definizione della finestra di dialogo. Vedere anche AUTOHOTSPOT.
			CB0	Preimpostazione: tutti i blocchi CHANGE della finestra di dialogo vengono elaborati all'apertura.
			CB1	I blocchi CHANGE vengono elaborati solo quando cambia il relativo valore.
		XG		Integrazione di un'animazione X3D come pagina di help (solo nella programmazione passi di lavorazione) Esempio: `//M(Meas/ $85605/"myx3dhelpfile.hmi,,Z_Animation,,G17"///30,10/ XG1)`
			XG0	Preimpostazione = 0
			XG1	L'attributo di maschera XG deve essere impostato su 1 già nella definizione della maschera; al runtime non è più possibile modificarlo. Nota: Anche l'indicazione nella proprietà di maschera HLP deve essere impostata di conseguenza.
		AL		Con l'attributo di maschera AL è possibile influire sull'allineamento del titolo della maschera. Esempio: Output centrato del titolo della finestra "Set password": `//M(MY_PWD_SET/”Set password”//"EasyPwdModalLayout"// AL2)`
			AL0	Allineato a sinistra, preimpostazione
			AL1	A destra

	AL2	Centrato

KM		Se si deve visualizzare la tastiera virtuale per MultiTouch, lo scorrimento libero nella maschera (SlGfwScrollArea) può essere controllato nel modo seguente:
		• Nella riga di definizione della maschera con l'attributo "KM" (Keyboard-Mode)
		Esempio:
		`//M(MyMTMask/"MultiTouch Mask"/////KM1)`
		• Come impostazione globale per tutte le maschere Run MyScreens nel file "easyscreen.ini"
		Esempio:
		`[GENERAL]`
		`DefaultVirtualKeyboardMode=1`
		
		Nota:
		Se nella maschera di progettazione è impostato esplicitamente l'attributo KM, quest'ultimo prevale sull'impostazione globale in "easyscreen.ini".
		
		(Vedere anche il capitolo SmartOperation e comando MultiTouch (Pagina 48))

	KM1	Scorrimento libero attivo (default)
	KM0	Scorrimento libero non attivo
PG		Orientamento e aspetto del titolo della finestra associato alle immagini PNG (ha effetto solo nell'editor!) Esempio: `//M(MyPg1SampleMask/"My PG1 Sample Mask"/////PG1)`
	PG0	(Default)
	PG1	Orientamento e aspetto del titolo della finestra associato alle immagini PNG Indicazione del percorso (tutto a sinistra), nome della maschera (a destra) In questa modalità non è più possibile visualizzare il testo completo. In alternativa è possibile servirsi delle descrizioni comandi.

MA		Adattamento automatico dell'altezza del campo nel comando MultiTouch (MultiTouch Adjustment)
		L'adattamento per un pannello operatore MultiTouch comprende l'eventuale ingrandimento necessario alla dimensione minima utilizzabile (larghezza, altezza) dei campi (eccezioni: barra di avanzamento, GIF animato, Custom-Widget) e l'interlinea.
		L'adattamento MultiTouch può essere:
		• globale per tutte le maschere Run MyScreens nel file "easyscreen.ini", ad es.
		`[GENERAL]`
		`DefaultMultiTouchAdjustmentLevel=1`
		• oppure specifico per ogni maschera nella riga di definizione della stessa con la property "MA", in modo da sovrascrivere l'impostazione globale nel file "easyscreen.ini", ad es.

		`//M(MyMTMask/"MultiTouch Mask"/////AM1)` Se nella maschera di progettazione è impostato esplicitamente l'attributo MA, quest'ultimo prevale sull'impostazione globale in "easyscreen.ini". (Vedere anche il capitolo SmartOperation e comando MultiTouch (Pagina 48))
	MA0	Non avviene alcun adattamento
	MA1	Vengono adattati solo i campi per i quali non è progettata una distanza dall'alto e/o nessuna altezza/larghezza di campo – quindi i campi che sfruttano le posizioni predefinite e di cui pertanto Run MyScreens può calcolare la distanza dall'alto e/o la l'altezza/larghezza. (Default)
	MA2	Vengono adattati tutti i campi indipendentemente dalle grandezze di campo progettate.

PA		Ridimensionamento ottimizzato dei campi rispettando il numero di pixel nel caso di risoluzioni superiori (Pixel Fine Adjustment)
		Finora venivano calcolate prima tutte le posizioni dei campo riferite a 640x480 pixel per poi zoomarle con i relativi fattori di ridimensionamento in senso orizzontale e verticale. Questa tecnica presenta tuttavia lo svantaggio di provocare errori di arrotondamento quando i fattori di ridimensionamento sono "svantaggiosi". Così l'altezza del campo o l'interlinea potrebbero ad es. "saltare" di un pixel. Per impedire che questo accada esiste la modalità "Pixel Fine Adjustment", in cui le posizioni dei campo vengono determinate direttamente dalla risoluzione effettiva.
		`//M(MyMask/"My Mask"/////PA1)`
		Questa impostazione può essere impostata anche a livello globale per tutte le maschere Run MyScreens nel file "easyscreen.ini", ad es.
		`[GENERAL]`
		`DefaultPixelFineAdjustment=1`
		Se nel progettare la maschera si imposta in modo esplicito l'attributo PA, questo avrà una priorità maggiore dell'impostazione globale in "easyscreen.ini".
		Se una maschera viene rappresentata con Font Adjustment = FA1, per questa maschera sarà automaticamente attiva anche la modalità Pixel Fine Adjustment.
		(Vedere anche il capitolo SmartOperation e comando MultiTouch (Pagina 48))

	PA0	Ridimensionamento invariato, ossia le posizioni vengono calcolate su 640x480 e quindi vengono estese. (Per ragioni di compatibilità: default)
	PA1	Ridimensionamento ottimizzato dei campi rispettando il rapporto di pixel nel caso di risoluzioni superiori

FA		Impostazione dell'altezza del campo e dell'interlinea proporzionali al font (Font Adjustment)
		Questa caratteristica può essere impostata anche a livello globale per tutte le maschere Run MyScreens nel file "easyscreen.ini", ad es.
		`[GENERAL]`
		`DefaultFontAdjustment=1`
		Se nella maschera di progettazione è impostato esplicitamente l'attributo FA, quest'ultimo prevale sull'impostazione globale in "easyscreen.ini".
		Le impostazioni per DefaultLineHeight e DefaultLineSpacing sono ininfluenti quando è attiva la modalità Font Adjustment.

			(Vedere anche il capitolo SmartOperation e comando MultiTouch (Pagina 48))
		FA0	L'altezza del campo e l'interlinea vengono ridimensionati come in precedenza (per ragioni di compatibilità: default)
		FA1	L'altezza del campo e l'interlinea sono impostate proporzionalmente al font (imposta automaticamente l'attributo di maschera PA=1)
		FA2	Uguale a FA1, ma inoltre vengono ridimensionate proporzionalmente al font anche la coordinata X e la larghezza del campo. (imposta automaticamente l'attributo di maschera PA=1) (Possibile solo in associazione all'attributo XG=1!)
	NT		Tipo di navigazione (Navigation Type)
		NT0	NT0 = modo standard La navigazione avviene in modo analogo a quello delle maschere dell'area operativa in cui è integrata la maschera Run MyScreens. Ciò significa che nelle maschere di ciclo nell'editor la navigazione avviene orientata alle righe (vedere NT2), in tutte le altre maschere "normali" in modo geometrico (vedere NT1).
		NT1	Navigazione geometrica (in alto, in basso, a sinistra, a destra) fino a raggiungere i bordi della maschera (impostazione predefinita nell'ambiente Run MyScreens normale ad es. area "Custom")
		NT2	Orientata alle righe, ossia all'interno della riga verso destra finché viene raggiunta la fine della riga (Standard nelle maschere di ciclo dell'editor)
	NR		Direzione di navigazione alla pressione del tasto Invio (Return Navigate Direction)
		NR0	NR0 = off (impostazione predefinita), ossia premendo il tasto Invio la navigazione avviene all'interno della maschera in sequenza tabellare (impostazione standard nell'ambiente Run MyScreens normale)
		NR1	Premendo il tasto Invio la navigazione si comporta come alla pressione del tasto freccia su
		NR2	Come NR1, ma in basso
		NR3	Come NR1, ma a sinistra
		NR4	Come NR1, ma a destra
	TA		Orientamento del tooltip
		TA0	automatico (predefinito)
		TA1	a sinistra
		TA2	a destra
		TA3	in alto
		TA4	in basso
		TA5	in alto a sinistra
		TA6	in basso a sinistra
		TA7	in alto a destra
		TA8	in basso a destra

	MC		Colore di sfondo della maschera (Mask Color)
			Esempio:
			Impostazione colore di sfondo della maschera blu (= 6)
			`//M(MyMask/"MyMask"//////6)`
			oppure
			`PRESS(HS1)`

		MC=6 END_PRESS
	Lista dei file della lingua specifici per maschera	separati da una virgola

Accesso alle proprietà della finestra di dialogo

All'interno dei metodi (ad es. blocco PRESS) è possibile accedere alle seguenti proprietà della finestra di dialogo in lettura o in scrittura:

- HD = intestazione (Header)
- HLP = pagina di help
- VAR = variabile di sistema o utente
- MC = colore di sfondo della maschera
- CM = allineamento delle colonne (solo in lettura)
- CB = comportamento all'apertura (solo in lettura)
- XG = integrazione X3d (solo in lettura)
- AL = allineamento intestazione della maschera (solo in lettura)

Esempio

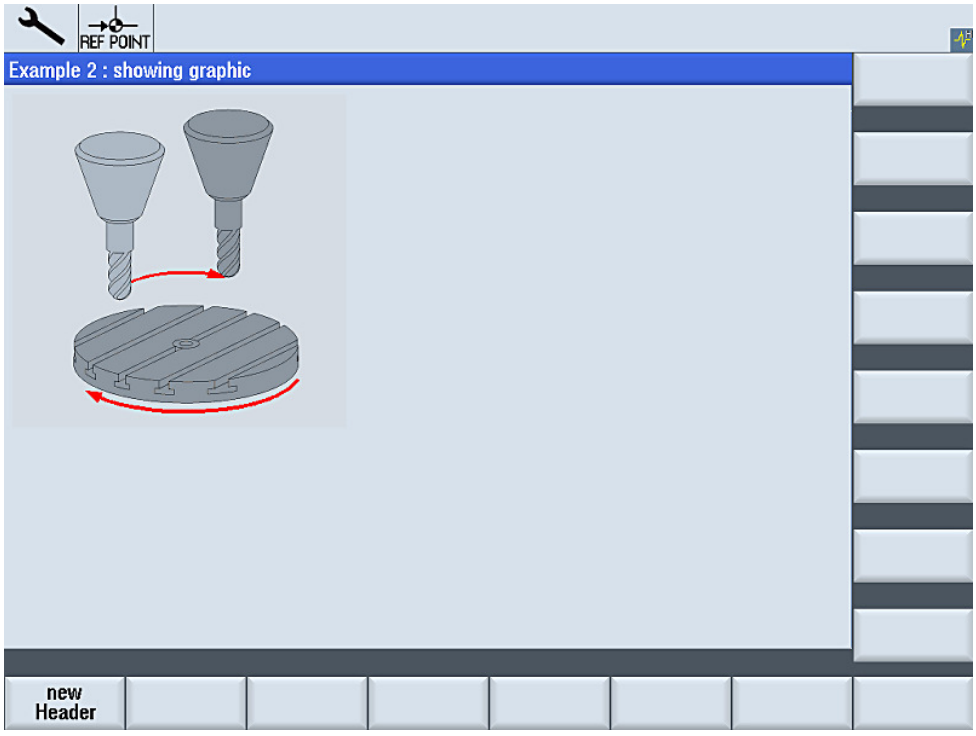


Figura 6-3 "Example 2: showing graphic"

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

PRESS(HS7)
  LM("Mask2")
END_PRESS

//END
//M(Mask2/"Example 2 : showing graphic"/"example.png")
HS1=("new%nHeader")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")

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

Vedere anche

Esempio di programmazione per il settore "Custom" (Pagina 278)

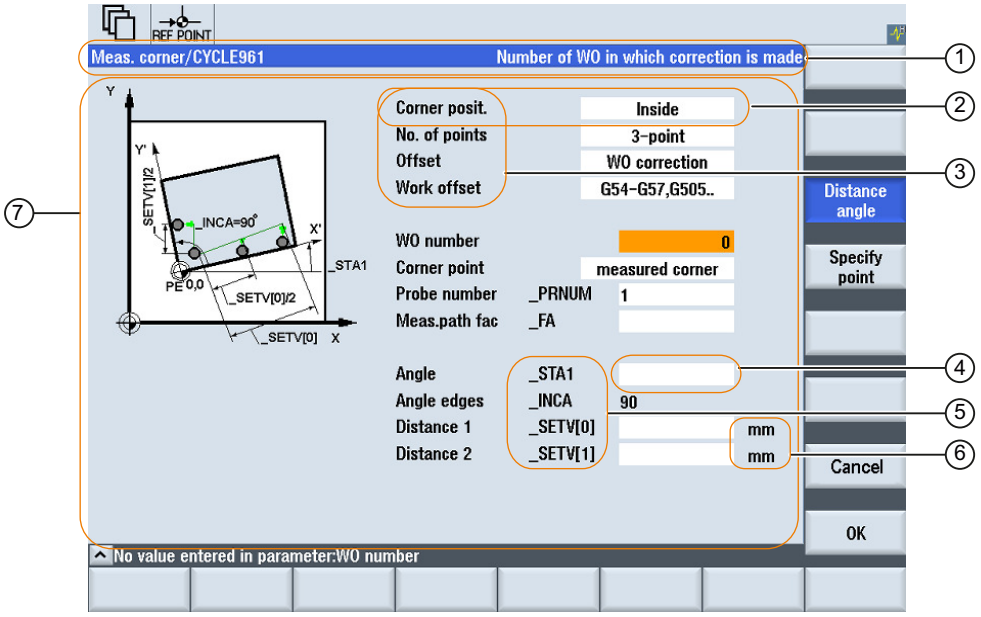
6.1.3 Definizione degli elementi delle finestre di dialogo

Elemento della finestra di dialogo

Con "elemento della finestra di dialogo" si intende la parte visibile di una variabile, ossia testo sintetico, testo grafico, campo di input/output e tooltip. Gli elementi della finestra di dialogo riempiono le righe nella parte principale della finestra di dialogo. Per ciascuna riga è possibile definire uno o più elementi della finestra di dialogo.

Proprietà delle variabili

Tutte le variabili sono valide solo nella finestra di dialogo attiva. Con la definizione di una variabile vengono assegnate queste proprietà. All'interno dei metodi (ad es. di un metodo PRESS) è possibile accedere ai valori delle proprietà delle finestre di dialogo.



① Riga di intestazione della finestra di dialogo, con titolo e testo completo
② Elemento della finestra di dialogo
③ Testo sintetico
④ Campo di input/output
⑤ Testo grafico
⑥ Testo unità
⑦ Parte principale della finestra di dialogo

Figura 6-4 Elementi di una finestra di dialogo

Panoramica della programmazione

Tra parentesi tonde sono riportati i singoli parametri da separare tramite virgola:

DEF [Identificatore] =	**Identificatore = nome della variabile**
	Tipo di variabile
	/[Valori limite/campo toggle]
	/[Preassegnazione]
	/[Testi(Testo completo, testo sintetico|Immagine, testo grafico, testo unità)]
	/[Attributi]
	/[Pagina di help]
	/[Variabile di sistema o utente]
	/[Posizione testo sintetico]
	/[Posizione campo di input/output(Left, Top, Width, Height)]
	/[Colori]
	/[Guida in linea] (Pagina 73)

Vedere anche

Parametri delle variabili (Pagina 91)

6.1.4 Definizione di finestra di dialogo a più colonne

Panoramica

All'interno di una finestra di dialogo è possibile rappresentare su una riga anche più variabili. Tutte le variabili in questo caso vengono definite nel file di progettazione nell'ambito di una sola riga di definizione.

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

Per poter rappresentare le singole variabili nel file di progettazione in modo più leggibile, le righe di definizione possono essere interrotte dopo ogni definizione di variabile e successiva virgola.

La parola chiave "DEF" caratterizza sempre l'inizio di una nuova riga:

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

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

Nota

Nel creare finestre di dialogo multilingue, considerare il fatto che un numero elevato di colonne può rallentare il sistema!

6.1.5 Finestre di dialogo per password

Utilizzo delle finestre di dialogo per password standard

Le seguenti finestre di dialogo predefinite per password possono essere inserite nella progettazione della maschera:

- Impostazione della password

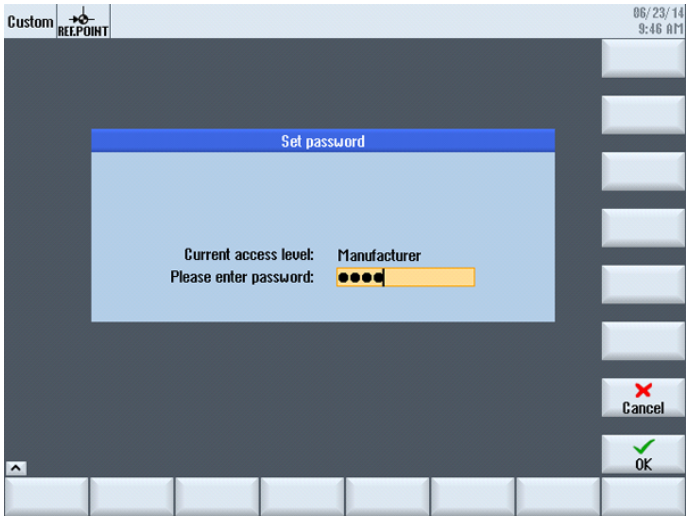


Figura 6-5 Finestra di dialogo Impostazione della password

- Modifica della password

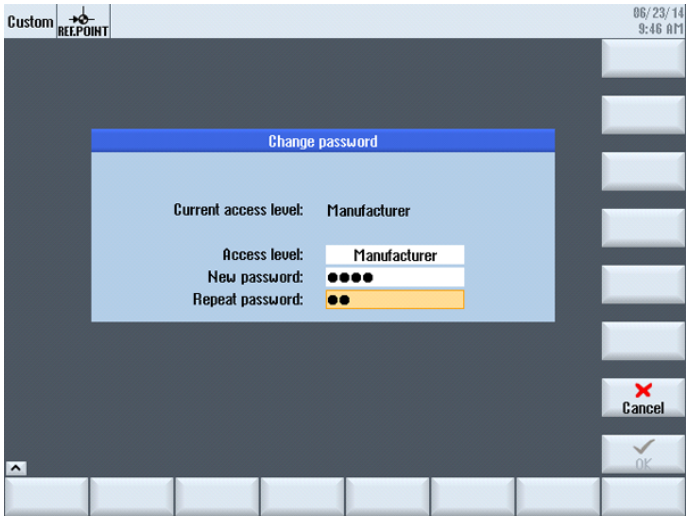


Figura 6-6 Dialogo "Modifica della password"

- Cancellazione della password

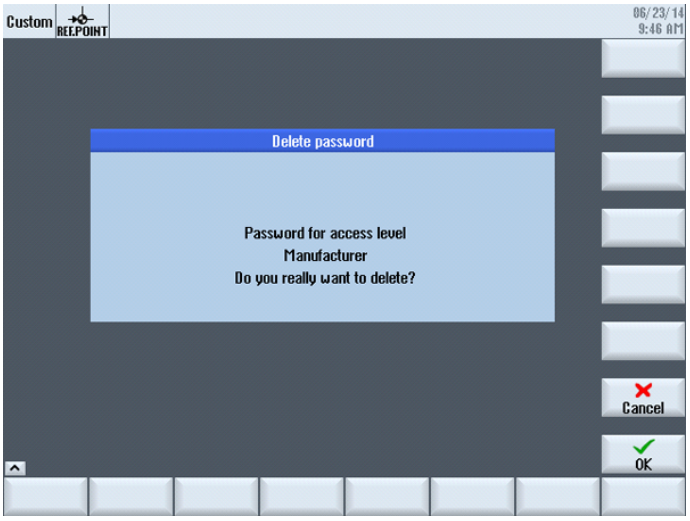


Figura 6-7 Finestra di dialogo Cancellazione della password

Nota

Mit diesen Dialogen wird Ihnen die Kennwort-Funktionalität bereitgestellt. Queste finestre di dialogo non corrispondono a quelle di SINUMERIK Operate.

Il richiamo delle finestre di dialogo può avvenire tramite la funzione Load Softkey (LS) o tramite la funzione Load Mask (LM):

1. Richiamo tramite la funzione Load Softkey (LS):
   `LS("Passwd","slespasswd.com",1)`
   Estensione dei softkey verticali:
   - Softkey 1: Impostazione della password
   - Softkey 2: Modifica della password
   - Softkey 3: Cancellazione della password
2. Alternativ können die drei Masken auch direkt durch den Aufruf der Funktion Load Mask (LM) angesprungen werden:
   - Impostazione della password: `LM("PWD_SET", "slespasswd.com",1)`
   - Modifica della password: `LM("PWD_CHG", "slespasswd.com",1)`
   - Cancellazione della password: `LM("PWD_CLEAR", "slespasswd.com",1)`

6.1.6 Utilizzo di immagini/grafica

Utilizzo della grafica

Si devono distinguere:

- Immagini/grafica nell'area della grafica
- Pagine di help, che ad esempio illustrano singole variabili e vengono visualizzate sull'area della grafica.
- Al posto del testo sintetico o del campo di input/output è possibile progettare altre pagine di help, posizionabili liberamente.

Luoghi di archiviazione

L'immagine adatta per la risoluzione del monitor collegato viene cercata in un primo tempo nella relativa directory della risoluzione:

[cartella di sistema user]`/ico/ico<risoluzione>`
[cartella di sistema oem]`/ico/ico<risoluzione>`
[cartella di sistema addon]`/ico/ico<risoluzione>`

Se l'immagine non viene visualizzata o trovata, copiarla in una delle seguenti directory per la risoluzione 640 x 480 pixel:

[cartella di sistema user]`/ico/ico640`
[cartella di sistema oem]`/ico/ico640`
[cartella di sistema addon]`/ico/ico640`

Nota

Con il variare della risoluzione del pannello, le immagini vengono riposizionate preservandone le proporzioni.

6.2 Definizione delle barre di softkey

Definizione

Tutti i softkey orizzontali e tutti i softkey verticali vengono complessivamente definiti come barra dei softkey. È possibile definire nuove barre dei softkey in aggiunta a quelle già esistenti, nonché sovrascrivere parzialmente o completamente quelle esistenti.

I nomi dei softkey sono predefiniti. I softkey non devono essere tutti riservati.

HSx x 1 - 8, Softkey orizzontali da 1 a 8

VSy y 1 - 8, Softkey verticali da 1 a 8

In linea di principio la definizione di una barra dei softkey (blocco di definizione) è strutturata come segue:

Blocco di descrizione	Commento	Rimando al capitolo
`//S...`	;Codice iniziale barra dei softkey	
`HSx=...`	;Definizione softkey	

`PRESS(HSx)`
`  LM...`
`END_PRESS`
	;Codice iniziale metodo
	;Azioni
	;Codice finale metodo
		Vedere il capitolo Metodi (Pagina 119)

`//END`	;Codice finale barra dei softkey	

Descrizione

Attraverso la definizione della barra dei softkey vengono assegnate contemporaneamente proprietà a un softkey.

Programmazione

Sintassi:	//S*(Identificatore)*
	...
	//END
		;Codice iniziale della barra dei softkey
		
		;Codice finale della barra dei softkey

	Vedere anche il capitolo Sintassi di progettazione estesa (Pagina 46).	
Descrizione:	Definizione della barra dei softkey	
Parametri:	Identificatore	Nome della barra dei softkey
	Testo o nome del file di immagine	
Sintassi:	**SK** = (testo[, livello di accesso][, stato])[, allineamento dell'immagine del softkey][, allineamento testo riferito all'immagine del softkey])	
Descrizione:	Definizione dei softkey	
Parametri:	SK	Softkey, ad es. da HS1 a HS8, da VS1 a VS8

	Testo	Indicazione del testo	
	Nome del file immagine	"\\my_pic.png" oppure tramite file di testo separato $85199 ad es. con il testo seguente nel file di testo (dipendente dalla lingua): 85100 0 0 "\\my_pic.png". La dimensione dell'immagine per la rappresentazione su un softkey dipende dall'OP utilizzato:	
		OP 08:	640 x 480 mm → 25 x 25 pixel
		OP 010:	640 x 480 mm → 25 x 25 pixel
		OP 012:	800 x 600 mm → 30 x 30 pixel
		OP 015:	1024 x 768 mm → 40 x 40 pixel
		OP 019:	1280 x 1024 mm → 72 x 72 pixel
	Livello di accesso	da ac0 a ac7 (ac7: Preimpostazione)	
	stato	se1: visibile (preimpostazione) se2: non azionabile (carattere grigio) se3: evidenziato (ultimo softkey premuto)	
	Allineamento dell'immagine del softkey	PA (PictureAlignment)	
		Valori validi: 0: a sinistra 1: a destra 2: centrato 3: in alto (standard) 4: in basso	
	Allineamento testo riferito all'immagine del softkey	TP (TextAlignedToPicture)	
		Valori validi: 0: il testo non viene allineato all'immagine 1: il testo viene allineato all'immagine (standard)	

Nota

Nella dicitura dei softkey l'interruzione di riga viene realizzata attraverso il segno %.

Sono a disposizione al massimo 2 righe di 9 caratteri ciascuna.

Assegnazione del livello di accesso

L'operatore ha accesso solo alle informazioni corrispondenti a questo livello di accesso e a quelle dei livelli di accesso inferiori (vedere anche AUTOHOTSPOT).

Esempio

```
//S(Barra1)                                    ; Codice iniziale della barra dei softkey
```

```
HS1=("NEU", ac6, se2)                          ; Definizione del softkey HS1, assegnazione della de-
                                                finizione "NUOVO", del grado di protezione 6 e dello
                                                stato "non azionabile"
HS2=("\\bild1.png")                            ; Assegnazione di una grafica al softkey
HS3=("Exit")
HS4=(["Confirm","\\sk_ok.png"],PA0,TP1)        ; Softkey con testo e grafica, testo="Confirm", imma-
                                                gine="sk_ok.png", allineamento dell'immagine del soft-
                                                key: a sinistra, il testo viene allineato all'immagine

VS1=("Sottomaschera")
VS2=($85011, ac7, se2)                         ; Definizione del softkey VS2, assegnazione del testo
                                                dal file della lingua, del livello di protezione 1 e
                                                dello stato "non azionabile".
VS3=("Interruzione", ac1, se3)                 ; Definizione del softkey VS3, assegnazione della de-
                                                finizione "Interruzione", del grado di protezione 1 e
                                                dello stato "evidenziato".
VS4=("OK", ac6, se1)                           ; Definizione del softkey VS4, assegnazione della de-
                                                finizione "OK", del grado di protezione 6 e dello sta-
                                                to "visibile"
VS5=(SOFTKEY_CANCEL,,se1)                      ; Definizione del softkey standard VS5 "Interruzione"
                                                e assegnazione dello stato "visibile"
VS6=(SOFTKEY_OK,,se1)                          ; Definizione del softkey standard VS6 "OK" e assegna-
                                                zione dello stato "visibile"
VS7=(["\\bild1.png","Testo OEM"],,se1)         ; Definizione del softkey VS7, assegnazione di una gra-
                                                fica, assegnazione, assegnazione della definizione "Te-
                                                sto OEM" e dello stato "visibile"
VS8=(["\\bild1.png",$83533],,se1)              ; Definizione del softkey VS8, assegnazione di una gra-
                                                fica, assegnazione del testo dal file della lingua e
                                                dello stato "visibile"

PRESS(HS1)                                     ; Codice iniziale del metodo
  HS1.st="Calcolo"                             ; Assegnazione di un testo di definizione al softkey
...
END_PRESS                                      ; Codice finale del metodo

PRESS(RECALL)                                  ; Codice iniziale del metodo
  LM("Maschera21")                             ; Caricamento della finestra di dialogo
END_PRESS                                      ; Codice finale del metodo
//END                                          ; Codice finale della barra dei softkey
```

6.2.1 Modifica delle proprietà di softkey durante il tempo di esecuzione

Descrizione

Le proprietà testo, livello di accesso e stato di un softkey possono essere modificate nei metodi durante il tempo di esecuzione.

Programmazione

Sintassi:	SK.st = "Testo" SK.ac = Livello di accesso SK.se = Stato SK.pa = "Allineamento dell'immagine del softkey" SK.tp = "Allineamento testo riferito all'immagine del softkey"		;Softkey con definizione ;Softkey con grado di protezione ;Softkey con stato ;Softkey con immagine ;Softkey con immagine e scritta
Descrizione:	Assegnazione di proprietà		
Parametri:	Testo		Testo di definizione tra virgolette
	Livello di accesso		Campo dei valori: 0 ... 7
	stato	1: 2: 3:	visibile e utilizzabile non utilizzabile (carattere grigio) evidenziato (ultimo softkey premuto)
	Allineamento dell'immagine del softkey	0: 1: 2: 3: 4:	a sinistra a destra centrato in alto (standard) in basso
	Allineamento testo riferito all'immagine del softkey	0: 1:	il testo non viene allineato all'immagine il testo viene allineato all'immagine (standard)

Esempio

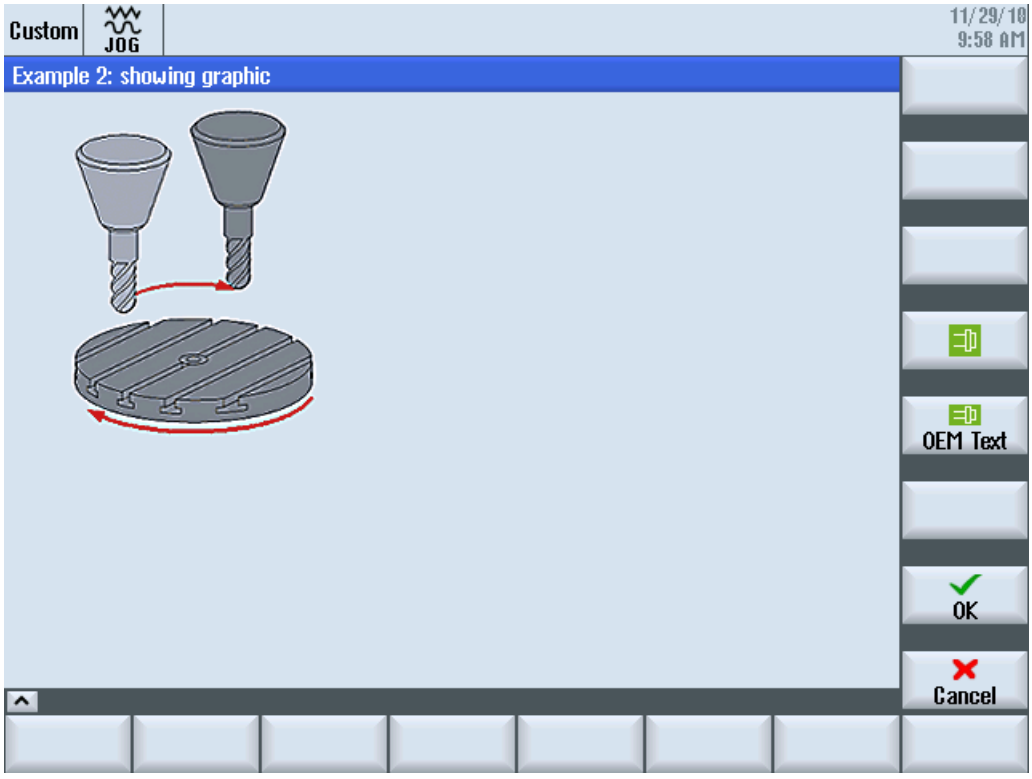


Figura 6-8 Esempio 3: Grafica e softkey

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

PRESS(HS7)
  LM("Maske3")
END_PRESS

//END

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

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

6.2.2 Testo dipendente dalla lingua

Panoramica

I testi dipendenti dalla lingua vengono utilizzati per:

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

I testi dipendenti dalla lingua per le finestre di dialogo sono memorizzati in file di testo.

I file di testo si trovano nelle seguenti directory:

- [cartella di sistema user]/lng
- [cartella di sistema oem]/lng
- [cartella di sistema addon]/lng

Nota

I file di testo devono essere memorizzati analogamente ai file di progetto.

Ad es.:

[cartella di sistema user]/lng/[file di testo]

[cartella di sistema user]/proj/[file di progettazione]

alsc.txt testi dipendenti dalla lingua per i cicli standard Siemens
almc.txt testi dipendenti dalla lingua per i cicli di misura Siemens

I file di testo utilizzati durante il tempo di esecuzione sono specificati nel file "easyscreen.ini":

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

Il file "user.txt" funge qui da esempio di file di testo. Generalmente il nome può essere selezionato a piacere. A seconda della lingua dei testi contenuti nel file, occorre aggiungere l'abbreviazione adatta in base alla sintassi seguente. Dopo il nome viene aggiunto un carattere di sottolineatura e quindi il codice della lingua appropriato, ad es. "user_eng.txt".

Nota

LngFile01 e LngFile02 non vanno usati per i testi utente, dato che questi sono assegnati al file "easyscreen.ini" standard.

Vedere anche

AUTOHOTSPOT

File della lingua specifici per maschera

Per evitare sovrapposizioni degli intervalli numerici nei file della lingua, conviene impiegare in una stessa maschera solo determinati file della lingua.

Nella riga di definizione della maschera i file della lingua da utilizzare sono pertanto separati da virgole. La priorità maggiore spetta al file elencato per ultimo (analogamente alla logica in "easyscreen.ini").

Tutti i testi dipendenti dalla lingua utilizzati nella maschera vengono cercati dapprima in questi file della lingua. Se il testo non viene trovato in questi file, la ricerca viene estesa ai file di testo progettati in "easyscreen.ini".

Se nella maschera non sono definiti dei file della lingua, la ricerca avviene solo in quelli specificati in "easyscreen.ini".

Nota

Se un file della lingua non esiste nella lingua attualmente selezionata, verrà impiegato il file della lingua inglese (default).

Esempi:

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

Questa procedura vale anche per i softkey di accesso:

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

Formato dei file di testo

I file di testo devono essere salvati nel formato codifica UTF-8.

Nota

Per il salvataggio dei file di progettazione e della lingua assicurarsi che nell'editor utilizzato la codifica sia impostata su UTF-8.

Formato di una riga di testo

Sintassi:	8xxxx 0 0 "Testo"		
Descrizione:	Assegnazione tra numero di testo e testo nel file		
Parametri:	xxxx	85.000 ... 89.999 900.000 ... 999.999	Settore dei numeri di identificazione dei testi riservato all'utente. I numeri devono essere sempre definiti in modo univoco.
	"Testo"		Testo visualizzato nella finestra di dialogo
	%n		Carattere di controllo nel testo per l'interruzione di riga

I parametri 2 e 3 separati da spazi sono caratteri di controllo per l'emissione del testo dell'allarme. Essi devono essere comunque impostati a zero per mantenere un'uniformità del formato del testo con i testi di allarme.

Esempi di testi dipendenti dalla lingua:

```
85000 0 0 "Piano di svincolo"
85001 0 0 "Profondità di foratura"
85002 0 0 "Passo del filetto"
85003 0 0 "Raggio della tasca"
```

6.3 Progettazione della guida in linea

6.3.1 Panoramica

Oltre alla completa guida in linea già esistente, è possibile creare una guida in linea specifica per il costruttore e integrarla in SINUMERIK Operate.

Questa guida in linea verrà creata in formato HTML, ossia si comporrà di documenti HTML collegati tra loro. L'argomento ricercato viene richiamato in un'apposita finestra attraverso un indice del contenuto o un indice analitico. Analogamente a un browser di documenti (ad es. Esplora risorse di Windows), nella metà sinistra della finestra viene mostrata una panoramica di selezione; facendo clic sull'argomento desiderato, nella metà destra della finestra viene visualizzata la relativa spiegazione.

Non è possibile una selezione contestuale delle pagine della Guida in linea.

Procedura

1. Creazione di file HTML
2. Creazione di un registro della guida
3. Integrazione della Guida in linea in SINUMERIK Operate
4. Archiviazione dei file della guida

Altri casi applicativi

È possibile creare Guide in linea relative ai seguenti ampliamenti specifici per OEM e integrarle nel sistema di Guida in linea di SINUMERIK Operate:

- Guida in linea per **Cicli e /o funzioni M** del costruttore della macchina che ampliano le possibilità di programmazione dei controllori SINUMERIK. Questa Guida in linea viene richiamata esattamente come la Guida on linea di SINUMERIK Operate "Programmazione".
- Guida in linea relativa a **Variabili** specifiche per OEM del costruttore della macchina. Questa Guida in linea viene richiamata dalla vista delle variabili di SINUMERIK Operate.

Programmazione della Guida in linea

Per le altre possibilità di creazione della guida in linea è possibile utilizzare il "Pacchetto di programmazione SINUMERIK HMI sl". Questo pacchetto di programmazione consente di sviluppare applicazioni in linguaggio evoluto utilizzando il linguaggio di programmazione C++ per SINUMERIK Operate sulla NCU 7x0.

Nota

Il "Pacchetto di programmazione SINUMERIK HMI sl" è un'opzione software da ordinare separatamente. La relativa documentazione viene fornita unitamente al pacchetto di programmazione.

6.3.2 Creazione di file HTML

Creare i file della guida in formato HTML. È possibile archiviare tutte le informazioni in un unico file HTML oppure separarle in più file HTML.

I nomi dei file possono essere scelti a piacere, tenendo però presente quanto segue:

- I riferimenti all'interno dei file HTML vanno sempre indicati con i relativi percorsi. Solo così si garantisce che i riferimenti funzionino in modo uniforme sia sul PC di sviluppo sia sul sistema di destinazione.
- Se all'interno di un file HTML si desidera spostarsi su punti particolari tramite link, è necessario definire allo scopo i cosiddetti anchor.
  Esempio di un anchor HTML:
  `<a name="myAnchor">This is an anchor</a>`
- Il contenuto dei documenti HTML deve essere archiviato con la codifica UTF-8. In tal modo si garantisce che i documenti HTML vengano visualizzati correttamente in tutte le lingue supportate da SINUMERIK Operate.
- Sono supportati i seguenti sottoinsiemi del pacchetto di funzionalità HTML:

Tag HTML

Tag	Descrizione	Commento
a	Anchor or link	Attributi supportati: href e name
address	Address	
big	Larger font	
blockquote	Indented paragraph	
body	Document body	Attributi supportati: bgcolor (#RRGGBB)
br	Line break	
center	Centered paragraph	
cite	Inline citation	Stessa funzione del tag i
code	Code	Stessa funzione del tag tt
dd	Definition data	
dfn	Definition	Stessa funzione del tag i

Tag	Descrizione	Commento
div	Document division	Vengono supportati gli attributi blocco standard
dl	Definition list	Vengono supportati gli attributi blocco standard
dt	Definition term	Vengono supportati gli attributi blocco standard
em	Emphasized	Stessa funzione del tag i
font	Font size, family, color	Attributi supportati: size, face, and color (#RRGGBB)
h1	Level 1 heading	Vengono supportati gli attributi blocco standard
h2	Level 2 heading	Vengono supportati gli attributi blocco standard
h3	Level 3 heading	Vengono supportati gli attributi blocco standard
h4	Level 4 heading	Vengono supportati gli attributi blocco standard
h5	Level 5 heading	Vengono supportati gli attributi blocco standard
h6	Level 6 heading	Vengono supportati gli attributi blocco standard
head	Document header	
hr	Horizontal line	Attributi supportati: width (può essere indicata come valore assoluto o relativo)
html	HTML document	
i	Italic	
img	Image	Attributi supportati: src, width, height
kbd	User-entered text	
meta	Meta-information	
li	List item	
nobr	Non-breakable text	
ol	Ordered list	Vengono supportati gli attributi standard per le liste
p	Paragraph	Vengono supportati gli attributi blocco standard (impostazione predefinita: left-aligned)
pre	Preformated text	
s	Strikethrough	
samp	Sample code	Stessa funzione del tag tt
small	Small font	
span	Grouped elements	
strong	Strong	Il testo continuo viene messo in risalto, sostituisce il tag b
sub	Subscript	
sup	Superscript	
table	Table	Attributi supportati: border, bgcolor (#RRGGBB), cellspacing, cellpadding, width (assoluta o relativa), height
tbody	Table body	Senza funzione
td	Table data cell	Vengono supportati gli attributi standard per le celle delle tabelle
tfoot	Table footer	Senza funzione
th	Table header cell	Vengono supportati gli attributi standard per le celle delle tabelle
thead	Table header	Viene utilizzato per la stampa di tabelle che si estendono su più pagine
title	Document title	
tr	Table row	Attributi supportati: bgcolor (#RRGGBB)

Tag	Descrizione	Commento
tt	Typewrite font	
u	Underlined	
ul	Unordered list	Vengono supportati gli attributi standard per le liste
var	Variabile	Stessa funzione del tag tt

Attributi blocco

I seguenti attributi vengono supportati dai tag div, dl, dt, h1, h2, h3, h4, h5, h6, p:

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

Attributi standard per liste

I seguenti attributi vengono supportati dai tag ol e ul:

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

Attributi standard per tabelle

I seguenti attributi vengono supportati dai tag td e th:

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

Proprietà CSS

La tabella che segue contiene le funzionalità CSS supportate:

Proprietà	Valori	Descrizione
background-color	<color>	Colore dello sfondo degli elementi
background-image	<uri>	Immagine di sfondo degli elementi
color	<color>	Colore in primo piano per il testo
text-indent	<length>px	Rientro della prima riga di un paragrafo in pixel
white-space	normal | pre | nowrap | pre-wrap	Definisce come trattare un carattere "white space" nei documenti HTML.
margin-top	<length>px	Larghezza del margine superiore in pixel
margin-bottom	<length>px	Larghezza del margine inferiore in pixel
margin-left	<length>px	Larghezza del margine sinistro in pixel
margin-right	<length>px	Larghezza del margine destro in pixel

Proprietà	Valori	Descrizione
vertical-align	baseline | sub | super | middle | top | bottom	Orientamento verticale del testo (nelle tabelle sono supportati soltanto i valori middle, top e bottom)
border-color	<color>	Colore dei bordi delle tabelle di testo
border-style	none | dotted | dashed | dot-dash | dot-dot-dash | solid | double | groove | ridge | inset | outset	Stile dei bordi per tabelle di testo
background	[<'background-color'> || <'background-image'>]	Notazione abbreviata per background Property
page-break-before	[auto | always]	Interruzione di pagina prima di un paragrafo/una tabella
page-break-after	[auto | always]	Interruzione di pagina dopo un paragrafo/una tabella
background-image	<uri>	Immagine di sfondo degli elementi

Selettori CSS supportati

Sono supportati tutti i selettori di classe CSS 2.1, ad eccezione delle cosiddette pseudo-classi, quali :first-child, :visited e :hover.

6.3.3 Creazione di un registro della guida

Il registro della guida è un file XML nel quale è definita la struttura della Guida in linea. In questo file vengono definiti:

- i documenti HTML
- l'indice del contenuto e l'indice analitico

Sintassi del registro della guida

Tag	Numero	Significato	
HMI_SL_HELP	1	Elemento Root del documento XML	
I-BOOK I I I I I I I I I I	+	Indica un registro della guida. Il nome è liberamente selezionabile a condizione che si utilizzi un nome predefinito dal sistema (come ad es. sinumerik_alarm_plc_pmc). Nell'esempio il nome del registro della guida è il seguente: "hmi_myhelp" Attributi:	
		ref	Definisce il documento HTML visualizzato come pagina di accesso al registro della guida.
		titel	Titolo del registro della guida visualizzato nell'indice del contenuto.
		helpdir	Indice che contiene la Guida in linea del registro della guida.

Tag	Numero	Significato	
I-ENTRY II II II II II II II	*	Capitolo della Guida in linea Attributi:	
		ref	Definisce il documento HTML visualizzato come pagina di accesso al capitolo.
		titel	Titolo del capitolo visualizzato nell'indice del contenuto.
II-INDEX_ENTRY II II II II II II II	*	Parola chiave da visualizzare Attributi:	
		ref	Definisce il documento HTML che verrà richiamato per questa parola chiave.
		titel	Titolo della parola chiave visualizzata nell'indice analitico.

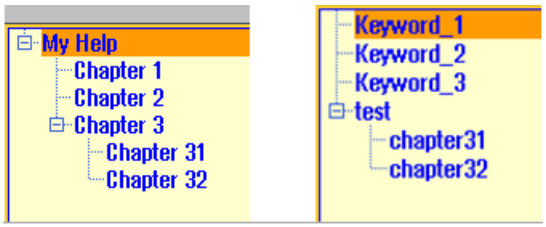
Per la colonna "Numero" vale quanto segue:
* significa 0 o più
+ significa 1 o più

Esempio di un registro della guida:

L'esempio che segue descrive la struttura di un registro della guida denominato "My Help". Inoltre, questo costituisce la base per l'indice del contenuto e l'indice analitico.

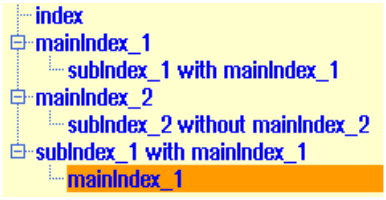
```
<?xml version="1.0" encoding="utf-8"?>
<HMI_SL_HELP language="en-US">
  <BOOK ref="index.html" title="My Help" helpdir="hmi_myhelp">
    <ENTRY ref="chapter_1.html" title="Chapter 1">
      <INDEX_ENTRY ref="chapter_1html#Keyword_1" title="Keyword_1"/>
      <INDEX_ENTRY ref="chapter_1.html#Keyword_2"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

Il registro si compone di tre capitoli, il terzo dei quali è suddiviso in due sottocapitoli. Le diverse parole chiave sono di volta in volta definite all'interno del capitolo.



Esistono tre possibilità di formattazione dell'indice analitico:

1. Voce singola:
   `<INDEX_ENTRY ...title="index"/>`
2. Due voci a due stadi, dove ciascun titolo dispone di una voce principale e di una voce secondaria. Separare le voci tra loro con una virgola.
   `<INDEX_ENTRY ...title="mainIndex_1,subIndex_1 with mainIndex_1"/>`
3. Voce a due stadi, dove il primo titolo è la voce principale e il secondo titolo la voce secondaria. Separare le voci tra loro con un punto e virgola.
   `<INDEX_ENTRY ...title="mainIndex_2;subIndex_2 without mainIndex_1"/>`



6.3.4 Integrazione della Guida in linea in SINUMERIK Operate

Per integrare il registro della guida creato nel sistema della Guida in linea di SINUMERIK Operate, è necessario il file "slhlp.xml".

Descrizione del formato di "slhlp.xml"

Tag	Numero	Significato
CONFIGURATION	1	Elemento Root del documento XML. Indica che si tratta di un file di configurazione.
I-OnlineHelpFiles	1	Introduce la sezione dei registri della Guida in linea.
II-<help_book>	*	Introduce la sezione di un registro della guida.

Tag	Numero	Significato	
III-EntriesFile III III III III III	1	Nome file del registro della guida contenente le voci dell'indice del contenuto e analitico. Attributi:	
		value	Nome del file XML
		type	Tipo di dati del valore (QString)
III-Technology III III III III III III III III III IIII	0, 1	Indica la tecnologia per la quale si applica il registro della guida. "All" vale quindi per tutte le tecnologie. Se il registro della guida è valido per più tecnologie, tali tecnologie vengono indicate separate da una virgola. Valori possibili: All, Universal, Milling, Turning, Grinding, Stroking, Punching Attributi:	
		value	Indicazione della tecnologia
		type	Tipo di dati del valore (QString)
III -DisableSearch III III III III III	0, 1	Disattivazione della ricerca per parola chiave per il registro della guida. Attributi:	
		value	true, false
		type	type Tipo di dati del valore (bool)
III-DisableFullTextSearch III III III III	0, 1	Disattivazione della ricerca a tutto testo per il registro della guida. Attributi:	
		value	true, false
		type	type Tipo di dati del valore (bool)
III-DisableIndex III III III III	0, 1	Disattivazione dell'indice analitico per il registro della guida. Attributi:	
		value	true, false
		type	type Tipo di dati del valore (bool)
III-DisableContent III III III III	0, 1	Disattivazione dell'indice del contenuto per il registro della guida. Attributi:	
		value	true, false
		type	type Tipo di dati del valore (bool)
III-DefaultLanguage III III III III III	0, 1	Sigla della lingua da visualizzare se la lingua corrente è disponibile per il registro della guida. Attributi:	
		value	chs, deu, eng, esp, fra, ita, ...
		type	Tipo di dati del valore (QString)

Per la colonna "Numero" vale quanto segue:
* significa 0 o più

Esempio di un file "slhlp.xml"

Nell'esempio che segue il registro della guida "hmi_myhelp.xml" viene reso disponibile in SINUMERIK Operate.

L'indice analitico non è attivato per il registro della guida.

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

6.3.5 Archiviazione dei file della guida

Archiviazione dei file della guida nel sistema di destinazione

1. Aprire la directory /**oem**/sinumerik/hmi/hlp e creare una nuova cartella per la lingua desiderata. Utilizzare allo scopo il codice della lingua predefinito.
   I nomi delle cartelle devono essere assolutamente scritti in caratteri minuscoli.
   Se ad es. si inserisce una guida per le lingue tedesco e inglese, occorre creare le cartelle "deu" e "eng".
2. Creare il registro della guida, ad es. "hmi_myhelp.xml", nelle cartelle "deu" e "eng" rispettivamente.
3. Copiare i file della guida nelle directory, ad es. /**oem**/sinumerik/hmi/hlp/deu/hmi_myhelp per i file della guida in tedesco e /**oem**/sinumerik/hmi/hlp/eng/hmi_myhelp per quelli in inglese.
4. Copiare il file di configurazione "slhlp.xml" nella directory /**oem**/sinumerik/hmi/cfg.
5. Riavviare HMI.

Nota

Quando si visualizzano l'indice dei contenuti e l'indice analitico di un registro della guida, vengono memorizzati per una rapida elaborazione nella directory /siemens/sinumerik/sys_cache/hmi/hlp i file della guida in formato binario (slhlp_<Hilfe-Buch_*.hmi). Se si modifica il registro della guida, occorre sempre cancellare questi file.

6.3.6 File della guida in formato PDF

Oltre ai file della Guida in formato HTML, è anche possibile integrare informazioni in formato PDF nel software operativo. Grazie ai collegamenti, dal sommario o dall'indice analitico si possono aprire le singole guide in formato PDF, accessibili direttamente anche dai file HTML.

Archiviazione delle guide PDF

Copiare le guide PDF in una delle seguenti directory:

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

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

Integrazione delle guide PDF

I file con estensione "pdf" vanno integrati nella progettazione delle finestre di dialogo o nelle progettazioni di sommari e indici analitici esattamente come i file con estensione "html":

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

Puntare tramite un link dai file HTML alla guida PDF:

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

Nota

Nella guida PDF non è possibile selezionare etichette di salto sensibili al contesto né saltare ad altri file HTML o PDF.

La funzione di ricerca è supportata solo all'interno di un file PDF. Non è supportata neppure la ricerca sovraordinata su più guide PDF.

Variabili 7

7.1 Definizione delle variabili

Valore della variabile

Proprietà principale di una variabile è il valore della variabile.

Il valore delle variabili può essere assegnato attraverso:

- la preassegnazione durante la definizione delle variabili
- l'assegnazione a una variabile di sistema o utente
- un metodo

Programmazione

Sintassi:	Identificatore.**val** = Valore della variabile Identificatore = Valore della variabile	
Descrizione:	Valore della variabile val (value)	
Parametri:	Identificatore:	Nome della variabile
	Valore della variabile:	Valore della variabile.
Esempio:	`VAR3 = VAR4 + SIN(VAR5)` `VAR3.VAL = VAR4 + SIN(VAR5)`	

Stato della variabile

Con la proprietà Stato della variabile è possibile effettuare un'interrogazione durante il tempo di esecuzione per verificare se una variabile contiene un valore valido. Questa proprietà può essere letta e scritta con il valore FALSE = 0.

Programmazione

Sintassi:	Identificatore.**vld**	
Descrizione:	Stato della variabile vld (validation)	
Parametri:	Identificatore:	Nome della variabile
		Il risultato dell'interrogazione può essere:
	FALSE =	un valore non valido
	TRUE =	un valore valido

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

7.2 Esempi applicativi

Variabili ausiliarie

Le variabili ausiliarie sono variabili di calcolo interne. Le variabili di calcolo vengono definite come variabili, ma non possiedono altre proprietà oltre al valore della variabile e allo stato; ciò significa che le variabili ausiliarie non sono visibili nella finestra di dialogo. Le variabili ausiliarie sono di tipo VARIANT.

Programmazione

Sintassi:	DEF[Identificatore]	
Descrizione:	Variabili di calcolo interne di tipo VARIANT	
Parametri:	Identificatore:	Nome della variabile ausiliaria
Esempio:	`DEF OTTO ;Definizione di una variabile ausiliaria`	

Sintassi:	Identificatore.val = valore della variabile ausiliaria Identificatore = valore della variabile ausiliaria	
Descrizione:	Il valore di una variabile ausiliaria viene assegnato in un metodo.	
Parametri:	Identificatore:	Nome della variabile ausiliaria
	Valore della variabile ausiliaria:	Contenuto della variabile ausiliaria

Esempio:

```
LOAD
  OTTO = "Test"             ; Assegna alla variabile ausiliaria Otto il valore
END_LOAD                    "Test"
LOAD
  OTTO = REG[9].VAL         ; Assegna alla variabile ausiliaria Otto il valore del
END_LOAD                    registro
```

Calcolo con variabili

Le variabili vengono calcolate ogni volta che si esce da un campo di input/output (tramite il tasto ENTER o di toggle). Il calcolo viene progettato in un metodo CHANGE ed eseguito ad ogni variazione del valore.

Per verificare se una variabile ha un valore valido, è possibile effettuare un'interrogazione sullo stato della variabile, ad es.:

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

Indirizzamento indiretto delle variabili di sistema

Una variabile di sistema può essere indirizzata anche in maniera indiretta, vale a dire in maniera dipendente da un'altra variabile:

```
PRESS(HS1)
  ACHSE=ACHSE+1
  WEG.VAR="$AA_DTBW["<<ACHSE<<"]"       ;Indirizzamento dell'indirizzo asse tramite
                                        variabile
END_PRESS
```

Modifica della dicitura del softkey

Esempio:

```
HS3.st = "Nuovo testo"                  ;Modifica della dicitura del softkey
```

7.3 Esempio 1: Assegnazione di tipo di variabile, testi, pagina di help, colori, Tooltips

Esempio 1a

L'esempio seguente definisce una variabile per cui è possibile impostare le proprietà tipo di variabile, testi, pagina di help e colori.

DEF Var1 = (R///,"Valore reale",,"mm"//"Var1.png"////8,2)			
	Tipo di variabile:		REAL
	Testi:		
		Testo sintetico:	valore reale
		Testo unità:	mm
	Pagina di help:		Var1.png
	Colori:		
		Colore di primo piano:	8 (marrone)
		Colore di sfondo:	2 (arancione)

Esempio 1b

L'esempio seguente definisce una variabile per cui è possibile impostare le proprietà tipo di variabile, preassegnazione, testi, descrizione comandi, modalità di immissione e posizione testo sintetico.

DEF Var2 = (I//5/"","valore","","," testo descrizione comandi"/wr2///20,250,50)			
	Tipo di variabile:		INTEGER
	Preassegnazione:		5
	Testi:		
		Testo sintetico:	Valore (possibile ID del testo della lingua)
		Descrizione comandi:	TooltipText
	Attributi:		
		Modalità di immissione	lettura e scrittura
	Posizione testo sintetico:		
		Distanza da sinistra	20
		Distanza dall'alto	250
		Larghezza:	50

Vedere anche

Parametri delle variabili (Pagina 91)

7.4 Esempio 2: Assegnazione di tipo di variabile, valori limite, attributi, posizione del testo sintetico

Esempio 2

L'esempio seguente definisce una variabile per cui è possibile impostare le proprietà tipo di variabile, valori limite, modalità di immissione, allineamento e posizione.

DEF Var2 = (I/0,10///wr1,al1/// , ,300)			
	Tipo di variabile:		INTEGER
	Valori limite o immissioni nel campo di toggle:		MIN: 0 MAX: 10
	Attributi:		
		Modalità di immissione	Sola lettura
		Allineamento testo sintetico	a destra
	Posizione testo sintetico:		
		Larghezza:	300

Vedere anche

Parametri delle variabili (Pagina 91)

7.5 Esempio 3: Assegnazione di tipo di variabile, preassegnazione, variabile di sistema o utente, posizione campo di input/output

Esempio 3

L'esempio seguente definisce una variabile per cui è possibile impostare le proprietà tipo di variabile, preassegnazione, variabile di sistema o variabile utente, posizione.

DEF Var3 = (R//10////"$R[1]"//300,10,200//)			
	Tipo di variabile:		REAL
	Preassegnazione:		10
	Variabile di sistema o utente:		$R[1] (Parametro R 1)
	Posizione testo sintetico:		Posizione standard rispetto al campo di input/output
	Posizione campo di input/output:		
		Distanza da sinistra	300
		Distanza dall'alto	10
		Larghezza:	200

Vedere anche

Parametri delle variabili (Pagina 91)

7.6 Esempio 4: Campo di toggle e casella di riepilogo

Esempio 4a

Diverse immissioni nel campo toggle:

```
DEF Var1 = (I/* 0,1,2,3)                ;Campo toggle semplice per attivare/disattivare i valori numerici
DEF Var2 = (S/* "On", "Off")            ;Campo toggle semplice per attivare/disattivare le stringhe
DEF Var3 = (B/* 1="On", 0="Off")        ;Campo toggle esteso per attivare/disattivare i valori numerici; a ciascun valore numerico è assegnato un testo di visualizzazione
DEF Var4 = (R/* ARR1)                   ;Campo toggle che acquisisce i suoi valori da attivare/disattivare traendoli da un array
```

Esempio 4b

La casella di riepilogo corrisponde alla progettazione di un campo toggle; in aggiunta deve anche essere impostato il tipo di visualizzazione per la casella di riepilogo (attributo di variabile DT = 4).

DEF VAR_LISTBOX_Text = (S/*$80000,$80001,$80002,$80003,$80004/$80001//DT4////200,,340,60)			
	Tipo di variabile:		STRING
	Valori limite/campo toggle:		*$80000,$80001,$80002,$80003,$80004 (lista dei testi dipendenti dalla lingua da visualizzare)
	Preassegnazione:		$80001
	Attributi:		
		Tipo di visualizzazione:	4 (casella di riepilogo)
	Posizione campo di input/output:		
		Distanza da sinistra:	200
		Larghezza:	340
		Altezza:	60
	Colori:		
		Colore di primo piano:	6 (blu)
		Colore di sfondo:	10 (bianco)

7.7 Esempio 5: Visualizzazione immagini

Esempio 5

Visualizzazione di un'immagine al posto di un testo sintetico: Dimensioni e posizione dell'immagine vengono indicati in "Posizione campo di input/output (Left, Top, Width, Height)".

DEF VAR6 = (V///,"\\bild1.png" ////160,40,50,50)			
	Tipo di variabile:		VARIANT
	Testi:		
		Testo sintetico:	bild1.png
	Posizione campo di input/output:		
		Distanza da sinistra:	160
		Distanza dall'alto:	40
		Larghezza:	50
		Altezza:	50

7.8 Esempio 6: Barra di avanzamento

La barra di avanzamento è un tipo di visualizzazione speciale del campo I/O ed è concepita solo come visualizzazione senza immissione.

Fondamentalmente esistono due tipi di barre di avanzamento:

1. Barre di avanzamento con fino a due evidenziazioni con colore per, ad es., la visualizzazione della temperatura o del fattore di utilizzo (vedere l'esempio 6a)
2. Fortschrittsbalken zur Anzeige eines Fortschrittes (ohne Farbumschlag) im Operate-Stil (siehe Beispiel 6b)

Esempio 6a

Barre di avanzamento con due evidenziazioni con colore:

Figura 7-1 Barre di avanzamento con due evidenziazioni con colore

DEF PROGGY0 = (R/0,150,50,100///DT1,DO0//"$R[10]"//,,150/3,4,,,,,,9,7)		
Tipo di variabile:		REAL
Valori limite/campo di toggle:		
	MIN:	0
	MAX:	150
	Valore di segnale SVAL1:	50
	Valore di segnale SVAL2:	100
Attributi:		
	Modalità di visualizzazione DT:	1 (barra di avanzamento)
	Opzione di visualizzazione DO:	0 (da sinistra a destra (default))
System- oder Anwendervariable:		$R[10]
Posizione campo di input/output:		
	Larghezza:	150
Colori:		
	Colore di primo piano:	3 (verde scuro)
	Colore di sfondo:	4 (grigio chiaro)
	Colore del segnale SC1:	9 (giallo)
	Colore del segnale SC2:	7 (rosso)

Per utilizzare una barra di avanzamento con evidenziazione con colore, la modalità di visualizzazione DT (DisplayType) deve essere impostata su 1.

L'orientamento dell'indicatore di avanzamento è determinato tramite l'attributo Opzione di visualizzazione DO (DisplayOption):

0: da sinistra a destra (default)

1: da destra a sinistra

2: dal basso verso l'alto

3: dall'alto verso il basso

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

Già con questa impostazione viene visualizzata, in funzione del valore attuale della variabile PROGGY0, una barra di avanzamento con il colore di primo piano 3 (= verde scuro) e il colore di sfondo 4 (= grigio chiaro).

Optional können ein oder zwei Signalwerte SVAL1 und SVAL2 (Parameter Grenzwert) definiert werden (im Beispiel SVAL1: 50 und SVAL2: 100). Con questi valori di segnale, il colore di primo piano dell'indicatore di avanzamento cambia. Die entsprechenden Signalfarben werden über die Parameter SC1 und SC2 festgelegt (in obigem Beispiel SC1: 9 (= gelb) und SC2: 7 (=rot)).

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

MIN < SVAL1 < SVAL2 < MAX.

Esempio 6b

Barra di avanzamento senza evidenziazione con colore:

Figura 7-2 Barra di avanzamento

DEF PROGGY0 = (R/0,150///DT2,DO0//"$R[10]"//,,150/6,10)		
Tipo di variabile:		REAL
Valori limite/campo di toggle:		
	MIN:	0
	MAX:	150
Attributi:		
	Tipo di visualizzazione DT:	2 (barra di avanzamento)
	Opzione di visualizzazione DO:	0 (da sinistra a destra (default))
System- oder Anwendervariable:		$R[10]
Posizione campo di input/output:		
	Larghezza:	150
Colori:		
	Colore di primo piano:	6 (blu)
	Colore di sfondo:	10 (bianco)

Per utilizzare una barra di avanzamento senza evidenziazione con colore, la modalità di visualizzazione DT (DisplayType) deve essere impostata su 2.

L'orientamento dell'indicatore di avanzamento è determinato tramite l'attributo Opzione di visualizzazione DO (DisplayOption) (vedere la descrizione dell'esempio 6a).

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

Con questa impostazione viene visualizzata, in funzione del valore attuale della variabile PROGGY0, una barra di avanzamento con il colore di primo piano 6 (= blu) e il colore di sfondo 10 (= bianco).

7.9 Esempio 7: Modalità di immissione della password (asterisco)

Esempio 7

Zur Realisierung eines Feldes mit verdeckter Eingabe, z. B. für die Eingabe eines Kennwortes, muss der Anzeigemodus DT auf 5 gesetzt werden. Invece dei caratteri immessi, vengono visualizzati degli asterischi.

Figura 7-3 Modalità di immissione della password (asterisco)

DEF VAR_SET_PWD=(S//""//DT5)		
Tipo di variabile:		STRING
Preimpostazione:		Stringa vuota
Attributi:		
	Modalità di visualizzazione DT:	5 (modalità di immissione della password)

7.10 Parametri delle variabili

Panoramica dei parametri

Nella seguente panoramica vengono brevemente illustrati i parametri delle variabili. Una descrizione dettagliata verrà fornita nei capitoli che seguono.

Parametri	**Descrizione**	
Tipo di variabile (Pagina 98)	Il tipo di variabile deve essere indicato.	
	R[x]:	REAL (+ cifre decimali)
	I:	INTEGER
	S[x]:	STRING (+ cifre per lunghezza stringa)
	C:	CHARACTER (carattere singolo)
	B:	BOOL
	V:	VARIANT

Parametri	Descrizione	
Valori limite (Pagina 86)	Valore limite MIN, valore limite MAX Preimpostazione: vuoto I valori limite vengono separati con la virgola. Per i tipi I, C e R i valori limite possono essere indicati nel formato decimale oppure come carattere nella forma "A", "F".	
	Per la rappresentazione di una barra di avanzamento con evidenziazione con colore (attributo di variabile DT = 1) è possibile progettare opzionalmente due colori di segnale SC1 e SC2 (Signal Color), che vengono utilizzati al superamento del valore di segnale SVAL1 o SVAL2 (Signal Value) come rispettivo colore di primo piano della barra. L'indicazione dei valori di segnale avviene come valori a numero intero. Vedere l'esempio pratico sulla barra di avanzamento (Pagina 88).	
Preassegnazione (Pagina 102)	Se non è progettata alcuna preassegnazione e alla variabile non è assegnata alcuna variabile di sistema o utente, viene assegnato il primo elemento del campo di toggle. Se non è definito alcun campo di toggle, non si verifica alcuna preassegnazione, ossia la variabile contiene lo stato "non calcolato". Preimpostazione: nessuna preassegnazione	
Campo di toggle (Pagina 100)	Elenco delle impostazioni predefinite nel campo di input/output: L'elenco è introdotto da un *, mentre i dati sono separati dalla virgola. Ai dati può essere assegnato un valore. Il dato per il valore limite viene interpretato nel campo di toggle come elenco. Se viene inserito solo un * viene creato un campo di toggle variabile. Preimpostazione: nessuno	
Testi (Pagina 85)	La successione è preimpostata. Al posto del testo sintetico può anche essere visualizzata un'immagine. Preimpostazione: vuoto	
	Testo completo:	Testo nella riga di visualizzazione
	Testo sintetico:	Nome dell'elemento della finestra di dialogo
	Testo grafico:	Il testo si riferisce a concetti nella grafica
	Testo unità:	Unità dell'elemento della finestra di dialogo
	Descrizione comandi	Fungono da informazione sintetica in una progettazione delle maschere per i campi di visualizzazione e di toggle. L'informazione viene progettata tramite testo in chiaro e ID del testo della lingua.

Parametri	Descrizione	
Attributi (Pagina 86)	Gli attributi influenzano le seguenti proprietà: • Modalità di visualizzazione • Opzione di visualizzazione • Frequenza di aggiornamento • Simbolo di toggle • Descrizione comandi • Modalità di immissione • Livello di accesso • Allineamento testo sintetico • Dimensioni carattere • Valori limite Gli attributi vengono separati dalla virgola, la sequenza è a piacere. Per ogni componente può avvenire una definizione.	
	Modalità di visualizzazione	dt0: standard (campo I/O o campo di toggle) (default) dt1: barra di avanzamento (progress bar) con evidenziazione con colore dt2: barra di avanzamento (progress bar) senza evidenziazione con colore nello stile Operate dt4: casella di riepilogo dt5: modalità di immissione della password (asterisco)
	Opzione di visualizzazione	Specialmente per le modalità di visualizzazione dt1 e dt2 (barra di avanzamento), ad es., può essere necessario progettare in combinazione anche un'opzione di visualizzazione. do0: da sinistra a destra (default) do1: da destra a sinistra do2: dal basso verso l'alto do3: dall'alto verso il basso
	Velocità di aggiornamento	Con l'attributo UR (update rate) è possibile controllare l'aggiornamento della visualizzazione e pertanto l'elaborazione del relativo blocco CHANGE progettato di variabili o colonne grid. A seconda della progettazione, è possibile ridurre drasticamente il carico della CPU e pertanto conseguire tempi di reazione più brevi dell'interfaccia utente. ur0: SlCap::standardUpdateRate() (attualmente = 200 ms, valore predefinito) ur1: 50 ms ur2: 100 ms ur3: 200 ms ur4: 500 ms ur5: 1000 ms ur6: 2000 ms ur7: 5000 ms ur8: 10000 ms
	Simbolo di toggle	tg0: simbolo di toggle OFF (default) tg1: simbolo di toggle ON Se si imposta l'attributo TG a 1, nella descrizione comandi del campo di immissione viene visualizzato anche il simbolo di commutazione.

Parametri	Descrizione	
		Esempio: `DEF OFFS = (R//123.456/,,,,"My ToolTip"/TG1)`

	Descrizione comandi	Il testo del tooltip può essere modificato anche per il tempo di esecuzione dalla proprietà di variabile "TT".
		Esempio:
		`PRESS(VS1)`
		`MyVar.TT = "My new ToolTip"`
		`END_PRESS`
		oppure
		`DEF MyVar=(R///,,,,"My ToolTip-text")`

	Modalità di immissione	wr0: campo di input/output non visibile, testo sintetico visibile wr1: lettura (non è possibile immettere dati nel campo) wr2: lettura e scrittura (la riga viene visualizzata in bianco) wr3: wr1 scrivibile wr4: tutti gli elementi della variabile non sono visibili, campo non scrivibile wr5: il valore introdotto viene immediatamente memorizzato a ogni pressione del tasto (al contrario di wr2, in cui viene memorizzato solo all'uscita dal campo o premendo RETURN). Preimpostazione: wr2
	Livello di accesso	vuoto: scrittura sempre possibile ac0...ac7: livelli di protezione Se il livello di accesso non è sufficiente, la riga viene visualizzata in grigio; impostazione standard: ac7
	Allineamento testo sintetico	al0: a sinistra al1: a destra al2: centrato Preimpostazione: al0
	Dimensioni carattere	fs1: dimensione standard del carattere (8 pt) fs2: dimensione doppia Preimpostazione: fs1 La distanza tra le righe è fissa. In caso di dimensione standard del carattere nella finestra di dialogo possono essere contenute 16 righe. Testo grafico e testo unità possono essere progettati solo nella dimensione standard del carattere.
	Valori limite	In tal modo è possibile verificare se il valore della variabile rientra nei valori limite MIN e MAX indicati. Preimpostazione: a seconda dei valori limite indicati li0: nessuna verifica li1: verifica del valore Min li2: verifica del valore Max li3: verifica dei valori Min e Max
	Comportamento all'apertura	Gli attributi cb, indicati in una definizione di variabile, hanno per la suddetta variabile la precedenza sull'indicazione forfettaria cb nella definizione della finestra di dialogo. Più attributi vengono annotati separati dalla virgola (vedere anche AUTOHOTSPOT).

Parametri	Descrizione	
	Richiamo del metodo CHANGE	CB0: il metodo CHANGE viene avviato con la visualizzazione della maschera se la variabile ha istantaneamente un valore valido (ad es. mediante preassegnazione o variabile NC/PLC). CB1: il metodo CHANGE non viene avviato esplicitamente con la visualizzazione della maschera. Anche se la variabile ha una variabile NC/PLC progettata viene naturalmente richiamato il metodo CHANGE.
	Empty Input (EI)	Con l'attributo variabile "EI" (= Empty Input) è possibile determinare la reazione del campo di immissione, se viene impostata una stringa vuota. EI0: standard, ovvero vengono accettate impostazioni vuote nel campo di immissione. EI1: impostazioni vuote provocano uno stato di errore nel campo di immissione e viene ripristinato il valore precedente valido (Undo).
Pagina di help (Pagina 85)	File della pagina di help:	Nome del file png Preimpostazione: vuoto
	Il nome del file per la pagina di help è indicato tra doppie virgolette. L'immagine viene visualizzata automaticamente (al posto della grafica precedente) appena il cursore arriva su questa variabile.	
Variabile di sistema o utente (Pagina 87)	Alle variabili può essere assegnato un dato di sistema o un dato utente dall'NC/PLC. La variabile di sistema o utente è indicata tra doppie virgolette. Bibliografia: Manuale delle liste, Variabili di sistema, /PGAsl/	
Posizione testo sintetico (Pagina 103)	Posizione del testo sintetico (distanza da sinistra, distanza dall'alto, larghezza) Le posizioni vengono indicate in pixel e si riferiscono all'angolo in alto a sinistra della parte principale della finestra di dialogo. I dati vengono separati da virgole.	
Posizione campo di input/output (Pagina 103)	Posizione del campo di input/output (distanza da sinistra, distanza dall'alto, larghezza, altezza) Le posizioni vengono indicate in pixel e si riferiscono all'angolo in alto a sinistra della parte principale della finestra di dialogo. I dati vengono separati da virgole. Se la posizione viene modificata, vengono modificate anche le posizioni di testo sintetico, testo grafico e testo unità.	

Parametri	Descrizione
Colori (Pagina 85)	Colore di primo piano e di sfondo per i campi I/O, testi sintetici, testi grafici, testi di unità e colori di segnale per le barre di avanzamento: I colori vengono separati con la virgola. Per la definizione del colore vedere il capitolo AUTOHOTSPOT. Preimpostazione per campo I/O: Colore di primo piano: nero, colore di sfondo: bianco I colori standard del campo di input/output dipendono dalla modalità di scrittura "wr". Preimpostazione per testo sintetico, testo grafico, testo dell'unità: Colore di primo piano: nero, colore di sfondo: trasparente.
	Per la rappresentazione di una barra di avanzamento con evidenziazione con colore (attributo di variabile DT = 1) è possibile progettare opzionalmente due colori di segnale SC1 e SC2 (Signal Color), che vengono utilizzati al superamento del valore di segnale SVAL1 o SVAL2 (Signal Value) come rispettivo colore di primo piano della barra. Vedere l'esempio pratico sulla barra di avanzamento (Pagina 88).
	Nella definizione delle variabili, i colori sono previsti nella seguente sequenza: 1. Colore di primo piano del campo I/O: FC 2. Colore di sfondo del campo I/O: BC 3. Colore di primo piano del testo sintetico: FC_ST 4. Colore di sfondo del testo sintetico: BC_ST 5. Colore di primo piano del testo grafico: FC_GT 6. Colore di sfondo del testo grafico: BC_GT 7. Colore di primo piano del testo delle unità: FC_UT 8. Colore di sfondo del testo delle unità: BC_UT 9. Colore del segnale 1 10. Colore del segnale 2
File della guida in linea	Il nome del file della Guida in linea è indicato tra doppie virgolette.

Campo I/O con selezione di unità integrata	Nel caso di campi di I/O con selezione di unità integrata è possibile passare da un'unità all'altra. Se ci si sposta con il cursore sul campo di immissione, viene evidenziata la selezione di unità integrata (non è necessario fare clic sul campo stesso). Inoltre viene visualizzata la descrizione comandi con un simbolo di toggle che allude a questa funzionalità.
	Esempi:
	`DEF VarEdit=(R/////////200,,100///"VarTgl")`
	`VarTgl=(S/*0="mm",1="inch"/0//WR2////302,,40)`
	oppure
	`DEF VarEdit_2={TYP="R", VAL=1.234, X=200, W=100, LINK_TGL="vartgl_2"},`
	`VARTgl_2={TYP="S", TGL="* 0=""mm"", 1=""inch""",WR=2, X=302, W=40}`

Variabile: Modifica delle proprietà

In caso di modifica, nella notazione Identificatore.Proprietà = Valore viene assegnato alle variabili un nuovo valore. L'espressione situata a destra dell'uguale viene interpretata e assegnata alla variabile o alla proprietà della variabile.

Identificatore.**ac** = livello di accesso	(ac: access level)
Identificatore.**al** = allineamento testo	(al: alignment)

Identificatore.**bc** = colore di sfondo del campo I/O	(bc: back color)
Identificatore.**bc_gt** = colore di sfondo del testo grafico	(bc: back color) (gt: graphic text)
Identificatore.**bc_st** = colore di sfondo del testo sintetico	(bc: back color) (st: short text)
Identificatore.**bc_ut** = colore di sfondo del testo unità	(bc: back color) (ut: unit text)
Identificatore.**do** = opzione di visualizzazione	(do: display option)
Identificatore.**dt** = modalità di visualizzazione	(dt: display type)
Identificatore**.fc** = colore di primo piano del campo I/O	(fc: front color)
Identificatore**.fc_gt** = colore di primo piano del testo grafico	(fc: front color) (gt: graphic text)
Identificatore.**fc_st** = colore di primo piano del testo sintetico	(fc: front color) (st: short text)
Identificatore.**fc_ut** = colore di primo piano del testo unità	(fc: front color) (ut: unit text)
Identificatore.**fs** = dimensione carattere	(fs: font size)
Identificatore.**gt** = testo grafico	(gt: graphic text)
Identificatore.**hlp** = pagina di help	(hlp: help)
Identificatore.**li** = valore limite	(li: limit)
Identificatore.**lt** = testo completo	(lt: long text)
Identificatore.**max** = valori limite MAX	(max: maximum)
Identificatore.**min** = valori limite MIN	(min: minimum)
Identificatore.**sc** = colore del segnale	(sc: signal color)
Identificatore.**st** = testo sintetico	(st: short text)
Identificatore.**tg** = simbolo di toggle	(tg: toggle)
Identificatore.**tt** = tooltip	(tt: tooltip
Identificatore.**typ** = tipo di variabile	(typ: type)
Identificatore.**ur** = velocità di aggiornamento	(ur: update rate)
Identificatore.**ut** = testo unità	(ut: unit text)
Identificatore.**val** = valore della variabile	(val: value)
Identificatore.**var** = variabile di sistema o utente	(var: variable)
Identificatore.**vld** = stato della variabile	(vld: validation)
Identificatore.**wr** = modalità di immissione	(wr: write)

Vedere anche

Sintassi di progettazione estesa (Pagina 46)

7.11 Particolarità sul tipo di variabile

Tipo di variabile INTEGER

Le seguenti estensioni per la determinazione della rappresentazione nel campo di input/output e dell'utilizzo della memoria sono possibili per il tipo "INTEGER":

2° carattere nel tipo di dati di estensione

Formato di rappresentazione	
B	binario
D	decimale con segno
H	esadecimale
Nessuna indicazione	decimale con segno

3° e/o 4° carattere nel tipo di dati di estensione

Utilizzo della memoria	
B	Byte
W	Word
D	Double Word
BU	Byte senza segno
WU	Word senza segno
DU	Double word senza segno

Sequenza dei caratteri nel tipo di dati INTEGER

1. "I" Identificazione di base quale INTEGER
2. Formato di rappresentazione
3. Utilizzo della memoria
4. "U" senza segno

Determinazioni di tipo valide per INTEGER:	
IB	Variabile Integer 32 bit in rappresentazione binaria
IBD	Variabile Integer 32 bit in rappresentazione binaria
IBW	Variabile Integer 16 bit in rappresentazione binaria
IBB	Variabile Integer 8 bit in rappresentazione binaria
I	Variabile Integer 32 bit in rappresentazione decimale con segno
IDD	Variabile Integer 32 bit in rappresentazione decimale con segno
IDW	Variabile Integer 16 bit in rappresentazione decimale con segno
IDB	Variabile Integer 8 bit in rappresentazione decimale con segno
IDDU	Variabile Integer 32 bit in rappresentazione decimale senza segno
IDWU	Variabile Integer 16 bit in rappresentazione decimale senza segno

Determinazioni di tipo valide per INTEGER:	
IDBU	Variabile Integer 8 bit in rappresentazione decimale senza segno
IH	Variabile Integer 32 bit in rappresentazione esadecimale
IHDU	Variabile Integer 32 bit in rappresentazione esadecimale
IHWU	Variabile Integer 16 bit in rappresentazione esadecimale
IHBU	Variabile Integer 8 bit in rappresentazione esadecimale

Tipo di variabile VARIANT

Il tipo di variabile VARIANT è determinato dal tipo di dato dell'ultima assegnazione di valore Se il valore assegnato o immesso inizia con '-'. '+', '.' o un numero ('0'-'9'), il valore viene interpretato numericamente. In tutti gli altri casi viene interpretato come stringa

e può essere interrogato con la funzione ISNUMo ISSTR. Il tipo VARIANT è in primo luogo adatto alla scrittura a scelta tra nomi di variabili o valori numerici nel codice NC.

Programmazione

È possibile controllare il tipo di dati della variabile:

Sintassi:	**ISNUM** (VAR)	
Parametri:	VAR	Nome della variabile il cui tipo di dati deve essere controllato.
		Il risultato dell'interrogazione può essere:
	FALSE =	nessuna variabile numerica (tipo di dati = STRING)
	TRUE =	variabile numerica (tipo di dati = REAL)

Sintassi:	**ISSTR** (VAR)	
Parametri:	VAR	Nome della variabile il cui tipo di dati deve essere controllato.
		Il risultato dell'interrogazione può essere:
	FALSE =	variabile numerica (tipo di dati = REAL)
	TRUE =	nessuna variabile numerica (tipo di dati = STRING)
Esempio:		

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

È possibile modificare la modalità di visualizzazione delle variabili:

- Per il tipo INTEGER il tipo di rappresentazione può essere modificato.

B	binario
D	decimale con segno
H	esadecimale
senza segno	
Inoltre, sempre U per unsigned (senza segno)	

- Per il tipo REAL è possibile modificare solo il numero di cifre decimali.
  La modifica del tipo non è consentita e provoca una segnalazione di errore nel file "easyscreen_log.txt".
  Esempio:
  ```
  Var1.typ = "IBW"
  Var2.typ = "R3"
  ```

Formati numerici

I numeri possono essere rappresentati in formato binario, decimale, esadecimale oppure esponenziale:

binario	B01110110
decimale	123.45
esadecimale	HF1A9
esponenziale	-1.23EX-3
Esempi:	
	`VAR1 = HF1A9`
	`REG[0]= B01110110`
	`DEF VAR7 = (R//-1.23EX-3)`

Nota

Nella generazione di codici mediante la funzione "GC" vengono considerati solo i valori numerici in formato decimale o esponenziale, **non** binario o esadecimale.

Vedere anche

Parametri delle variabili (Pagina 91)

7.12 Particolarità sul campo di toggle

Descrizione

Con l'ampliamento del campo toggle si possono visualizzare testi (dati immessi nel campo toggle) in funzione delle variabili NC/PLC. Una variabile che utilizza un'estensione di un campo toggle può solo essere letta. Premendo il tasto INSERT si apre la lista del campo toggle.

Programmazione

Sintassi:	**DEF VAR1=(IB/+ $85000/15////"DB90.DBB5")** oppure **DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run")**	
Descrizione:	Quando viene visualizzata la finestra di dialogo, nel campo di input/output viene emesso il contenuto del numero di testo $85015. Nella variabile di sistema DB90.DBB5 viene preassegnato 15. Se il valore nella variabile di sistema DB90.DBB5 cambia, ad ogni variazione il numero di testo visualizzato $(85000 + <DB90.DBB5>) viene ricalcolato.	
Parametri:	Tipo di variabile	Tipo di variabili specificate nella variabile di sistema o utente
	Numero di testo	Numero (base) del testo dipendente dalla lingua, che vale come numero di base.
	Variabile di sistema o utente	Variabile di sistema o utente (offset) attraverso cui viene creato il numero di testo finale (base + offset).

Immagini dipendenti dal campo toggle

Nel campo toggle vengono visualizzate alternativamente immagini diverse: Se il byte di merker ha valore 1, viene visualizzato "bild1.png", se il byte di merker ha valore 2, viene visualizzato "bild2.png".

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

Posizione e dimensioni dell'immagine vengono indicati in "Posizione campo di input/output(Left, Top, Width, Height)".

Tasto toggle virtuale

Per i campi toggle senza lista di scelta progettata, ad es. `DEF NoTglList=(R/*)`, non esiste alcuna lista. L'elemento successivo viene generato solo azionando il tasto toggle o eseguendo il metodo CHANGE() pertinente della relativa variabile.

In questo caso, per il comando con un touch panel viene visualizzata una piccola tastiera virtuale a destra del campo toggle variabile, che consta solo di un tasto toggle.

La visualizzazione del tasto virtuale di toggle può essere imposta anche indipendentemente da un touch panel con la seguente immissione nel file di configurazione "slguiconfig.ini".

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

Vedere anche

Parametri delle variabili (Pagina 91)

7.13 Particolarità sulla preassegnazione

Panoramica

A seconda del fatto che al campo di una variabile (campo di input/output o campo di toggle) siano associati una preassegnazione, una variabile di sistema o utente o entrambi, si ottengono diversi stati della variabile (non calcolati: Toggle possibile solo quando alla variabile è assegnato un valore valido).

Effetto delle preassegnazioni

Condizione			**Reazione del tipo di campo**
Tipo di campo	**Preassegnazione**	**Variabile di sistema o utente**	
Campo I/O	Sì	Sì	Scrittura della preassegnazione nella variabile di sistema o utente
	No	Sì	Utilizzare la variabile di sistema o utente come preassegnazione
	Errore	Sì	Non calcolato, la variabile di sistema o utente non viene configurata/utilizzata
	Sì	No	Preassegnazione
	No	No	Non calcolato
	Errore	No	Non calcolato
	Sì	Errore	Non calcolato
	No	Errore	Non calcolato
	Errore	Errore	Non calcolato
Toggle	Sì	Sì	Scrittura della preassegnazione nella variabile di sistema o utente
	No	Sì	Utilizzare la variabile di sistema o utente come preassegnazione
	Errore	Sì	Non calcolato, La variabile di sistema o utente non viene descritta/utilizzata
	Sì	No	Preassegnazione
	No	No	Preassegnazione = primo elemento del campo toggle
	Errore	No	Non calcolato
	Sì	Errore	Non calcolato
	No	Errore	Non calcolato
	Errore	Errore	Non calcolato

Vedere anche

Parametri delle variabili (Pagina 91)

7.14 Particolarità sulla posizione del testo sintetico, posizione del campo di input/output

Übersicht

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

Programmierung

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

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

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

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

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

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

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

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

Oder:

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

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

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

Besonderheit beim assoziierten Toggle-Feld

Voraussetzungen:

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

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

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

Beispiel

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

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

Vedere anche

Parametri delle variabili (Pagina 91)

7.15 Utilizzo di stringhe

Concatenamento di stringhe

Durante la progettazione è anche possibile utilizzare stringhe per creare una visualizzazione dinamica del testo oppure per riunire testi diversi per la generazione di codici.

Regole

Nell'utilizzo delle variabili String si devono rispettare le seguenti regole:

- I collegamenti vengono elaborati da sinistra verso destra.
- Espressioni inscatolate vengono elaborate dall'interno verso l'esterno.
- La scrittura in maiuscolo/minuscolo viene ignorata.
- Le variabili String vengono visualizzate generalmente allineate a sinistra.

Le stringhe possono essere cancellate semplicemente assegnando loro una stringa vuota.

Le stringhe possono essere unite a destra del segno dell'uguale utilizzando l'operatore "<<". Le doppie virgolette (") contenute in una stringa vengono rappresentate con due doppie virgolette in successione. L'uguaglianza di stringhe può essere verificata con istruzioni IF.

Esempio

Preimpostazione per gli esempi seguenti:

```
VAR1.VAL = "Questo è un"

VAR8.VAL = 4

VAR14.VAL = 15

VAR2.VAL = "errore"

$85001 = "Questo è un"

$85002 = "testo di allarme"
```

Elaborazione di stringhe:

- Unione di stringhe:
  ```
  VAR12.VAL = VAR1 << " errore."   ;Risultato: "Questo è un errore"
  ```
- Cancellazione di una variabile:
  ```
  VAR10.VAL = ""   ;Risultato: stringa vuota
  ```
- Impostazione di una variabile con una variabile di testo:
  ```
  VAR11.VAL = VAR1.VAL   ;Risultato: "Questo è un"
  ```
- Adattamento del tipo di dati:
  ```
  VAR13.VAL ="Questo è l'" << (VAR14 - VAR8) << ". errore"
              ;Risultato: "Questo è l'errore n. 11"
  ```
- Gestione di valori numerici:
  ```
  VAR13.VAL = "errore " << VAR14.VAL << ": " << $85001 << $85002
           ;Risultato: "Errore 15: Questo è un testo di allarme"
  IF VAR15 == "errore"   ;stringhe nell'istruzione IF
    VAR16 = 18.1234
           ;Risultato: VAR16 uguale a 18.1234,
           ;se VAR15 è uguale a "errore".
  ENDIF
  ```
- Doppie virgolette all'interno di una stringa:
  ```
  VAR2="Ciao questo è un " test""
     ;Risultato: Ciao questo è un " test"
  ```
- Stringhe di variabili di sistema o utente dipendenti dai contenuti delle variabili:
  ```
  VAR2.Var = "$R[" << VAR8 << "]"   ;Risultato: $R[4]
  ```

Vedere anche

Funzioni STRING (Pagina 180)

7.16 Variabile CURPOS

Descrizione

Con la variabile CURPOS è possibile richiamare o manipolare la posizione del cursore nel campo di input attivo della finestra di dialogo corrente. La variabile indica quanti caratteri si trovano prima del cursore. Se il cursore si trova all'inizio del campo di immissione, CURPOS assume il valore 0. Se si modifica il valore di CURPOS, il cursore si posiziona nella corrispondente posizione nell'ambito del campo di immissione.

Per poter reagire a variazioni del valore della variabile, è possibile rilevarle con l'ausilio di un metodo CHANGE. Se cambia il valore di CURPOS viene richiamato il metodo CHANGE ed eseguita l'istruzione in esso contenuta.

7.17 Variabile CURVER

Descrizione

La proprietà CRUVER (Current Version) consente l'adattamento della programmazione per l'utilizzo delle diverse versioni. La variabile CURVER può essere soltanto letta.

Nota

Durante la generazione di codici viene utilizzata automaticamente la versione più nuova, anche nel caso in cui precedentemente sia stata effettuata la riconversione con una versione più vecchia. Il comando "GC" genera sempre la versione più nuova. Nel codice generato viene inserita, nel commento operativo delle versioni > 0, un'identificazione aggiuntiva della versione generata.

Regole

Occorre sempre guardare la finestra di dialogo più nuova con tutte le sue variabili.

- Le variabili precedenti non devono essere modificate.
- Nuove variabili vengono introdotte in sequenza a piacere nella programmazione (di cicli) finora utilizzata.
- Non è consentito rimuovere variabili da una finestra di dialogo di una versione per spostarle alla successiva.
- La finestra di dialogo deve contenere tutte le variabili di tutte le versioni.

Esempio

```
(IF CURVER==1 ...)                ; In caso di riconversione, CURVER viene automa-
                                    ticamente impostato sulla versione del codice
                                    riconvertito.
```

7.18 Variabile ENTRY

Descrizione

Con la variabile ENTRY è possibile verificare come è stata richiamata la finestra di dialogo.

Programmazione

Sintassi:	**ENTRY**	
Descrizione:	La variabile ENTRY è di sola lettura.	
Valore di ritorno:		Il risultato dell'interrogazione può essere:
	0 =	Nessun supporto alla programmazione
	1 =	Start einer Maske über Softkey ; es wurde noch kein Code generiert (Vorbelegung, wie projektiert)
	2 =	Start einer Maske über Softkey ; es wurde Code generiert (preimpostazione a partire dall'ultimo codice generato per questa maschera)
	3 =	Decompilazione con commento operativo (righe #)
	4 =	Code_typ = 0 : NC-Code mit Nutzkommentar (#-Zeilen)
	5 =	Code_typ = 1 : NC-Code ohne Nutzkommentar (#-Zeilen)

Esempio

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

7.19 Variabile ERR

Descrizione

Con la variabile ERR si può verificare se la rispettiva riga precedente è stata eseguita senza errori.

Programmazione

Sintassi:	**ERR**
Descrizione:	La variabile ERR è di sola lettura.

Valore di ritorno:		Il risultato dell'interrogazione può essere:
	FALSE =	la riga precedente è stata eseguita correttamente
	TRUE =	la riga precedente non è stata eseguita correttamente

Esempio

```
VAR4 = Filetto[VAR1,"KDM",3]          ; Emissione del valore dall'array
IF ERR == TRUE                        ; Viene richiesto se il valore è stato trova-
                                      to nell'array
  VAR5 = "Errore durante l'accesso al-
l'array"
                                      ; Se il valore non è stato trovato nell'ar-
                                      ray, alla variabile viene assegnato il valo-
                                      re "Errore durante l'accesso all'array".
ELSE
  VAR5 = "Tutto OK"                   ;Se il valore è stato trovato nell'array, al-
                                      la variabile viene assegnato il valore "Tut-
                                      to OK".
ENDIF
```

7.20 Variabile FILE_ERR

Descrizione

Con la variabile FILE_ERR si può verificare se le precedenti istruzioni GC o CP sono state eseguite senza errori.

Programmazione

Sintassi:	**FILE_ERR**	
Descrizione:	La variabile FILE_ERR è di sola lettura.	
Valore di ritorno:		Sono possibili i seguenti risultati:
	0 =	Operazione corretta
	1 =	Drive/percorso non esistente
	2 =	Errore di accesso al percorso/file
	3 =	Drive non pronto
	4 =	Nome del file errato
	5 =	File già aperto
	6 =	Accesso negato
	7 =	Percorso di destinazione non esistente o non consentito
	8 =	Sorgente e destinazione di copia sono identiche
	10 =	Errore interno: nel caso di FILE_ERR = 10 si tratta di un errore che non può essere classificato in altre categorie.

Esempio

```
CP("D:\source.mpf","E:\target.mpf")
                                                ;Copia di source.mpf in E:\target.mpf
IF FILE_ERR > 0                                 ;Esamina se si è verificato un errore
   IF FILE_ERR == 1                             ; Richiamo di particolari numeri di
                                                errore ed emissione del testo di er-
                                                rore relativo
      VAR5 = "Drive/percorso non esistente”
   ELSE
      IF FILE_ERR == 2
        VAR5 = “Errore di accesso al percorso/
file”
      ELSE
         IF FILE_ERR == 3
            VAR5 = "Nome di file errato"
ENDIF
      ENDIF
   ENDIF
ELSE
   VAR5 = "Tutto OK"                            ; se non si è verificato alcun errore
                                                in CP (oppure GC), si ha l'emissione
                                                "Tutto OK"
ENDIF
```

7.21 Variabile FOC

Descrizione

Con la variabile FOC si controlla il campo di immissione (attuale campo di input/output attivo) in una finestra di dialogo. Le reazioni del cursore a sinistra, a destra, in alto, in basso nonché PGUP/PGDN sono predefinite in modo fisso.

Nota

FOC non può essere attivato con un evento di navigazione. La posizione del cursore può essere modificata soltanto in metodi di softkey PRESS, CHANGE, ecc.

Variabili con modo di immissione wr = 0 e wr = 4 e variabili ausiliarie non possono essere selezionate per l'immissione.

Programmazione

Sintassi:	**FOC**	
Descrizione:	La variabile può essere letta e scritta.	
Valore di ritorno:	Lettura	Come risultato viene fornito il nome della variabile selezionata.
	Scrittura	È possibile assegnare una stringa oppure un valore numerico. Una stringa viene interpretata come nome di una variabile ed un valore numerico come indice della variabile.

Esempio

```
IF FOC == "Var1"              ; Lettura dell'elemento selezionato
  REG[1] = Var1
ELSE
  REG[1] = Var2
ENDIF

FOC = "Var1"                  ; Il campi di immissione viene assegnato alla variabi-
                              le 1.
FOC = 3                       ; Il campo di immissione viene assegnato al 3° elemen-
                              to della finestra di dialogo con WR ≥ 2.
```

Vedere anche

FOCUS (Pagina 123)

7.22 Variable S_ALEVEL

Descrizione

Il livello di accesso attuale può essere oggetto di interrogazione nella programmazione con la proprietà di maschera S_ALEVEL.

Programmazione

Sintassi:	**S_ALEVEL**
Descrizione:	Interrogazione del livello di accesso attuale
Valore di ritorno:	0: Sistema 1: Marchio 2: Service 3: Utente 4: Interruttore a chiave 3 5: Interruttore a chiave 2 6: Interruttore a chiave 1 7: Interruttore a chiave 0

Esempio

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

Vedere anche

ACCESSLEVEL (Pagina 120)

7.23 Variabile S_CHAN

Descrizione

Con la variabile S_CHAN è possibile rilevare il numero del canale corrente per scopi di visualizzazione o analisi.

Programmazione

Sintassi:	**S_CHAN**
Descrizione:	Interrogazione del numero di canale attuale
Valore di ritorno:	Numero di canale

Esempio

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

Vedere anche

CHANNEL (Pagina 122)

7.24 Variable S_CONTROL

Descrizione

Il nome del controllore attuale può essere oggetto di interrogazione nella programmazione con la proprietà di maschera S_CONTROL.

Programmazione

Sintassi:	**S_CONTROL**
Descrizione:	Interrogazione del nome del controllore attuale
Valore di ritorno:	Il nome del controllore è il nome della sezione nel file "mmc.ini"

Esempio

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

Vedere anche

CONTROL (Pagina 122)

7.25 Variable S_LANG

Descrizione

La lingua attuale può essere oggetto di interrogazione nella programmazione con la proprietà di maschera S_LANG.

Programmazione

Sintassi:	**S_LANG**
Descrizione:	Interrogazione della lingua attuale
Valore di ritorno:	Codice lingua da resources.xml z.B. "deu", "eng", ecc.

Esempio

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

Vedere anche

LANGUAGE (Pagina 124)

7.26 Variable S_NCCODEREADONLY

Descrizione

La proprietà delle maschere S_NCCODEREADONLY ha rilevanza esclusivamente se nell'editor viene decompilato un ciclo con una finestra di dialogo "Run MyScreens". Con S_NCCODEREADONLY si accerta se un codice NC decompilato a partire dall'editor può essere modificato o no.

Per il valore di ritorno vale

- TRUE: il codice NC decompilato proveniente dall'editor può essere modificato
- FALSE: il codice NC decompilato a partire dall'editor non può essere modificato (readonly) poiché lo stesso, ad es., è già in preelaborazione blocchi (= TRUE).

7.27 Variable S_RESX e S_RESY

Descrizione

La risoluzione attuale o il relativo componente X/Y può essere oggetto di interrogazione nella progettazione con le proprietà di maschera S_RESX e S_RESY.

Esempio

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

Vedere anche

RESOLUTION (Pagina 128)

Comandi di programmazione

8

8.1 Operatori

Panoramica

Durante la programmazione è possibile utilizzare i seguenti operatori:

- Operatori matematici
- Operatori di confronto
- Operatori logici (booleani)
- Operatori a bit
- Funzioni trigonometriche

8.1.1 Operatori matematici

Panoramica

Operatori matematici	Definizione
+	Addizione
-	Sottrazione
*	Moltiplicazione
/	Divisione
MOD	Operazione modulo
( )	Parentesi
AND	Operatore AND
OR	Operatore OR
NOT	Operatore NOT
ROUND	Arrotondamendo numeri decimali

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

ROUND

L'operatore ROUND viene impiegato per l'arrotondamento di numeri fino a 12 cifre decimali durante l'elaborazione della progettazione di una finestra di dialogo. Le cifre decimali non possono essere visualizzate dai campi delle variabili.

Impiego

ROUND viene controllato dall'utente attraverso due parametri:

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

Risultato: `VAR2 = 5,2339`

VAR1 contiene il numero da arrotondare. Il parametro "4" indica il numero di cifre decimali risultanti da inserire in VAR2.

Funzioni trigonometriche

Funzioni trigonometriche	Definizione
SIN(x)	Seno di x
COS(x)	Coseno di x
TAN(x)	Tangente di x
ATAN(x, y)	Arcotangente di x/y
SQRT(x)	Radice quadrata di x
ABS(x)	Valore assoluto di x
SDEG(x)	Conversione in gradi
SRAD(x)	Conversione in radianti
CALC_ASIN(x)	Arcoseno di x
CALC_ACOS(x)	Arcocoseno di x

Nota

Le funzioni operano in misura di arco. Per la conversione si possono utilizzare le funzioni SDEG() und SRAD().

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

Funzioni matematiche

Funzioni matematiche	Definizione
CALC_CEIL(x)	determina il numero intero immediatamente superiore di x (arrotondamento per eccesso)
CALC_FLOOR(x)	determina il numero intero immediatamente inferiore di x (arrotondamento per difetto)
CALC_LOG(x)	determina il logaritmo (naturale) in base e di x
CALC_LOG10(x)	determina il logaritmo in base 10 di x
CALC_POW(x, y)	determina x elevato a y (x alla y-esima potenza)
CALC_MIN(x, y)	determina il numero minore di x e y
CALC_MAX(x, y)	determina il numero minore di x e y

Funzione Random: numero casuale

Sintassi:	**RANDOM** (valore limite inferiore, valore limite superiore)	
Descrizione:	La funzione **RANDOM** restituisce un pseudonumero casuale in un intervallo predefinito.	
Parametri:	Valore limite inferiore	valore limite inferiore >= -32767, valore limite inferiore < valore limite superiore
	Valore limite superiore	Valore limite superiore <= 32767

Esempio

```
REG[0] = RANDOM(-10,10)         ; Risultato possibile = -3
```

Costanti

Costanti	
PI	3.14159265358979323846
FALSE	0
TRUE	1

Esempio: `VAR1.VAL = PI`

Operatori di confronto

Operatori di confronto	
==	uguale a
<>	diverso
>	maggiore
<	minore
>=	maggiore o uguale
<=	minore o uguale

Esempio:

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

Condizioni

Il livello di annidamento è illimitato.

Condizione con un'istruzione:

```
IF
...
ENDIF
```

Condizione con due istruzioni:

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

8.1.2 Operatori a bit

Panoramica

Operatori a bit	Definizione
BOR	OR a bit
BXOR	XOR a bit
BAND	AND a bit
BNOT	NOT a bit
SHL	Shift a bit verso sinistra
SHR	Shift a bit verso destra

Operatore SHL

Con l'operatore SHL (SHIFT LEFT) i bit vengono spostati verso sinistra. Sia il valore da spostare che il numero di incrementi di spostamento possono essere impostati direttamente o come variabili. Quando viene raggiunto il limite del formato di dati, i bit vengono "spinti fuori" senza nessuna segnalazione di errore.

Impiego

Sintassi:	variabile = valore **SHL**numero di incrementi	
Descrizione:	Sposta a sinistra	
Parametri:	valore	valore da spostare
	numero di incrementi	numero di incrementi di spostamento

Esempio

```
PRESS(VS1)
```

```
  VAR01 = 16 SHL 2           ;Risultato = 64
  VAR02 = VAR02 SHL VAR04    ; Il contenuto di VAR02 viene trasformato in un 32 bit
                             senza segno e spostato del contenuto di VAR04 bit verso
                             sinistra. Successivamente, il valore a 32 bit viene ri-
                             convertito nel formato della variabile VAR02.
END_PRESS
```

Operatore SHR

Con l'operatore SHR (SHIFT RIGHT) i bit vengono spostati verso destra. Sia il valore da spostare che il numero di incrementi di spostamento possono essere impostati direttamente o come variabili. Quando viene raggiunto il limite del formato di dati, i bit vengono "spinti fuori" senza nessuna segnalazione di errore.

Impiego

Sintassi:	variabile = valore **SHR**numero di incrementi	
Descrizione:	Sposta a destra	
Parametri:	valore	valore da spostare
	numero di incrementi	numero di incrementi di spostamento

Esempio

```
PRESS(VS1)
  VAR01 = 16 SHR 2           ;Risultato = 4
  VAR02 = VAR02 SHR VAR04    ; Il contenuto di VAR02 viene trasformato in un 32
                             bit senza segno e spostato del contenuto di VAR04
                             bit verso destra. Successivamente, il valore a 32
                             bit viene riconvertito nel formato della variabile
                             VAR02.
END_PRESS
```

8.2 Metodi

Panoramica

Nelle finestre di dialogo e nelle barre di softkey dipendenti dalle finestre di dialogo (barre di softkey richiamate da una nuova finestra di dialogo progettata) è possibile attivare particolari azioni attraverso eventi differenti (uscita dal campo di input, pressione di softkey). Queste azioni vengono progettate nei metodi.

La programmazione di base di un metodo si presenta nel seguente modo:

Blocco di definizione	Commento	Rimando al capitolo
`PRESS(HS1)`	;Codice iniziale metodo	
`LM...` `  LS...`	;Funzioni	Vedere il capitolo Funzioni (Pagina 132)
`Var1.st = ...`	;Modifica di proprietà	Vedere il capitolo Definizione delle barre di softkey (Pagina 64) e il capitolo Definizione delle proprietà delle finestre di dialogo (Pagina 51)

`Var2 = Var3 + Var4`
`...`
`EXIT`
	;Calcolo con variabili	Vedere il capitolo Definizione delle variabili (Pagina 83)

`END_PRESS`	;Codice finale metodo	

8.2.1 ACCESSLEVEL

Descrizione

Il metodo ACCESSLEVEL viene eseguito quando, a maschera aperta, il livello di accesso attuale si è modificato.

Programmazione

Sintassi:	ACCESSLEVEL <Istruzioni> END_ACCESSLEVEL
Descrizione:	Livello di accesso
Parametri:	- nessuno -

Vedere anche

Variable S_ALEVEL (Pagina 110)

8.2.2 CHANGE

Descrizione

I metodi CHANGE vengono eseguiti alla variazione del valore di una variabile. Ciò significa che nell'ambito di un metodo CHANGE vengono progettati dei calcoli di variabili che vengono eseguiti subito ad ogni variazione di una variabile.

Viene operata una distinzione tra metodo CHANGE specifico per l'elemento e globale:

- Il **metodo CHANGE specifico per l'elemento** viene eseguito quando il valore della variabile specificata cambia. Se ad una variabile viene associata una variabile di sistema o utente, in un metodo CHANGE è possibile aggiornare ciclicamente il valore della variabile.
- Il **metodo CHANGE globale** viene eseguito, quando il valore di una qualsiasi variabile cambia e non è progettato alcun metodo CHANGE specifico per l'elemento.

Programmazione "specifica per l'elemento"

Sintassi:	CHANGE(Identifcatore) ... END_CHANGE	
Descrizione:	Modifica del valore della variabile specificata	
Parametri:	Identificatore	Nome della variabile

Esempio

```
DEF VAR1=(I//////"DB20.DBB1")          ; A Var1 viene assegnata una variabile di si-
                                       stema
CHANGE(VAR1)
  IF VAR1.Val <> 1
  VAR1.st="Utensile OK!"               ; Se il valore della variabile di sistema è ≠
                                       1, il testo sintetico della variabile sarà:
                                       Utensile OK!
    otto=1
  ELSE
    VAR1.st="Attenzione errore!"       ; Se il valore della variabile di sistema è =
                                       1, il testo sintetico della variabile sarà: At-
                                       tenzione errore!
    otto=2
  ENDIF
  VAR2.Var=2
END_CHANGE
```

Programmazione "globale"

Sintassi:	CHANGE() ... END_CHANGE
Descrizione:	Modifica di un valore di variabile qualsiasi
Parametri:	- nessuno -

Esempio

```
CHANGE()
  EXIT                                ; Se si modifica un valore di variabile qual-
                                      siasi, si esce dalla finestra di dialogo.
END_CHANGE
```

8.2.3 CHANNEL

Descrizione

Il metodo CHANNEL viene eseguito quando, a maschera aperta, il canale attuale si è modificato, ossia ha avuto luogo una commutazione di canale.

Programmazione

Sintassi:	CHANNEL <Istruzioni> END_CHANNEL
Descrizione:	Commutazione del canale
Parametri:	- nessuno -

Vedere anche

Variabile S_CHAN (Pagina 111)

8.2.4 CONTROL

Descrizione

Il metodo CONTROL viene eseguito quando, a maschera aperta, il controllore attuale si è modificato, tipicamente durante una commutazione 1:n.

Programmazione

Sintassi:	CONTROL <Istruzioni> END_CONTROL
Descrizione:	Commutazione controllore
Parametri:	- nessuno -

Vedere anche

Variable S_CONTROL (Pagina 112)

8.2.5 FOCUS

Descrizione

Il metodo FOCUS viene eseguito quando nella finestra di dialogo il cursore viene posizionato su un altro campo.

Il metodo FOCUS non può essere attivato con un evento di navigazione. La posizione del cursore può essere modificata soltanto in metodi di softkey PRESS, CHANGE, ecc. La reazione dei movimenti del cursore è predefinita in modo fisso.

Nota

All'interno del metodo FOCUS non è consentito effettuare il posizionamento su un'altra variabile né caricare una nuova finestra di dialogo.

Programmazione

Sintassi:	FOCUS ... END_FOCUS
Descrizione:	Posizionamento del cursore
Parametri:	- nessuno -

Esempio

```
FOCUS
  DLGL("L'immissione stata posizionata sulla variabile " << FOC << ".")
END_FOCUS
```

Vedere anche

Variabile FOC (Pagina 109)

8.2.6 LANGUAGE

Descrizione

Il metodo LANGUAGE viene eseguito quando, a maschera aperta, la lingua attuale si è modificata.

Programmazione

Sintassi:	LANGUAGE <Istruzioni> END_LANGUAGE
Descrizione:	Lingua
Parametri:	- nessuno -

Vedere anche

Variable S_LANG (Pagina 112)

8.2.7 LOAD

Descrizione

Il metodo LOAD viene eseguito quando sono state interpretate le definizioni dei softkey e delle variabili (DEF Var1= ..., HS1= ...). A questo punto la finestra di dialogo non è ancora visualizzata.

Programmazione

Sintassi:	LOAD ... END_LOAD
Descrizione:	Caricamento
Parametri:	- nessuno -

Esempio

```
LOAD                          ; Codice iniziale
  Maske1.Hd =  $85111         ; Assegnazione del testo per il titolo della finestra
                                di dialogo dal file di lingua
  VAR1.Min = 0                ; Assegnazione del valore limite MIN alla variabile
  VAR1.Max = 1000             ; Assegnazione del valore limite MAX alla variabile
END_LOAD                      ; Codice finale
```

Vedere anche

Definizione degli elementi grafici (Pagina 191)

8.2.8 UNLOAD

Descrizione

Il metodo UNLOAD viene eseguito prima che venga scaricata una finestra di dialogo.

Programmazione

Sintassi:	UNLOAD ... END_UNLOAD
Descrizione:	Scaricamento di un utensile
Parametri:	- nessuno -

Esempio

```
UNLOAD
  REG[1] = VAR1            ; Archiviazione della variabile nel registro
END_UNLOAD
```

8.2.9 OUTPUT

Descrizione

Il metodo OUTPUT viene eseguito quando viene richiamata la funzione "GC". In un metodo OUTPUT vengono progettate variabili principali e ausiliarie come codice NC. Il concatenamento di singoli elementi di una riga di codice avviene con caratteri di spaziatura.

Nota

Il codice NC può essere generato in un file extra con le funzioni file e trasferito poi nell'NC.

Programmazione

Sintassi:	OUTPUT(Identificatore) ... END_OUTPUT	
Descrizione:	Emissione di variabili nel programma NC	
Parametri:	Identificatore	Nome del metodo OUTPUT

Numero di blocco e codici di esclusione

Il metodo OUTPUT non deve contenere alcun numero di riga o codice di esclusione se questi ultimi, impostati direttamente nel programma pezzo con il supporto alla programmazione attivo, devono essere mantenuti anche in caso di ricompilazione.

Eventuali modifiche con l'editor nel programma pezzo generano il seguente comportamento:

Condizione	Comportamento
Il numero di blocchi resta invariato.	I numeri di blocco vengono mantenuti.
Il numero di blocchi si riduce.	I numeri di blocco maggiori vengono cancellati.
Il numero di blocchi aumenta.	I nuovi blocchi non ricevono un numero di blocco.

Esempio

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

8.2.10 PRESS

Descrizione

Il metodo PRESS viene eseguito se è stato premuto il softkey corrispondente.

Programmazione

Sintassi:	PRESS(Softkey) ... END_PRESS
Definizione:	Pressione di un softkey

Parametri:	Softkey	Nome del softkey: HS1 - HS8 e VS1 - VS8	
	RECALL	Tasto <RECALL>	
	ENTER	Tasto <ENTER>, vedere PRESS(ENTER) (Pagina 127)	
	TOGGLE	Tasto <TOGGLE>, vedere PRESS(TOGGLE) (Pagina 128)	
	PU	Page Up	Pagina su
	PD	Page Down	Pagina giù
	SL	Scroll Left	Cursore verso sinistra
	SR	Scroll Right	Cursore verso destra
	SU	Scroll Up	Cursore verso l'alto
	SD	Scroll Down	Cursore verso il basso

Esempio

```
HS1 = ("altra barra dei softkey")
HS2 = ("nessuna funzione")
PRESS (HS1)
  LS("Barra1")                    ; Caricamento di un'altra barra dei softkey
  Var2 = Var3 + Var1
END_PRESS
PRESS (HS2)
END_PRESS
PRESS(PU)
  INDEX = INDEX -7
  CALL("UP1")
END_PRESS
```

8.2.11 PRESS(ENTER)

Descrizione

Il metodo PRESS(ENTER) viene sempre richiamato se, in presenza di una variabile con campo I/O con modalità di immissione WR3 o WR5, si preme il tasto Invio:

- WR3: spostamento sul campo e pressione del tasto Invio
- WR5: nella modalità di immissione, applicazione del valore con il tasto Invio

Programmazione

Sintassi:	PRESS(ENTER) <Istruzioni> END_PRESS
Descrizione:	Tasto Invio premuto
Parametri:	- nessuno -

8.2.12 PRESS(TOGGLE)

Descrizione

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

All'occorrenza, con l'ausilio della proprietà maschera FOC è possibile determinare quale variabile sia attualmente attiva.

Programmazione

Sintassi:	PRESS(TOGGLE) <Istruzioni> END_PRESS
Descrizione:	Tasto di toggle premuto
Parametri:	- nessuno -

Esempio

```
PRESS(TOGGLE)
   DLGL("Toggle key pressed at variable " << FOC)       ; con la proprietà maschera FOC si determina
                                                        quale variabile sia attualmente attiva
END_PRESS
```

8.2.13 RESOLUTION

Descrizione

Il metodo RESOLUTION viene eseguito quando, a maschera aperta, la risoluzione attuale si è modificata, tipicamente durante una commutazione TCU.

Programmazione

Sintassi:	RESOLUTION <Istruzioni> END_RESOLUTION
Descrizione:	Risoluzione
Parametri:	- nessuno -

Vedere anche

Variable S_RESX e S_RESY (Pagina 113)

8.2.14 RESUME

Descrizione

Il metodo RESUME viene richiamato quando la maschera, ad es., è stata interrotta da un cambio di area e successivamente viene visualizzata di nuovo. Nella circostanza le modifiche di valore vengono nuovamente elaborate, eventualmente i timer avviati e il blocco RESUME viene eseguito.

Nota

Con i metodi SUSPEND e RESUME non si possono progettare le funzioni LS, LM, DLGL, EXIT e EXITLS. Se si utilizzano le funzioni in questi metodi, l'esecuzione delle stesse viene impedita.

Programmazione

Sintassi:	RESUME <Istruzione> END_RESUME
Descrizione:	Maschera di nuovo attiva
Parametri:	- nessuno -

8.2.15 SUSPEND

Descrizione

Il metodo SUSPEND è richiamato quando la maschera viene interrotta e non scaricata. Ciò accade, ad es., se si esce dalla maschera per effetto di un semplice cambio di area ma la maschera stessa non viene esplicitamente scaricata. La maschera in tal caso resta conservata in background, tuttavia non si elaborano modifiche di valore o eventualmente i timer. La maschera è sospesa.

Nota

Con i metodi SUSPEND e RESUME non si possono progettare le funzioni LS, LM, DLGL, EXIT e EXITLS. Se si utilizzano le funzioni in questi metodi, l'esecuzione delle stesse viene impedita.

Programmazione

Sintassi:	SUSPEND <Istruzione> END_SUSPEND
Descrizione:	Interruzione della maschera
Parametri:	- nessuno -

Esempio

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

8.2.16 Esempio: Gestione delle versioni con metodi OUTPUT

Panoramica

Nell'ambito di ampliamenti le finestre di dialogo esistenti possono essere integrate con ulteriori variabili. Nelle definizioni di queste variabili supplementari, tra parentesi tonde dopo il nome della variabile, viene inserito il codice della versione. (0 = Originale, non viene scritto), 1 = Versione 1, 2 = Versione 2, ...

Esempio:

```
DEF var100=(R//1)               ;Originale, corrisponde alla versione 0
DEF var101(1)=(S//"Hallo")      ;Ampliamento, dalla versione 1
```

Scrivendo i metodi di OUTPUT si può fare riferimento ad una determinata versione, riguardante la totalità delle definizioni.

Esempio:

```
OUTPUT(NC1)                    ; Nel metodo OUTPUT vengono proposte solo le varia-
                               bili dell'originale
OUTPUT(NC1,1)                  ; Le variabili dell'originale e le integrazioni con
                               identificatore della versione 1 vengono proposte nel
                               metodo di OUTPUT.
```

Il metodo di OUTPUT per l'originale non necessita di un identificatore della versione, eventualmente si può scrivere 0. OUTPUT(NC1) corrisponde a OUTPUT(NC1,0). L'identificatore della versione n nel metodo di OUTPUT comprende tutte le variabili dell'originale 0, 1, 2, ... fino a n incluso.

Programmazione con identificatore della versione

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

OUTPUT(NC1)
var100",,"var101
END_OUTPUT

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

OUTPUT(NC1)                    Originale e, in aggiunta, la nuova versione
var100","var101
END_OUTPUT
```

```
...

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

8.3 Funzioni

Panoramica

Nelle finestre di dialogo e nelle barre di softkey dipendenti dalle finestre di dialogo sono disponibili diverse funzioni, le quali vengono attivate e progettate nei metodi attraverso eventi, quali ad es. uscita dal campo di input, pressione del softkey.

Sottoprogrammi

Le istruzioni di progettazione ricorsive o di altro tipo, che comprendono un particolare processo, possono essere progettate in sottoprogrammi. I sottoprogrammi possono essere caricati nel programma principale o in altri sottoprogrammi in qualunque momento ed essere elaborate con frequenza a piacere; ciò significa che le istruzioni non devono essere progettate più volte. Come programma principale valgono i blocchi di definizione delle finestre di dialogo o la barra di softkey.

Funzioni esterne

Con l'ausilio delle funzioni esterne si possono integrare ulteriori funzioni specifiche dell'utente. Le funzioni esterne vengono inserite in un file DLL e rese note attraverso una registrazione nella riga di definizione del file di progettazione.

Servizi PI

Attraverso la funzione PI_START è possibile avviare servizi PI (servizi Program Instance) dal PLC in campo NC.

Vedere anche

Function (FCT) (Pagina 152)

Servizi PI (Pagina 168)

8.3.1 Lettura e scrittura dei parametri di azionamento: RDOP, WDOP, MRDOP

Descrizione

Con l'ausilio delle funzioni RDOP, WDOP e MRDOP è possibile leggere o scrivere i parametri di azionamento.

Nota

Leggere i parametri dell'azionamento con un clock non più veloce di 1 secondo, se possibile anche più lento.

Motivo: La comunicazione con gli azionamenti potrebbe risultarne molto disturbata o potrebbe addirittura interrompersi.

Nota

Se durante la lettura o scrittura dei parametri di azionamento si verificassero errori, verrebbe di conseguenza impostata la proprietà di maschera ERR.

Programmazione

Sintassi:	**RDOP**("identificatore dell'oggetto di azionamento", "numero di parametro")	
Descrizione:	Lettura di un parametro di azionamento (Drive Object Parameter)	
Parametri:	Identificatore dell'oggetto di azionamento	È possibile evincere l'identificatore dell'oggetto di azionamento dalla colonna "Nome DO" in "Diagnostica sistema di azionamento" nel settore operativo Diagnostica (> ETC > HSK 8: sistema di azionamento) (vedere la seguente figura).
	Numero parametro	

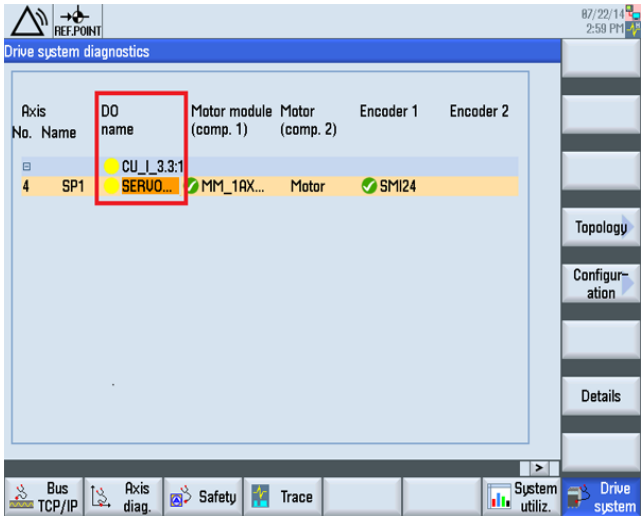


Figura 8-1 Identificatore dell'oggetto di azionamento

Sintassi:	**MRDOP**("identificatore dell'oggetto di azionamento", "numero di parametro1"*"numero di parametro"[*...], indice registro)	
Descrizione:	Multi Read di parametri di azionamento. Con il comando MRDOP si possono trasferire nel registro più parametri di azionamento di un oggetto di azionamento con un solo accesso. Questo accesso è notevolmente più veloce di una lettura tramite singoli accessi.	
Parametri:	Identificatore dell'oggetto di azionamento	L'identificatore dell'oggetto di azionamento si può desumere, ad esempio, dalla pagina base del settore operativo "Messa in servizio"
	Numero di parametro1 ... n	Nel nome delle variabili il segno "*" funge da carattere separatore. Nella sequenza in cui i nomi delle variabili risultano nel comando vengono acquisiti i valori nei registri REG[Indice registro].
	Indice registro	il valore della prima variabile si trova in REG[indice registro] il valore della seconda variabile si trova in REG[indice registro + 1]

Sintassi:	**WDOP**("identificatore dell'oggetto di azionamento", "numero di parametro", valore)
Descrizione:	Scrittura di un parametro di azionamento (Drive Object Parameter)

Parametri:	Identificatore dell'oggetto di azionamento	L'identificatore dell'oggetto di azionamento si può desumere, ad esempio, dalla pagina base del settore operativo "Messa in servizio"
	Numero parametro	Numero parametro
	Valore	Valore da scrivere

Esempi

Lettura della temperatura del motore r0035 dell'oggetto di azionamento "SERVO_3.3:2":

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

Lettura della temperatura del motore r0035 e del valore attuale di coppia r0080 dell'oggetto di azionamento "SERVO_3.3:2" e memorizzazione dei relativi risultati a partire dall'indice di registro 10:

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

8.3.2 Richiamo del sottoprogramma (CALL)

Descrizione

Con la funzione CALL un sottoprogramma caricato può essere richiamato in un qualsiasi punto di un metodo. È consentito l'inscatolamento, cioè il richiamo di un sottoprogramma da un altro sottoprogramma.

Programmazione

Sintassi:	**CALL**("Identificatore")	
Descrizione:	Richiamo di sottoprogrammi	
Parametri:	Identificatore	Nome del sottoprogramma

Esempio

```
//M(MASCHERA1)
DEF VAR1 = ...
DEF VAR2 = ...
CHANGE(VAR1)
  ...
  CALL("MY_UP1")             ; Richiamo ed elaborazione del sottoprogramma
  ...
END_CHANGE
```

```
//M(MASCHERA1)
CHANGE(VAR2)
  ...
  CALL("MY_UP1")              ; Richiamo ed elaborazione del sottoprogramma
  ...
END_CHANGE

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

8.3.3 Definizione del blocco (//B)

Descrizione

I sottoprogrammi vengono contrassegnati nel file di programma con il codice di blocco //B e terminati con //END. Per ogni codice di blocco si possono definire più sottoprogrammi.

Nota

Le variabili utilizzate nel sottoprogramma devono essere definite nella finestra di dialogo in cui viene richiamato il sottoprogramma.

Programmazione

Un blocco ha la seguente struttura:

Sintassi:	**//B**(Nome blocco) **SUB**(Identificatore) END_SUB [SUB(identificatore) ... END_SUB] ... //END	
Descrizione:	Definizione del sottoprogramma	
Parametri:	nome del blocco	Nome del codice di blocco
	Identificatore	Nome del sottoprogramma

Esempio

```
//B(PROG1)                          ; Inizio blocco
SUB(UP1)                            ; Inizio del sottoprogramma
  ...
  REG[0] = 5                        ; Assegnazione del valore 5 al registro 0
  ...
END_SUB                             ; Fine del sottoprogramma
SUB(UP2)                            ; Inizio del sottoprogramma
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN
  ENDIF
  VAR1.val="Otto"
END_SUB                             ; Fine del sottoprogramma
//END                               ;Fine blocco
```

8.3.4 Check Variable (CVAR)

Descrizione

Con l'aiuto della funzione CVAR (Check Variable) è possibile richiedere se tutte o solo particolari variabili o variabili ausiliarie di una finestra di dialogo siano corrette.

La verifica del contenuto delle variabili può essere sensata, ad esempio, prima di generare il codice NC con la funzione GC.

Una variabile è corretta se lo stato della variabile Identificatore.vld = 1.

Programmazione

Sintassi:	**CVAR**(VarN)	
Descrizione:	Verifica della validità del contenuto delle variabili	
Parametri:	VarN	Elenco delle variabili da verificare. È possibile verificare fino a 29 varibili separate dalla virgola. La lunghezza massima di caratteri da rispettare è pari a 500. Il risultato dell'interrogazione può essere:
	1 =	TRUE (tutte le variabili hanno un contenuto valido)
	0 =	FALSE (almeno una variabile non ha un contenuto valido)

Esempio

```
IF CVAR == TRUE                          ; Verifica di tutte le variabili
```

```
  VS8.SE = 1                                ; Se tutte le variabili sono corrette, il
                                            softkey VS8 è visibile.
ELSE
  VS8.SE = 2                                ; Se una variabile contiene un valore non
                                            corretto, il softkey VS8 non è utilizza-
                                            bile.
ENDIF

IF CVAR("VAR1", "VAR2") == TRUE
                                            ; Verifica delle variabili VAR1 e VAR2
  DLGL ("VAR1 e VAR2 sono OK")
                                            ; Se VAR1 e VAR2 sono completate in modo
                                            corretto, nella riga di dialogo viene vi-
                                            sualizzato "VAR1 e VAR2 sono OK"
ELSE
    DLGL ("VAR1 e VAR2 non sono OK")
                                            ; Se VAR1 e VAR2 non sono completate in
                                            modo corretto, nella riga di dialogo vie-
                                            ne visualizzato "VAR1 e VAR2 non sono OK"
ENDIF
```

8.3.5 Funzione file Copy Program (CP)

Descrizione

La funzione CP (Copy Program) copia i file nel file system HMI oppure nel file system NC.

Programmazione

Sintassi:	**CP**("File sorgente", "File di destinazione")	
Descrizione:	Copia file	
Parametri:	File sorgente	Percorso completo del file sorgente
	File di destinazione	Percorso completo del file di destinazione

Il valore di ritorno (VAR1, definito come variabile ausiliaria) consente di chiedere se la funzione è stata eseguita correttamente:

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

Esempio

Caso applicativo con valore di ritorno:

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

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

Caso applicativo senza valore di ritorno:

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

Vedere anche

Supporto di FILE_ERR: Variabile FILE_ERR (Pagina 108)

8.3.6 Funzione file Delete Program (DP)

Descrizione

La funzione DP (Delete Program) elimina un file del file system HMI passivo oppure del file system dell'NC attivo.

Programmazione

Sintassi:	**DP**("File")	
Descrizione:	File:cancellazione	
Parametri:	File	Indicazione completa del percorso del file da cancellare

Esempio

Per questa funzione viene utilizzata la seguente sintassi della gestione dati:

- con valore di ritorno

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

  VAR1 = 0 Il file è stato eliminato.

  VAR1 = 1 Il file non è stato eliminato.
- senza valore di ritorno:

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

8.3.7 Funzione file Exit Program (EP)

Descrizione

La funzione EP (Exist Program) verifica se nel file system dell'NC oppure nel file system HMI sia presente un particolare programma dell'NC al percorso indicato.

Programmazione

Sintassi:	**EP**("File")	
Descrizione:	Verifica dell'esistenza del programma dell'NC	
Parametri:	File	Percorso completo del file nel file system dell'NC o dell'HMI
Valore di ritorno:	Nome di una variabile a cui deve essere associato il risultato dell'interrogazione.	

La funzione EP si avvale della nuova sintassi e della vecchia logica (con sintassi adattata).

Il file viene richiamato direttamente con un nome qualificante:

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

oppure
`CF_CARD: /MPF.DIR/XYZ.MPF` (punta a /user/sinumerik/data/prog)
o
`LOC:` (corrisponde a CF_CARD)

Esempi di nuova sintassi:

```
EP("//NC/WKS.DIR/TEST.WPD/XYZ.MPF",VAR1)
EP("CF_CARD:/mpf.dir/XYZ.MPF",VAR1)
EP("LOC:/mpf.dir/XYZ.MPF",VAR1)
;con valore di ritorno:
; VAR1 = 0, il file esiste.
; VAR1 = 1, il file non esiste.
```

Esempi di vecchia sintassi:

```
EP("/MPF.DIR/CFI.MPF", VAR1)
;con valore di ritorno:
; VAR1 = M, il file si trova nel file system HMI.
; VAR1 = , il file si trova nel file system NC.
; VAR1 = B, il file si trova nei file system HMI e NC.
```

Esempio

```
EP("/MPF.DIR/GROB.MPF",VAR1)                                    ; vecchio - percorso ora con / invece di
                                                                \


IF VAR1 == "M"
   DLGL("Il file si trova nel file system HMI")
ELSE
   IF VAR1 == "N"
      DLGL("Il file si trova nel file system NC")
   ELSE
     DLGL("Il file non si trova né nel file system dell'HMI
né in quello dell'NC")
   ENDIF
ENDIF
```

8.3.8 Funzione file Move Program (MP)

Descrizione

La funzione MP (Move Program) copia i file nell'ambito del file system HMI o del file system NC.

Programmazione

Sintassi:	**MP**("sorgente", "destinazione")	
	`MP("CF_CARD:/MPF.DIR/MYPROG.MPF","//NC/MPF.DIR")`	
Descrizione:	Spostamento file	
Parametri:	File sorgente	Indicazione completa del percorso
	File di destinazione	Indicazione completa del percorso
Valore di ritorno	Risultato dell'interrogazione	

Esempi

Con valore di ritorno:

```
MP("//NC/MPF.DIR/123.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)        ;Nell'ambito NC
MP("//NC/MPF.DIR/123.MPF",VAR0,VAR1)                           ;Destinazione tramite variabile
MP(VAR4,VAR0,VAR1)                                             ;Sorgente e destinazione tramite va-
                                                                riabile
MP("CF_CARD:/mpf.dir/myprog.mpf","//NC/MPF.DIR/123.MPF",VAR1)  ;Da scheda CF a NC
```

```
MP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/XYZ/123.mpf",VAR1)          ;Da NC a scheda CF

;con valore di ritorno:
; VAR1 = 0, eseguito
; VAR1 = 1, non eseguito
```

8.3.9 Funzione file Select Program (SP)

Descrizione

La funzione SP (Select Program) seleziona un file del file system dell'NC attivo per elaborarlo; ciò significa che il file deve essere prima caricato nell'NC.

Programmazione

Sintassi:	**SP**("file")	
Definizione:	Selezione di un programma	
Parametri:	"File"	Indicazione completa del percorso del file NC

Esempio

Per questa funzione viene utilizzata la seguente sintassi della gestione dati:

- con valore di ritorno

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

  VAR1 = 0 Il file è stato caricato.

  VAR1 = 1 Il file non è stato caricato.
- senza valore di ritorno:

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

8.3.10 Accessi ai file: RDFILE, WRFILE, RDLINEFILE, WRLINEFILE

Descrizione

Per l'accesso in lettura e scrittura ai file con la sintassi INI sono disponibili le funzioni RDFILE e WRFILE.

Per l'accesso in lettura e scrittura a righe singole di un file sono disponibili le funzioni RDLINEFILE e WRLINEFILE.

Programmazione

Sintassi:	**RDFILE**("filename + percorso", "section", "key")	
Descrizione:	Lettura da un file	
Parametri:	Filename + percorso	Nome percorso e nome file
	Section	Sezione nei file INI
	Key	Chiave nei file INI

Sintassi:	**WRFILE**(Value, "filename + percorso", "section", "key")	
Descrizione:	Scrittura in un file	
Parametri:	Value	Valore da scrivere
	Filename + percorso	Nome percorso e nome file
	Section	Sezione nei file INI
	Key	Chiave nei file INI

Sintassi:	**RDLINEFILE**("filename + percorso", numero di riga)	
Descrizione:	Lettura di una riga da un file	
Parametri:	Filename + percorso	Nome percorso e nome file
	Numero di riga	Numero di riga La numerazione inizia con 0 per a prima riga.

Sintassi:	**WRLINEFILE**(valore, "filename + percorso")	
Descrizione:	Scrittura di una riga alla fine di un file	
Parametri:	Valore	Valore da scrivere
	Filename + percorso	Nome percorso e nome file

Nota

- I file non devono trovarsi nel file system dell'NC (gestione dati).
- Se il file non esiste, se viene raggiunta la fine del file o si verificano altri errori, viene conseguentemente impostata la variabile FILE_ERR ed ERR. L'esistenza di un file può essere verificata in precedenza con la funzione file Exist Program (EP).
- Il file elaborato viene generato nella codifica "UTF-8" (senza BOM (Byte Order Mask). Per la lettura di un file, lo stesso viene richiesto nella codifica "UTF-8".
- Con la funzione file Delete Program (DP) è possibile all'occorrenza cancellare il file esplicitamente.

Esempi

Lettura da un file INI:

Presupposto/ipotesi:

Contenuto del file "C:/tmp/myfile.ini":

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

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

Risultato:

MyVar contiene ora il valore "Daniel".

Scrittura in un file INI:

Presupposto/ipotesi:

VARS=12

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

Risultato:

Contenuto del file "C:/tmp/myfile.ini":

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

Lettura della 4ª riga di un file:

Presupposto/ipotesi:

Contenuto di c:/tmp/myfile.mpf:

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

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

Risultato:

MyVar contiene ora il valore " R[0]=R[0]+1"

Scrittura alla fine di un file:

Presupposto/ipotesi:

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

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

Risultato:

Contenuto di c:/tmp/mypp.mpf:

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

8.3.11 Dialog Line (DLGL)

Descrizione

Nella riga di dialogo della finestra di dialogo possono essere visualizzati, a seconda delle particolari situazioni, brevi testi (segnalazioni o istruzioni per l'immissione).

Possibile numero di caratteri in caso di dimensione carattere standard: ca. 50 caratteri

Nota

Con i metodi SUSPEND e RESUME non si possono progettare le funzioni LS, LM, DLGL, EXIT e EXITLS. Se si utilizzano le funzioni in questi metodi, l'esecuzione delle stesse viene impedita.

Programmazione

Sintassi:	**DLGL**("Stringa")	
Descrizione:	Visualizzazione testo nella riga di dialogo	
Parametri:	String	di testo che viene visualizzata nella riga di dialogo

Esempio

```
IF Var1 > Var2
                              ; Nella riga di dialogo viene visualizzato il testo
                              "Valore troppo grande" se Variabile1>Variabile2.
  DLGL("Valore troppo gran-
de!")
ENDIF
```

8.3.12 DEBUG

Descrizione

La funzione DEBUG rende disponibile un'ausilio all'analisi nella fase di progettazione delle maschere utente Run MyScreen. Con la funzione DEBUG viene valutata un'espressione inserita in parentesi in runtime. Il risultato viene aggiunto nel logbook "easyscreen_log.txt" come voce autonoma.

Ogni voce viene fatta precedere dal time stamp attuale tra parentesi quadre (vedere l'esempio seguente).

Si raccomanda di rimuovere nuovamente gli output di DEBUG, quando possibile, nelle parti con criticità temporale. In particolare una volta conclusa la fase di progettazione, si raccomanda di commentare gli output di DEBUG. Gli accessi in scrittura al logbook "easyscreen_log.txt", le cui dimensioni continuano ad aumentare, provocano un rallentamento.

Programmazione

Sintassi:	**DEBUG**(espressione))
Descrizione:	Effettuazione della registrazione nel logbook "easyscreen_log.txt"
Parametri:	Espressione da valutare, da cui viene generata una voce nel logbook

Esempio

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

8.3.13 Uscita dalla finestra di dialogo (EXIT)

Descrizione

Con la funzione EXIT si esce da una finestra di dialogo e si torna alla finestra di dialogo principale. Nel caso in cui non esista una finestra di dialogo principale, uscire dalla nuova interfaccia operativa creata e tornare all'applicazione standard.

Nota

Con i metodi SUSPEND e RESUME non si possono progettare le funzioni LS, LM, DLGL, EXIT e EXITLS. Se si utilizzano le funzioni in questi metodi, l'esecuzione delle stesse viene impedita.

Programmazione (senza parametri)

Sintassi:	**EXIT**
Descrizione:	Uscita dalla finestra di dialogo
Parametri:	- nessuno -

Esempio

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

Descrizione

Se la finestra di dialogo corrente è stata richiamata tramite una variabile di trasferimento, è possibile modificare il valore della variabile e tornare alla finestra di dialogo di partenza.

I valori delle variabili vengono sempre assegnati alle variabili trasferite dalla finestra di dialogo di partenza alla finestra di dialogo seguente attraverso la funzione "LM". È possibile trasferire fino a 20 valori delle variabili separati dalla virgola.

Nota

La successione delle variabili o dei valori delle variabili deve corrispondere alla successione delle variabili di trasferimento della funzione LM, in modo che l'assegnazione risulti univoca. Se non vengono impostati alcuni valori di variabili, le relative variabili di trasferimento non vengono modificate. Le variabili di trasferimento modificate sono valide immediatamente dopo la funzione LM nella finestra di dialogo di partenza.

Programmazione con variabile di trasferimento

Sintassi:	**EXIT**[(VARx)]	
Descrizione:	Uscita dalla finestra di dialogo con trasferimento di una o più variabili	
Parametri:	VARx	Definizione delle variabili

Esempio

```
//M(Maschera1)
...
PRESS(HS1)
  LM("MASHERA2","CFI.COM",1, POSX, POSY,
DIAMETRO)
                                        ;Interruzione maschera1 e visualizzazione
                                        maschera2. Trasferimento variabili POSX,
                                        POSY e DIAMETRO.
  DLGL("Maschera2 chiusa")              ; Una volta usciti dalla maschera2, nella
                                        riga di dialogo della maschera1 viene vi-
                                        sualizzato il testo: maschera2 chiusa.
END_PRESS
...
//END

//M(Maschera2)
...
PRESS(HS1)
  EXIT(5, , DIAMETRO_CALCOLATO)
                                        ; Uscita dalla maschera2 e ritorno alla
                                        maschera1 nella riga successiva a LM. As-
                                        segnazione del valore 5 alla variabile
                                        POSX e del valore della variabile DIAME-
                                        TRO_CALCOLATO alla variabile DIAMETRO. La
                                        variabile POSY mantiene il valore attuale.
END_PRESS
...
//END
```

8.3.14 Manipolazione dinamica delle liste dei campi di toggle o ListBox

Descrizione

Le funzioni LISTADDITEM, LISTINSERTITEM, LISTDELETEITEM e LISTCLEAR servono alla manipolazione dinamica delle liste dei campi di toggle o di ListBox.

Queste funzioni hanno effetto solo sulle variabili che possiedono una propria lista, quali ad es.

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

Se la variabile punta ad un array, ad es. `DEF VAR_AC3 = (I/* MYARRAY)`, queste funzioni non sono disponibili perché altrimenti l'array globale potrebbe essere modificato.

Una variabile deve avere un valore definito perlomeno nella riga DEF. In questo modo viene stabilito il tipo lista "semplice" o "estesa".

Successivamente è consentito comunque cancellare completamente la lista ed eventualmente crearla completamente ex novo. Il tipo "semplice" o "esteso" deve però essere mantenuto ovvero non può essere modificato in modo dinamico.

Programmazione

Sintassi:	**LISTINSERTITEM**(nome di variabile, posizione, ItemValue[, ItemDispValue])	
Descrizione:	Inserimento di un elemento in una determinata posizione da un file	
Parametri:	Nome della variabile	
	Posizione	Posizione in corrispondenza della quale un elemento deve essere aggiunto nella lista
	ItemValue	Valore della voce della lista
	ItemDispValue	Valore come deve essere rappresentato nella lista

Sintassi:	**LISTADDITEM**(nome di variabile, ItemValue[, ItemDispValue])	
Descrizione:	Aggiunta di un elemento alla fine della lista	
Parametri:	Nome della variabile	
	ItemValue	Valore della voce della lista
	ItemDispValue	Valore come deve essere rappresentato nella lista

Sintassi:	**LISTDELETEITEM**(nome di variabile, posizione)	
Descrizione:	Cancellazione di un determinato elemento	
Parametri:	Nome della variabile	
	Posizione	Posizione dell'elemento da cancellare nella lista

Sintassi:	**LISTCOUNT**(nome di variabile)	
Descrizione:	Restituisce il numero attuale degli elementi della lista	
Parametri:	Nome della variabile	

Sintassi:	**LISTCLEAR**(nome di variabile)	
Descrizione:	Cancellazione della lista completa	
Parametri:	Nome della variabile	

Esempi

Nota

I seguenti esempi si basano l'uno sull'altro. Per immaginare i rispettivi risultati è essenziale la sequenza degli esempi.

Presupposto/ipotesi:

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

Aggiunta di un elemento -1="Undefined" alla fine della lista:

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

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

Inserimento di un elemento 99="Maybe" nella posizione 2:

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

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

Determinazione del numero attuale degli elementi della lista:

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

Risultato: REG[10] = 4

Cancellazione dell'elemento nella posizione 1:

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

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

Cancellazione della lista completa:

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

Risultato: la lista è vuota

8.3.15 Evaluate (EVAL)

Descrizione

La funzione EVAL interpreta un'espressione trasmessa e la esegue. In questo modo le espressioni possono essere create durante l'elaborazione ciclica. Questo è utile ad esempio per accessi indicizzati alle variabili.

Programmazione

Sintassi:	**EVAL**(exp)	
Descrizione:	Valutazione espressione	
Parametri:	exp	Espressione logica

Esempio

```
VAR1=(S)
VAR2=(S)
VAR3=(S)
VAR4=(S)
CHANGE()
  REG[7] = EVAL("VAR"<<REG[5])  ; L'espressione tra parentesi indica VAR3 se il va-
                                lore di REG[5] è uguale a 3. A REG[7] viene quindi
                                assegnato il valore di VAR3.
  IF REG[5] == 1
    REG[7] = VAR1
  ELSE
    IF REG[5] == 2
      REG[7] = VAR2
    ELSE
      IF REG[5] == 3
        REG[7] = VAR3
      ELSE
        IF REG[5] == 4
          REG[7] = VAR4
        ENDIF
      ENDIF
```

```
    ENDIF
  ENDIF
END_CHANGE
```

8.3.16 Exit Loading Softkey (EXITLS)

Descrizione

Con la Funzione EXITLS si esce dall'interfaccia operativa corrente e viene caricata una barra dei softkey definita.

Nota

Con i metodi SUSPEND e RESUME non si possono progettare le funzioni LS, LM, DLGL, EXIT e EXITLS. Se si utilizzano le funzioni in questi metodi, l'esecuzione delle stesse viene impedita.

Programmazione

Sintassi:	**EXITLS**("Barra dei softkey"[, "Nome del percorso"])	
Descrizione:	All'uscita, caricamento della barra dei softkey	
Parametri:	Barra softkey	Nome della barra dei softkey da caricare
	Nome del percorso	Percorso della directory della barra softkey da caricare

Esempio

```
PRESS(HS1)
    EXITLS( "Barra1", "AEDITOR.COM" )
END_PRESS
```

8.3.17 Function (FCT)

Descrizione

Le funzioni esterne vengono inserite in un file DLL e rese note attraverso una registrazione nella riga di definizione del file di progettazione.

Nota

La funzione esterna deve avere almeno un parametro di ritorno.

Programmazione

Sintassi:	**FCT**Nome della funzione = ("File"/Tipo di ritorno/Tipo di parametri fissi/Tipo di parametri variabili)	
	FCT InitConnection = ("c:\tmp\xyz.dll"/I/R,I,S/I,S)	
Descrizione:	Il richiamo di una funzione esterna può essere eseguito ad es. dal metodo LOAD o nel metodo PRESS.	
Parametri:	Nome della funzione	Nome delle funzioni esterne
	File	Percorso completo del file DLL
	Tipo di ritorno	Tipo di dati del valore di ritorno
	Tipo di parametri fissi	Parametri Value
	Tipo di parametri variabili	Parametri di riferimento
	I tipi di dati vengono separati con la virgola.	

Il richiamo della funzione esterna può essere eseguito ad es. dal metodo LOAD o nel metodo PRESS.

Esempio:

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

Struttura della funzione esterna

La funzione esterna deve rispettare una determinata firma preimpostata:

Sintassi:	extern "C" dllexport void InitConnection (ExtFctStructPtr FctRet, ExtFctStructPtr FctPar, char cNrFctPar)	
Descrizione:	Esportazione DLL solo in caso di implementazione Windows Specificatori e parametri di trasferimento sono predefiniti in modo fisso. Con le strutture trasferite vengono trasmessi anche gli effettivi parametri di richiamo.	
Parametri:	cNrFctPar	Numero dei parametri di richiamo = numero degli elementi di struttura in FctPar
	FctPar	Indicatore su un campo di elementi di struttura che contengono i parametri di richiamo con il tipo di dati.
	FctRet	Indicatore su una struttura per la restituzione del valore della funzione con tipo di dati.

Definizione della struttura di trasferimento

```
union CFI_VARIANT
    (
    char                                b;
    short int                           i;
```

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

Se la funzione esterna deve essere sviluppata indipendentemente dalla piattaforma (Windows, Linux), non va utilizzata la parola chiave __declspec(dllexport). Questa parola chiave è necessaria solo in Windows. In Qt si può utilizzare ad esempio la macro seguente:

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

La dichiarazione della funzione è la seguente:

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

Se le schermate progettate con "Run MyScreens" vengono utilizzate su NCU e PCU/PC, l'estensione del file binario deve essere omessa:

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

Se si omettono le informazioni assolute sul percorso, "Run MyScreens" cerca in un primo tempo il file binario nella directory progettata.

8.3.18 Generate Code (GC)

Descrizione

La funzione GC (Generate Code) genera un codice NC dal metodo OUTPUT.

Programmazione

Sintassi:	**GC**("Identificatore"[,"File di destinazione"][,Opz],[Append])
Descrizione:	Generazione del codice NC (la funzione GC è solo possibile all'interno dell'NC)

Parametri:	Identificatore	Nome del metodo OUTPUT che funge da base per la generazione del codice
	File di destinazione	Indicazione del percorso del file di destinazione per il file system HMI o NC. Se il file di destinazione non è stato indicato (possibile solo all'interno del supporto alla programmazione), il codice viene scritto nel punto in cui si trova il cursore all'interno del file attualmente aperto.
	Opz	Opzione per la generazione del commento
	0:	(Preimpostazione) creazione di un codice con commento per la possibilità di decompilazione (vedere anche Decompilare (Pagina 173)).
	1:	Non creare commenti nel codice generato. **Nota:** Questo codice non può essere decompilato (vedere anche Decompilare (Pagina 173)).
	Append	Questo parametro è significativo solo se è stato impostato un file di destinazione.
	0:	(preimpostazione) Se il file è già esistente, il vecchio contenuto viene cancellato.
	1:	Se il file è già esistente, il nuovo codice viene scritto all'inizio del file.
	2:	Se il file è già esistente, il nuovo codice viene aggiunto alla fine del file.

Esempio

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

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                ;Scrittura del codice NC dal metodo OUTPUT nel
                                    file \MPF.DIR\MESSEN.MPF:
                                    Cycle123(123, -6)
                                    M30
END_PRESS
```

Memorizzazione del file di destinazione

La funzione GC è solo possibile all'interno dell'NC. Per lo spostamento del file utilizzare le funzioni CP risp. MP. Vedere al riguardo i seguenti esempi:

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

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

Decompilazione

- **Nessuna indicazione del file di destinazione:**
  La funzione GC può essere utilizzata solo nel supporto alla programmazione e scrive il codice NC nel file attualmente aperto con l'editor. La decompilazione del codice NC è possibile. Se la funzione GC viene progettata in "Run MyScreens" senza indicare il file di destinazione, durante l'esecuzione si verifica una segnalazione di errore.
- **Indicazione del file di destinazione:**
  Il codice generato dal metodo OUTPUT viene inserito nel file di destinazione. Se il file di destinazione non esiste, viene creato nel file system NC. Se il file di destinazione è presente nel file system HMI, il file viene creato sul disco rigido. Le righe di commento operativo (informazioni importanti per la decompilazione) non vengono create, ossia la decompilazione non è possibile.

Particolarità durante la decompilazione

Nelle sottofinestre di dialogo non è possibile richiamare la funzione GC, in quanto nelle sottofinestre di dialogo possono essere utilizzate variabili che derivano da finestre di dialogo principali, ma che non sarebbero disponibili in caso di richiamo diretto.

In caso di interventi manuali sul codice generato tramite l'editor, il numero di caratteri per i valori creati attraverso la generazione del codice non deve essere modificato. Un'eventuale modifica ne impedirebbe la decompilazione.

Rimedio:

1. Decompilazione
2. Immettere la modifica con l'aiuto della finestra di dialogo progettata (ad es. 99 → 101)
3. GC

8.3.19 Funzioni per password

Panoramica

Sono disponibili le seguenti funzioni per password:

- Impostazione della password
- Cancellazione della password
- Modifica della password

Funzione HMI_LOGIN: Impostare una nuova password

Sintassi:	**HMI_LOGIN**(Passwd)	
Descrizione:	Con la funzione HMI_LOGIN() viene inviata una password all'NCK che imposta il livello di accesso attuale.	
Parametri:	Passwd	Password

Esempio

```
REG[0] = HMI_LOGIN("CUSTOMER")          ; Risultato = TRUE se password correttamente
                                        impostata, altrimenti FALSE
```

Funzione HMI_LOGOFF: Cancellazione della password

Sintassi:	**HMI_LOGOFF**
Descrizione:	Con la funzione HMI_LOGOFF è possibile ripristinare il livello di accesso attuale.
Parametri:	- nessuno -

Esempio

```
REG[0] = HMI_LOGOFF                     ; Risultato = TRUE se password correttamente
                                        cancellata, altrimenti FALSE
```

Funzione HMI_SETPASSWD: Modifica della password

Sintassi:	**HMI_SETPASSWD**(AC, Passwd)	
Descrizione:	Con la funzione HMI_SETPASSWD è possibile modificare la password del livello attuale o di un livello inferiore di password.	
Parametri:	AC	Livello di accesso
	Passwd	Password

Esempio

```
REG[0] = HMI_SETPASSWD(4,"MYPWD")      ; Risultato = TRUE se password correttamente
                                       modificata, altrimenti FALSE
```

8.3.20 Load Array (LA)

Descrizione

Con la funzione LA (Load Array) è possibile caricare un array da un altro file.

Programmazione

Sintassi:	**LA**(Identificatore [, File])	
Descrizione:	Caricamento dell'array dal file	
Parametri:	Identificatore	Nome dell'array da ricaricare
	File	File nel quale è definito l'array

Nota

Se un array nel file di progettazione attuale deve essere sostituito da un array di altro file di progettazione, i due array devono avere lo stesso nome.

Esempio

```
                                       ; Estratto dal file maske.com
DEF VAR2 = (S/*ARR5/"Off"/,"Campo di tog-
gle")
PRESS(HS5)
  LA("ARR5","arrayext.com")            ; Caricamento dell'array ARR5 dal file ar-
                                       rayext.com
  VAR2 = ARR5[0]                       ; Al posto di "Off"/"On" viene visualiz-
                                       zato nel campo di toggle di VAR2
                                       "In alto"/"In basso"/"A destra"/"A sini-
                                       stra"
END_PRESS
//A(ARR5)
("Off"/"On")
//END
                                       ; Estratto dal file arrayext.com
//A(ARR5)
```

```
("In alto"/"In basso"/"A destra"/"A sini-
stra")
//END
```

Nota

Tenere in considerazione che ad una variabile deve essere assegnato un valore valido dopo che è stata utilizzata la funzione LA per assegnare un altro array al campo di Toggle della variabile.

8.3.21 Load Block (LB)

Descrizione

Con la funzione LB (Load Block) è possibile caricare blocchi con sottoprogrammi durante il tempo di esecuzione. Preferibilmente LB dovrebbe essere programmato in un metodo LOAD in modo che i sottoprogrammi caricati possano essere richiamati in un qualsiasi momento.

Nota

I sottoprogrammi possono essere definiti anche direttamente in una finestra di dialogo; in questo caso, però, non devono poi essere caricati.

Programmazione

Sintassi:	**LB**("Nome del blocco"[,"File"])	
Descrizione:	Caricamento di un sottoprogramma durante il tempo di esecuzione	
Parametri:	nome del blocco	Nome del codice di blocco
	File	Indicazione del percorso del file di progettazione Preimpostazione = File di progettazione corrente

Esempio

```
LOAD
  LB("PROG1")             ; Il blocco "PROG1" viene ricercato nel file di progetta-
                          zione corrente e successivamente caricato.
  LB("PROG2","XY.COM")    ; Il blocco "PROG2" viene ricercato nel file di progetta-
                          zione XY.COM e successivamente caricato.
END_LOAD
```

8.3.22 Load Mask (LM)

Descrizione

Con la funzione LM viene caricata una nuova finestra di dialogo. A seconda della modalità di passaggio tra finestre di dialogo (vedere la tabella seguente), con questa funzione è anche possibile realizzare una finestra di dialogo Message.

Nota

Con i metodi SUSPEND e RESUME non si possono progettare le funzioni LS, LM, DLGL, EXIT e EXITLS. Se si utilizzano le funzioni in questi metodi, l'esecuzione delle stesse viene impedita.

Finestra di dialogo principale / sottofinestra di dialogo

Una finestra di dialogo che richiama un'altra finestra di dialogo senza essere a sua volta chiusa, viene definita finestra di dialogo principale. Una finestra di dialogo richiamata da una finestra di dialogo principale viene definita sottofinestra di dialogo.

Programmazione

Sintassi:	**LM**("Identificatore"[,"File"] [,MSx [, VARx]])
Descrizione:	Caricamento della finestra di dialogo

Parametri:	Identificatore	Nome della finestra di dialogo da caricare
	File	Indicazione del percorso (system file HMI o NC) del file di progettazione, impostazione standard: Dati di progettazione correnti
	MSx	Modalità di passaggio tra finestre di dialogo
	0:	(preimpostazione) La finestra di dialogo corrente viene chiusa, la nuova finestra di dialogo viene caricata e visualizzata. Con EXIT si torna all'applicazione standard. Con il parametro MSx è possibile stabilire se nel passaggio tra finestre di dialogo la finestra di dialogo corrente debba essere chiusa oppure no. Se la finestra di dialogo corrente viene mantenuta, è possibile trasferire variabili nella nuova finestra di dialogo. Il vantaggio del parametro MSx consiste nel fatto che le finestre di dialogo non debbano sempre essere reinizializzate ad ogni passaggio, ma che i dati e il layout della finestra di dialogo corrente vengano mantenuti facilitando il trasferimento dei dati.
	1:	La finestra di dialogo principale corrente viene interrotta a partire dalla funzione LM e la nuova sottofinestra di dialogo viene caricata e visualizzata (ad es. per la realizzazione di una finestra di dialogo Message). Con EXIT viene chiusa la sottofinestra di dialogo e si torna al punto di interruzione della finestra di dialogo principale. Nella finestra di dialogo principale, in caso di interruzione il blocco UNLOAD non viene elaborato.
	VARx	Presupposto: MS1 Elenco delle variabili che possono essere trasferite dalla finestra di dialogo principale alla sottofinestra di dialogo. È possibile trasferire fino a 20 variabili separate dalla virgola.

Nota

Il parametro VARx trasferisce sempre solo il valore della variabile, ciò significa che nella sottofinestra di dialogo le variabili possono essere lette e scritte, ma non sono visibili. La restituzione della variabile dalla sottofinestra di dialogo alla finestra di dialogo principale è possibile con la funzione EXIT.

Esempio

MASCHERA1(Finestra di dialogo principale): Salto alla sottofinestra di dialogo MASCHERA2 con trasferimento di variabile

```
PRESS(HS1)

  LM("MASHERA2","CFI.COM",1, POSX, POSY, DIAMETRO)
                                    ; Interruzione maschera1 e visualizzazione ma-
                                    schera2: Trasferimento variabili POSX, POSY e
                                    DIAMETRO.
  DLGL("Maschera2 chiusa")          ; Una volta usciti dalla maschera2, nella riga
                                    di dialogo della maschera1 viene visualizzato
                                    il testo: maschera2 chiusa.
```

```
MASCHERA1(Finestra di dialogo principale): Salto alla sottofinestra di dialogo MA-
SCHERA2 con trasferimento di variabile
END_PRESS
```

```
MASCHERA2(Sottofinestra di dialogo): ritorno alla finestra di dialogo principale MA-
SCHERA1 con trasferimento dei contenuti della variabile
PRESS(VS8)

  EXIT(POSX, POSY, DIAMETRO)
                                    ; sfumare maschera2 e continuare maschera1:
                                    vengono trasferite le variabili POSX, POSY e
                                    DIAMETRO.
END_PRESS
```

8.3.23 Load Softkey (LS)

Descrizione

Con la funzione LS è possibile visualizzare un'altra barra dei softkey.

Nota

Con i metodi SUSPEND e RESUME non si possono progettare le funzioni LS, LM, DLGL, EXIT e EXITLS. Se si utilizzano le funzioni in questi metodi, l'esecuzione delle stesse viene impedita.

Programmazione

Sintassi:	**LS**("Identificatore"[, "File"][, Merge])	
Descrizione:	Visualizzazione della barra dei softkey	
Parametri:	Identificatore	Nome della barra dei softkey
	File	Percorso (file system HMI oppure file system dell'NC) del file di progettazione Preimpostazione: file di progettazione attuale
	Merge	
	0:	Tutti i softkey esistenti vengono eliminati, i nuovi softkey progettati vengono inseriti.
	1:	Preimpostazione Solo i nuovi softkey progettati sovrascrivono i softkey esistenti, gli altri softkey (ovvero quelli dell'applicazione HMI) mantengono la loro funzionalità e il testo.

Esempio

```
PRESS(HS4)
  LS("Barra2",,0)     ; Barra2 sovrascrive la barra dei softkey esistente, i softkey
                      visualizzati vengono cancellati.
END_PRESS
```

Nota

Finché l'interprete non ha ancora visualizzato alcuna finestra di dialogo, ossia non è stata ancora elaborata alcuna funzione LM, nel metodo PRESS del blocco di definizione del softkey di accesso e della barra dei softkey è possibile progettare rispettivamente solo un comando LS o LM e nessun'altra azione.

Le funzioni LS e LM possono essere richiamate esclusivamente all'interno di un metodo PRESS di softkey, e comunque non come reazione ai tasti di navigazione (PU, PD, SL, SR, SU, SD).

8.3.24 Load Grid (LG)

Descrizione

La descrizione della tabella (Grid) può essere realizzata in modo dinamico all'interno dei metodi (come LOAD) attraverso il metodo LG.

Affinché una tabella possa essere assegnata con il metodo LG, la variabile deve essere già stata definita come variabile Grid e fare riferimento a una tabella valida presente.

Programmazione

Sintassi:	**LG**(nome grid, nome di variabile [,nome file])	
Descrizione:	Caricamento di una tabella	
Parametri:	Nome grid	nome della tabella (Grid) tra virgolette
	Nome della variabile	nome della variabile che deve essere assegnata alla tabella, tra virgolette
	nome del file	nome del file in cui è definita la tabella (Grid), tra virgolette. Da indicare solo se la tabella non è definita all'interno del file in cui è definita anche la variabile

Esempio

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

```
END_LOAD
```

Contenuto di mygrids.com:

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

8.3.25 Selezione multipla SWITCH

Descrizione

Con un comando SWITCH è possibile verificare diversi valori di una variabile.

L'espressione progettata nell'istruzione SWITCH viene confrontata in successione con tutti i valori progettati nelle seguenti istruzioni CASE.

In caso di discordanza viene eseguito un confronto con l'istruzione CASE successiva. Se segue un'istruzione DEFAULT e se finora nessuna istruzione CASE era uguale all'espressione progettata nell'istruzione SWITCH, vengono eseguite le istruzioni che seguono l'istruzione DEFAULT.

In caso di concordanza le istruzioni seguenti vengono eseguite finché segue un'istruzione CASE, DEFAULT o END_SWITCH.

Esempio

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

8.3.26 Multiple Read NC PLC (MRNP)

Descrizione

Con il comando MRNP si possono trasferire più variabili NC/PLC con un solo accesso al registro. Questo accesso è notevolmente più veloce di una lettura tramite singoli accessi. Le variabili NC/PLC incluse in un comando MRNP devono appartenere allo stesso settore.

I settori delle variabili NC/PLC sono suddivisi nel seguente modo:

- Dati NC generali ($MN..., $SN.., /nck/...)
- Dati NC specifici per canale ($MC..., $SC.., /channel/...)
- Dati PLC (DB..., MB.., /plc/...)
- Dati NC specifici per asse ($MA..., $SA..) e dello stesso asse

Programmazione

Sintassi:	**MRNP**(Nome della variabile1*Nome della variabile2[* ...], Indice registro)
Descrizione:	lettura di più variabili
Parametri:	Nel nome delle variabili il segno "*" funge da carattere separatore. Nella sequenza in cui i nomi delle variabili risultano nel comando vengono acquisiti i valori nei registri REG[Indice registro] e seguenti. In questo contesto vale quanto segue: il valore della prima variabile si trova in REG[indice registro] il valore della seconda variabile si trova in REG[indice registro + 1] ecc.

Nota

Prestare attenzione al fatto che il numero di registri è limitato e che l'elenco delle variabili è limitato a 500 caratteri.

Esempio

```
MRNP("$R[0]*$R[1]*$R[2]*$R[3]",1)      ;Da REG[1] a REG[4] la scrittura utilizza il
                                        valore delle variabili da $R[0] a $R[3].
```

Variabile NC

Sono disponibili tutti i dati macchina e di setting nonché il parametro R, ma solo particolari variabili di sistema (vedere anche: AUTOHOTSPOT).

Sono accessibili tutte le variabili utente globali (GUD) e specifiche per canale. Le variabili utente locali e globali del programma non possono essere elaborate.

Dati macchina	
Dato macchina globale	$MN_...
Dato macchina specifico per asse	$MA_...
Dato macchina specifico per canale	$MC_...

Dati setting	
Dato setting globale	$SN_...
Dato setting specifico per asse	$SA_...
Dato setting specifico per canale	$SC_...

Variabili di sistema	
Parametro R 1	$R[1]

Variabili PLC

Sono disponibili tutti i dati PLC

Dati PLC	
Byte y Bit z del blocco dati x	DBx.DBXy.z
Byte y del blocco dati x	DBx.DBBy
Word y del blocco dati x	DBx.DBWy
Doubleword y del blocco dati x	DBx.DBDy
Real y del blocco dati x	DBx.DBRy
Merker byte x Bit y	Mx.y
Merker byte x	MBx
Merker word x	MWx
Merker doubleword x	MDx
Byte di ingresso x Bit y	Ix.y oppure Ex.y
Byte di ingresso x	IBx oppure EBx
Word di ingresso x	IWx oppure EWx
Doubleword di ingresso x	IDx oppure EDx
Byte di uscita x Bit y	Qx.y oppure Ax.y
Byte di uscita x	QBx oppure ABx
Word di uscita x	QWx oppure AWx
Doubleword di uscita x	QDx oppure ADx
Stringa y con lunghezza x del blocco dati x	DBx.DBSy.z

8.3.27 P_LOGICAL_FILEPATH

Descrizione

La funzione P_LOGICAL_FILEPATH restituisce il nome e il percorso logico del programma pezzo in elaborazione nell'editor.

Nota

La funzione è solo disponibile nel contesto delle maschere del ciclo.

Programmazione

Sintassi:	P_LOGICAL_FILEPATH()	
Descrizione:	Determinazione del nome e del percorso reale del programma pezzo in elaborazione nell'editor ad es. "//NC/MPF.DIR/HUGO.MPF"	
Parametri:	- nessuno -	

Vedere anche

P_REAL_FILEPATH (Pagina 167)

8.3.28 P_REAL_FILEPATH

Descrizione

La funzione P_REAL_FILEPATH restituisce il nome e il percorso reale del programma pezzo in elaborazione nell'editor.

Nota

La funzione è solo disponibile nel contesto delle maschere del ciclo.

Programmazione

Sintassi:	P_REAL_FILEPATH()	
Descrizione:	Determinazione nell'editor del nome e del percorso reale del programma pezzo in elaborazione ad es. "/_N_MPF_DIR/_N_HUGO_MPF"	
Parametri:	- nessuno -	

Vedere anche

P_LOGICAL_FILEPATH (Pagina 167)

8.3.29 Servizi PI

Descrizione

Attraverso la funzione PI_START è possibile avviare servizi PI (servizi Program Instance) dal PLC in campo NC.

Nota

Una lista dei servizi PI disponibili si trova nel Manuale di guida alle funzioni Funzioni base.

Programmazione

Sintassi:	**PI_START**("stringa di trasferimento")	
Descrizione:	Esecuzione del servizio PI	
Parametri:	"Stringa di trasferimento"	Contrariamente a quanto specificato nella documentazione OEM, la stringa di trasferimento deve essere scritta tra doppie virgolette.

Esempio

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

Nota

I servizi PI dipendenti dal canale si riferiscono sempre al canale attuale.

Servizi PI delle funzioni utensili (campo TO), si riferiscono sempre al campo TO che viene assegnato al canale attuale.

8.3.30 Lettura (RNP) e scrittura (WNP) di variabili di sistema e variabili utente

Descrizione

Con l'istruzione RNP (Read NC PLC) è possibile leggere variabili NC o PLC oppure dati macchina.

Programmazione

Sintassi:	**RNP** ("Variabile di sistema o utente", valore)	
Descrizione:	Lettura della variabile NC o PLC o di dati macchina	
Parametri:	Variabile di sistema o utente	Nome della variabile NC o PLC
	Valore	Valore che deve essere scritto nella variabile di sistema o utente. Se il valore è di tipo String va scritto tra doppie virgolette.

Esempio

```
VAR2=RNP("$AA_IN[2]")         ;Lettura della variabile NC
```

Descrizione

Con l'istruzione WNP (Write NC PLC) è possibile scrivere variabili NC o PLC oppure dati macchina.

Accessi alle variabili NC/PLC vengono rieseguiti a ogni elaborazione della funzione WNP, ossia un accesso NC/PLC in un metodo CHANGE viene sempre eseguito. Questo è sensato quando una variabile di sistema o utente varia spesso il suo valore. Se un accesso NC/PLC deve essere eseguito una sola volta, ciò deve essere progettato in un metodo LOAD oppure UNLOAD.

Programmazione

Sintassi:	**WNP** ("Variabile di sistema o utente", valore)	
Descrizione:	Scrittura della variabile NC o PLC o di dati macchina	
Parametri:	Variabile di sistema o utente	Nome della variabile NC o PLC
	Valore	Valore che deve essere scritto nella variabile di sistema o utente. Se il valore è di tipo String va scritto tra doppie virgolette.

Esempio

```
WNP("DB20.DBB1",1)        ; Scrittura variabile PLC
```

8.3.31 RESIZE_VAR_IO e RESIZE_VAR_TXT

Descrizione

Con i comandi `RESIZE_VAR_IO()` e `RESIZE_VAR_TXT()` è possibile modificare la geometria del componente di ingresso/uscita o del componente di testo di una variabile.

Dopo l'impostazione della nuova geometria, tutti i campi della maschera vengono posizionati come se la maschera stessa fosse stata progettata a priori con queste posizioni. In questo modo tutti i campi, indipendentemente dalla progettazione, vengono allineati in modo reciprocamente corretto.

Programmazione

Sintassi:	**RESIZE_VAR_IO**(nome di variabile, [X], [Y], [larghezza], [altezza]) **RESIZE_VAR_TXT**(nome di variabile, [X], [Y], [larghezza], [altezza])	
Descrizione:	Modifica della geometria del componente di ingresso/uscita o del componente di testo di una variabile.	
Parametri:	Nome della variabile	Nome della variabile la cui geometria del componente di ingresso/uscita o del componente di testo deve essere modificata.
	X	Coordinata X in alto a sinistra
	Y	Coordinata Y in alto a sinistra
	Larghezza	Larghezza
	Altezza	Altezza

Nota

Per ogni valore non specificato di X, Y, larghezza o altezza viene mantenuto il valore precedente. Se si specifica -1 per uno di questi valori, viene impostato il componente della posizione standard predefinito da "Run MyScreens".

Esempio

```
RESIZE_VAR_IO("MyVar1", 200, , 100)    ; Il componente IO della variabile "MyVar1"
                                       viene spostato sulla posizione X di 200 pi-
                                       xel, la larghezza viene impostata a 100 pi-
                                       xel. La posizione Y e l'altezza del valore
                                       precedente vengono mantenute.
```

8.3.32 Register (REG)

Descrizione dei registri

I registri sono necessari allo scambio di dati tra le diverse finestre di dialogo. I registri sono sempre assegnati a una finestra di dialogo e vengono creati al caricamento della prima finestra di dialogo e associati a 0 o a una stringa vuota.

Nota

I registri non devono essere utilizzati direttamente in un metodo OUTPUT per la generazione del codice NC.

Programmazione

Sintassi:	**REG**[x]	
Descrizione:	Definizione di registri	
Parametri:	x	Indice registro con x = 0...19; Tipo: REAL oppure STRING = VARIANT I registri con x ≥ 20 sono già assegnati a Siemens.

Descrizione del valore del registro

L'abbinamento dei valori ai registri viene progettato in un metodo.

Nota

Se da una finestra di dialogo viene creata un'altra finestra di dialogo con la funzione LM, il contenuto dei registri viene automaticamente acquisito nella nuova finestra di dialogo e resta disponibile nella seconda finestra di dialogo per ulteriori calcoli.

Programmazione

Sintassi:	Identificatore.val = Valore registro oppure Identificatore= Valore del registro	
Descrizione:		
Parametri:	Identificatore	Nome del registro
	Valore del registro	Valore del registro

Esempio

```
UNLOAD
```

```
  REG[0] = VAR1          ; Assegnazione del valore della variabile 1 al registro 0
END_UNLOAD

UNLOAD
  REG[9].VAL = 84        ; Assegnazione del valore 84 al registro 9
END_UNLOAD
                         ; Nella finestra di dialogo seguente è poi possibile asse-
                         gnare nuovamente questi registri a variabili locali in un
                         metodo.
LOAD
  VAR2 = REG[0]
END_LOAD
```

Descrizione dello stato del registro

Con la proprietà Stato si può interrogare nella progettazione se il registro contiene un valore valido.

L'interrogazione dello stato del registro può essere utilizzata per scrivere un valore in un registro solo quando una finestra di dialogo viene utilizzata come finestra di dialogo principale.

Programmazione

Sintassi:	Identificatore.vld	
Descrizione:	Questa proprietà è di sola lettura.	
Parametri:	Identificatore	Nome del registro
Valore di ritorno:		Il risultato dell'interrogazione può essere:
	FALSE =	un valore non valido
	TRUE =	un valore valido

Esempio

```
IF REG[15].VLD == FALSE    ; Verifica della validità del valore del registro
  REG[15] = 84
ENDIF
VAR1 = REG[9].VLD          ; Assegnazione del valore dell'interrogazione dello stato
                           di REG[9] a Var1.
```

8.3.33 RETURN

Descrizione

Con la funzione RETURN si può interrompere in anticipo l'elaborazione del sottoprogramma attuale e ritornare al punto di esecuzione dell'ultima istruzione CALL.

Se nel sottoprogramma non è progettata la funzione RETURN, il sottoprogramma viene eseguito fino alla fine, quindi si torna al punto di esecuzione.

Programmazione

Sintassi:	**RETURN**	
Descrizione:	ritorno al punto di esecuzione	
Parametri:	- nessuno -	

Esempio

```
//B(PROG1)                ; Inizio blocco
SUB(UP2)                  ; Inizio del sottoprogramma
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
RETURN                    ; Se il valore della variabile è = Otto, alla variabile
                          viene assegnato il valore "Hans" e il sottoprogramma viene
                          terminato in questo punto.
ENDIF
  VAR1.val="Otto"         ; Se il valore della variabile è ≠ Otto, alla variabile
                          viene assegnato il valore "Otto".
END_SUB                   ; Fine del sottoprogramma
//END                     ;Fine blocco
```

8.3.34 Decompilare

Descrizione

Nel supporto alla programmazione è possibile **riconvertire** il codice NC ottenuto con la funzione GC e visualizzare nuovamente i valori delle variabili nel campo di input/output della finestra di dialogo di immissione relativa.

Programmazione

Dal codice NC le variabili vengono acquisite nella finestra di dialogo. In questo modo i valori delle variabili del codice NC vengono confrontati con i valori calcolati delle variabili del file di progettazione. Se i valori non coincidono viene emessa una segnalazione di errore nel file di log in quanto sono stati modificati dei valori nel codice NC generato.

Se una variabile è presente più volte nel codice NC, durante la decompilazione viene sempre utilizzato l'ultimo trovato. Inoltre viene emessa una segnalazione nel file di log.

Le variabili non utilizzate nel codice NC durante la generazione del codice sono memorizzate come commento operativo. Tutte le informazioni necessarie per la decompilazione vengono anch'esse identificate come commenti operativi. I commenti operativi non devono essere modificati.

Nota

Il blocco con codice NC e commenti operativi può essere decompilato solo se comincia all'inizio di una riga.

Esempi:

Nel programma è presente il seguente codice NC:

```
DEF VAR1 = (I//101)
OUTPUT(CODE1)
  "X" VAR1 " Y200"
  "X" VAR1 " Y0"
END_OUTPUT
```

Nel partprogram è stato inserito il seguente codice:

```
;NCG#TestGC#\cus.dir\aeditor.com#CODE1#1#3#
X101 Y200
X101 Y0
;#END#
```

L'editor durante la decompilazione legge il seguente codice:

```
X101 Y200
X222 Y0          ; Il valore per X è stato modificato nel partprogram (X101 → X222)
```

Nella finestra di dialogo di immissione viene prescritto il seguente valore per VAR1: `VAR1 = 222`

Vedere anche

Generate Code (GC) (Pagina 154)

8.3.35 Decompilazione senza commento operativo

Descrizione

Nel supporto alla programmazione è possibile **decompilare senza commenti operativi** il codice NC generato con la funzione GC e visualizzare nuovamente i valori delle variabili nel campo di input/output della finestra di dialogo di immissione relativa.

Programmazione

Per sopprimere righe di commento create con la regolare generazione del codice, è possibile eseguire il comando GC nel modo seguente:

```
GC("CODE1",D_NAME,1)
```

Il codice così creato non può essere regolarmente decompilato. Per poter tuttavia decompilare i richiami di cicli così creati, sono necessari i seguenti passaggi:

- **Ampliare il file "easyscreen.ini"**
  Nel file "easyscreen.ini" viene aggiunta la sezione `[RECOMPILE_INFO_FILES]`. In questa sezione vengono elencati tutti i file ini che contengono le descrizioni per i cicli da decompilare senza commento operativo:
  ```
  [RECOMPILE_INFO_FILES]
  IniFile01 = cycles1.ini
  IniFile02 = cycles2.ini
  ```
  Possono essere forniti più file ini i cui nomi sono liberamente selezionabili.
- **Creazione di un file ini per la descrizione dei cicli**
  Salvare il file ini con le descrizioni dei cicli nel seguente percorso:
  [cartella di sistema user]/cfg
  [cartella di sistema oem]/cfg
  [cartella di sistema addon]/cfg
  Per ciascun ciclo è necessario una propria sezione. Il nome della sezione corrisponde al nome del ciclo:
  ```
  [Cycle123]
  Mname = TestGC
  Dname = testgc.com
  OUTPUT = Code1
  Anzp = 3
  Version = 0
  Code_typ = 1
  Icon = cycle123.png
  Desc_Text = This is describing text
  ```

Mname	Nome della maschera
Dname	Nome del file in cui è definita la maschera
OUTPUT	Nome del relativo metodo Output
Anzp	Numero dei parametri della maschera da decompilare (tutte le variabili create con DEF, anche variabili ausiliarie)
Version	(opzionale) dato della versione per ciclo

Icon	(opzionale) icona per la visualizzazione del programma a catene sequenziali , formato *.png Dimensioni dell'immagine per la risoluzione corrispondente: 640 x 480 mm → 16 x 16 pixel 800 x 600 mm → 20 x 20 pixel 1024 x 768 mm → 26 x 26 pixel 1280 x 1024 mm → 26 x 26 pixel 1280 x 768 mm → 26 x 26 pixel Percorso: [cartella di sistema user]/ico/ico<risoluzione> **Note:** • per le risoluzioni da 1280 si impiega la cartella per 1024 x 768 mm (adatta soltanto per programmi a catene sequenziali). • La dimensione schermo dipende non solo dalla risoluzione, ma anche dalla dimensione carattere impostata nell'editor.
Desc_Text	(opzionale) testo di spiegazione per la visualizzazione nel programma a catene sequenziali , lunghezza della stringa max. 17 caratteri (adatto soltanto per programmi a catene sequenziali)
Param_Text	(opzionale) testo per la lista dei parametri T=%1 D=%2

Nota

Note su Icon, Desc_Text e Param_Text:

1. L'icona utente viene sempre visualizzata nell'editor del codice G. Nell'editor ShopMill-/ShopTurn viene sempre visualizzata l'icona del codice G. Questo serve a distinguere meglio tra i passi in ShopMill-/ShopTurn e i passi NON in ShopMill-/ShopTurn.
2. Il passo Run MyScreen dipende dall'impostazione dell'editor "Rappresentare i cicli come passo di lavorazione". Questo vale per il codice G e anche per l'editor JobShop.
   Ciò significa che con "Rappresentare i cicli come passo di lavorazione" = "SÌ" viene applicato il "Desc_Text" della progettazione Run MyScreens, mentre con "NO" viene rappresentato il ciclo.
3. Desc_Text e Param_Text
   – possono essere specificati in funzione della lingua mediante la notazione $xxxxx
   – mediante %1, %2, ... si può accedere ai parametri del richiamo del ciclo generato. In questo caso il segnaposto viene sostituito dal parametro che si trova nella posizione specificata dietro il '%' nella lista dei parametri generata.

Esempio

```
//M(TestGC/"Generazione codice:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
```

```
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                          ;Scrittura del codice NC dal metodo OUT-
                                              PUT nel file \MPF.DIR\MESSEN.MPF:
                                              Cycle123(123, -6)
                                              M30

END_PRESS
```

Condizioni secondarie

- La funzionalità "decompilazione senza commento operativo" non dispone di tutte le funzioni che caratterizzano la "decompilazione con commento operativo".
  Nella "decompilazione senza commento operativo" sono supportati i tipici richiami dei cicli, come MYCYCLE(PAR1, PAR2, PAR3, ...). Nella riga del richiamo della funzione non devono tuttavia trovarsi commenti operativi. I parametri opzionali che non vengono trasmessi al richiamo della funzione e che sono del tipo di stringa S, devono però essere specificati almeno tra virgolette vuote, ad es. "". Altrimenti "Run MyScreens" tenterà di riempire questi parametri con virgole per poi decompilare il "richiamo del ciclo riempito".
- I parametri del tipo Stringa non devono contenere virgole o punti e virgola nella stringa da trasmettere.
- Nella "decompilazione senza commento operativo" tutte le variabili contenute nel metodo OUTPUT devono sempre essere racchiuse tra parentesi affinché sia supportata la funzionalità "riempimento con virgole dei parametri dei cicli mancanti".

Esempio:

Ammesso:

```
OUTPUT
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")"
END_OUTPUT
```

Non consentito (la variabile MYCOMMENT è collocata dopo la parentesi di chiusura):

```
OUTPUT
```

```
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")" MYCOMMENT
END_OUTPUT
```

8.3.36 Search Forward, Search Backward (SF, SB)

Descrizione

Con la funzione **SF, SB (Search Forward, Search Backward)** è possibile, partendo dalla posizione corrente del cursore, effettuare la ricerca di una stringa nel programma NC corrente dell'editor e visualizzarne il valore.

Programmazione

Sintassi:	**SF**("String")	
Definizione:	**S**earch **F**orward: ricerca in avanti a partire dalla posizione corrente del cursore	
Sintassi:	**SB**("String")	
Definizione:	**S**earch **B**ackward: ricerca all'indietro a partire dalla posizione corrente del cursore	
Parametri:	String	testo da ricercare

Regole per la ricerca

- Prima e dopo l'unità della stringa da ricercare e del suo valore, nel programma NC attuale deve essere presente uno spazio.
- L'elemento non viene ricercato nei commenti e nell'ambito di una stringa.
- Il valore da emettere deve essere un'espressione numerica, espressioni del tipo "X1=4+5" non vengono riconosciute.
- Le costanti esadecimali di forma X1='HFFFF', le costanti binarie di forma X1='B10010' e le costanti esponenziali di forma X1='-.5EX-4' vengono riconosciute.
- Il valore di una stringa può essere emesso se tra la stringa ed il valore
  - non sono presenti:
  - Spazio (Blank)
  - Segni di uguaglianza

Esempio

Sono possibili i seguenti tipi di rappresentazione:

```
X100 Y200               ;Alla variabile abc viene associato il valore 200
Abc = SB("Y")
X100 Y 200              ;Alla variabile abc viene associato il valore 200
Abc = SB("Y")
```

```
X100 Y=200              ;Alla variabile abc viene associato il valore 200
Abc = SB("Y")
```

8.3.37 SL_HMI_INSTALL_DIR

Descrizione

La funzione SL_HMI_INSTALL_DIR restituisce il percorso della directory di installazione di SINUMERIK Operate.

Programmazione

Sintassi:	SL_HMI_INSTALL_DIR()	
Descrizione:	Determinazione della directory di installazione di SINUMERIK Operate	
Parametri:	- nessuno -	

8.3.38 SL_TEMP_DIR

Descrizione

La funzione SL_TEMP_DIR restituisce un percorso per i file temporanei SINUMERIK Operate.

Nota

Il contenuto di questa directory non è ritentivo e viene cancellato ogni volta che si avvia SINUMERIK Operate.

Programmazione

Sintassi:	SL_TEMP_DIR()	
Descrizione:	Determinazione della directory non ritentiva (volatile) della directory di SINUMERIK Operate	
Parametri:	- nessuno -	

8.3.39 Funzioni STRING

Panoramica

Le funzioni seguenti consentono l'elaborazione di stringhe:

- determinazione della lunghezza di stringhe
- ricerca di un carattere in una stringa
- estrazione di una parte di stringa da sinistra
- estrazione di una parte di stringa da destra
- estrazione di una parte di stringa al centro
- Sostituzione di parti di stringhe
- Confronto di stringhe
- Inserimento di una stringa in una stringa
- Rimozione di una stringa da una stringa
- Rimozione di spazi (da sinistra o da destra)
- Inserimento di valori o stringhe con codici di formattazione

Funzione LEN: Lunghezza di una stringa

Sintassi:	**LEN**(string | varname)	
Descrizione:	Determinazione del numero di caratteri di una stringa	
Parametri:	string	Qualsiasi espressione valida con stringhe. In caso di stringa vuota viene restituito ZERO.
	varname	Qualsiasi nome di variabile valido e dichiarato
	è consentito solo uno dei due parametri.	

Esempio

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HELLO"
  VAR02=LEN(VAR01)          ; Risultato = 5
END_LOAD
```

Funzione INSTR: Ricerca di caratteri in un concatenamento di stringhe:

Sintassi:	**INSTR**(Start, String1, String2 [,direzione])
Descrizione:	Ricerca di caratteri

Parametri:	Start	Posizione di partenza da cui viene effettuata la ricerca da string1 a string2. Se la ricerca deve cominciare all'inizio della string2, va indicato 0.
	String1	Carattere che deve essere ricercato.
	String2	Concatenamento di stringhe nella quale avviene la ricerca.
	Direzione (opzionale)	Direzione in cui deve essere effettuata la ricerca 0: da sinistra a destra (preimpostazione) 1: da destra a sinistra
	Se string1 non è contenuta in string2 viene restituito uno 0.	

Esempio

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HELLO/WORLD"
  VAR02=INST(1,"/",VAR01)        ; Risultato = 6
END_LOAD
```

Funzione LEFT: Stringa da sinistra

Sintassi:	**LEFT**(stringa, lunghezza)	
Descrizione:	LEFT restituisce un concatenamento di caratteri, che contiene il numero di caratteri indicato dal lato sinistro di una stringa.	
Parametri:	string	Concatenamento di caratteri o variabile con il concatenamento di caratteri da elaborare
	lunghezza	Numero di caratteri che devono essere letti

Esempio

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,5)           ; risultato = "HELLO"
END_LOAD
```

Funzione RIGHT: Stringa da destra

Sintassi:	**RIGHT**(stringa, lunghezza)	
Descrizione:	RIGHT restituisce un concatenamento di caratteri, che contiene il numero di caratteri indicato dal lato destro di una stringa.	
Parametri:	string	Concatenamento di caratteri o variabile con il concatenamento di caratteri da elaborare
	lunghezza	Numero di caratteri che devono essere letti

Esempio

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,4)         ; Risultato = "WORLD"
END_LOAD
```

Funzione MIDS: Stringa dal centro

Sintassi:	**MIDS**(stringa, start [, lunghezza])	
Descrizione:	MIDS restituisce un concatenamento di caratteri, che contiene il numero di caratteri indicato a partire dal punto indicato di una stringa.	
Parametri:	string	Concatenamento di caratteri o variabile con il concatenamento di caratteri da elaborare
	start	Punto dal quale avviene la lettura nella sequenza di caratteri
	lunghezza	Numero di caratteri che devono essere letti

Esempio

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HELLO/WORLD"
  VAR02=LEFT(VAR01,4,4)           ;Risultato = "LO/W"
END_LOAD
```

Funzione REPLACE: Sostituzione di caratteri

Sintassi:	**REPLACE**(string, FindString, ReplaceString [, start [, count]])
Descrizione:	La funzione REPLACE sostituisce un carattere/concatenamento di caratteri in una stringa con un altro carattere/concatenamento di caratteri.

Parametri:	string	Stringa in cui FindString deve essere sostituito attraverso ReplaceString.
	FindString	Stringa da sostituire
	ReplaceString	Stringa sostitutiva (viene inserita al posto di FindString)
	start	Posizione di partenza dalla quale deve essere effettuata la ricerca/sostituzione
	count	Numero di caratteri che devono essere ricercati dalla posizione di partenza a FindString

Valore di ritorno:		
	stringa = stringa vuota	Copia di stringa
	FindString = stringa vuota	Copia di stringa
	ReplaceString = stringa vuota	Copia di stringa in cui vengono cancellate tutte le occorrenze di FindString
	start > Len(String)	Stringa vuota
	count = 0	Copia di stringa

Funzione STRCMP: Confronto di stringhe

Sintassi:	**STRCMP**(string1, string2 [, CaseInsensitive])	
Descrizione:	STRCMP confronta una stringa di caratteri con un'altra stringa di caratteri.	
Parametri:	string1	Stringa di caratteri da confrontare con una seconda stringa di caratteri
	string2	Stringa di caratteri da confrontare con la prima stringa di caratteri
	CaseInsensitive	Confronto con/senza considerazione delle maiuscole/minuscole: senza indicazione o FALSE = si tiene conto delle maiuscole/minuscole TRUE = non si tiene conto delle maiuscole/minuscole

Esempio

```
REG[0]=STRCMP("Hugo", "HUGO")              ; Risultato = 32
                                           ; (<> 0, le stringhe non sono identiche)

REG[0]=STRCMP("Hugo", "HUGO", TRUE)        ; Risultato = 0
                                           ; (== 0, le stringhe sono identiche
```

Funzione STRINSERT: Inserimento stringa

Sintassi:	**STRINSERT**(string1, string2 , insert position)
Descrizione:	STRINSERT inserisce una stringa di caratteri, in una determinata posizione, in una stringa di caratteri.

Parametri:	string1	Stringa di caratteri da inserire in una stringa di caratteri
	string2	Sequenza di caratteri da inserire
	insert position	Posizione in cui va inserita in una stringa di caratteri

Esempio

```
REG[0]=STRINSERT("Hello!", " world", 5)    ; risultato = "Hello world!")
```

Funzione STRREMOVE: Rimozione stringa

Sintassi:	**STRREMOVE**(string1, remove position, count)	
Descrizione:	STRREMOVE rimuove un certo numero di caratteri in una determinata posizione all'interno di una stringa di caratteri.	
Parametri:	string1	Stringa di caratteri dalla quale vanno rimossi dei caratteri
	remove position	Posizione dalla quale vanno rimossi dei caratteri dalla stringa di caratteri
	count	Numero di caratteri da rimuovere

Esempio

```
REG[0]=STRREMOVE("Hello world!", 5, 6)     ; Risultato = "Hello!")
```

Funzione TRIMLEFT: rimozione di spazi a sinistra della stringa

Sintassi:	**TRIMLEFT**(string1)	
Descrizione:	TRIMLEFT rimuove spazi a sinistra di una stringa di caratteri.	
Parametri:	string1	Stringa di caratteri dalla quale vanno rimossi degli spazi a sinistra della stringa

Esempio

```
REG[0]=TRIMLEFT("    Hello!")              ; Risultato = "Hello!")
```

Funzione TRIMRIGHT: rimozione di spazi a destra della stringa

Sintassi:	**TRIMRIGHT**(string1)	
Descrizione:	TRIMRIGHT rimuove spazi a destra di una stringa di caratteri.	
Parametri:	string1	Stringa di caratteri dalla quale vanno rimossi degli spazi a destra della stringa

Esempio

```
REG[0]=TRIMRIGHT("Hello!    ")            ; Risultato = "Hello!")
```

Funzione FORMAT: inserimento di valori o stringa con codice di formattazione

Sintassi:	**FORMAT**(testo con codice di formattazione [, valore1] ... [, valore28])	
Descrizione:	La funzione FORMAT offre la possibilità di inserire, grazie all'utilizzo di codici di formattazione, fino a 28 valori o stringhe in determinati punti in un testo predefinito.	
Codici di formattazione:	Sintassi	%[flags] [larghezza] [.cifre decimali] tipo
	Flags	Carattere opzionale per la definizione della formattazione di emissione: • allineato a destra o a sinistra ("-" per allineato a sinistra) • aggiunta di zeri non significativi ("0") • default: riempimento con spazi se il valore da emettere ha meno cifre di quanto indicato con [larghezza]
	Larghezza	L'argomento stabilisce la larghezza minima di emissione di un numero non negativo. Se il valore 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 del tipo stabilisce quali formati di dati dell'istruzione Print vengono trasferirti. Questi caratteri devono essere specificati. • d: Valore intero • f: Numero a virgola mobile • s: String • o: ottale • x: esadecimale • b: binario

Esempio

```
DEF VAR1
DEF VAR2
LOAD
VAR1 = 123
VAR2 = FORMAT("Hello %08b %.2f %s!", VAR1 + 1, 987.654321, "world")
                                    ; risultato = "Hello 01111100 987.65 world!"
END_LOAD
```

Vedere anche

Utilizzo di stringhe (Pagina 104)

8.3.40 Loop WHILE/UNTIL

Descrizione

Con i comandi DO-LOOP è possibile realizzare un loop. Je nach Projektierung wird diese entweder solange durchlaufen wie eine Bedingung erfüllt ist (WHILE) oder bis eine Bedingung zutrifft (UNTIL).

Poiché, in funzione della progettazione, i loop possono pregiudicare le prestazioni del sistema, gli stessi vanno adottati con cautela e rinunciando ad azioni che richiedano molto tempo.

Come variabile di esecuzione, ad es., si raccomanda l'impiego di un registro (REG[]) poiché anche le variabili di visualizzazione normali (in particolare quelle con collegamento alle variabili di sistema o utente) possono pregiudicare le prestazioni del sistema per effetto degli aggiornamenti o delle operazioni di scrittura estremamente frequenti.

Con l'ausilio della funzione DEBUG (vedere il capitolo DEBUG (Pagina 146)) è possibile determinare il tempo di esecuzione dei metodi "Run MyScreens". In questo modo è possibile identificare i problemi generati dai loop (carico della CPU elevato, capacità di reazione ridotta).

Nota

Essendo possibile sostituire ogni loop FOR con un loop WHILE, in EasyScreen non è supportata la sintassi per la formulazione di un loop FOR.

Programmazione

```
DO
   <Istruzioni>
LOOP_WHILE <condizione per la continuazione del loop>

DO
   <Istruzioni>
LOOP_UNTIL <condizione per la conclusione del loop>

DO_WHILE <condizione per la continuazione del loop>
   <Istruzioni>
LOOP

DO_UNTIL <condizione per la conclusione del loop>
   <Istruzioni>
LOOP
```

Esempio

```
  REG[0] = 5
DO
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
     REG[1] = -5

   DO
      DEBUG("INNER: " << REG[1])
      REG[1] = REG[1] + 1
   LOOP_WHILE REG[1] < 0

LOOP_WHILE REG[0] < 10
```

```
  REG[0] = 5
DO
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
     REG[1] = -5

   DO
      DEBUG("INNER: " << REG[1])
      REG[1] = REG[1] + 1
   LOOP_UNTIL 0 <= REG[1]

LOOP_WHILE 10 > REG[0]
```

```
  REG[0] = 5
DO_WHILE 10 > REG[0]
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
     REG[1] = -5

   DO_UNTIL 0 <= REG[1]
     DEBUG("INNER: " << REG[1])
     REG[1] = REG[1] + 1
   LOOP

LOOP
```

```
  REG[0] = 5
DO_WHILE 10 > REG[0]
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
     REG[1] = -5

   DO
     DEBUG("INNER: " << REG[1])
     REG[1] = REG[1] + 1
   LOOP_UNTIL 0 <= REG[1]

LOOP
```

8.3.41 Esecuzione ciclica degli script: START_TIMER, STOP_TIMER

Descrizione

Con l'ausilio dei timer è possibile richiamare ciclicamente i metodi SUB. A questo scopo sono previste le funzioni START_TIMER() e STOP_TIMER().

Nota

È possibile progettare soltanto un timer per ciascun metodo SUB.

Programmazione

Sintassi:	**START_TIMER**("nome SUB", intervallo)	
Descrizione:	Avvio dell'elaborazione ciclica di un metodo SUB	
Parametri:	Nome SUB:	Nome del metodo SUB da richiamare ciclicamente
	Intervallo:	Intervallo in millisecondi

Sintassi:	**STOP_TIMER**("nome SUB")	
Descrizione:	Arresto dell'elaborazione ciclica di un metodo SUB	
Parametri:	Nome SUB:	Nome del metodo SUB il cui timer deve essere arrestato

Esempio

```
//M(TimerSample/"My timer")
DEF MyVariable=(I//0/,"Number of cyclic calls:"/WR1)
VS1=("Start%ntimer")
VS2=("Stop%ntimer")

SUB(MyTimerSub)
   MyVariable = MyVariable + 1
END_SUB

PRESS(VS1)
   ;Calls SUB "MyTimerSub" every 1000 milliseconds
   START_TIMER("MyTimerSub", 1000)
END_PRESS

PRESS(VS2)
   STOP_TIMER("MyTimerSub")
END_PRESS
```

Se START_TIMER viene nuovamente richiamato per un timer già assegnato ad un metodo SUB, si applica il nuovo intervallo se questo è diverso da quello precedente. In caso contrario, questo secondo richiamo viene ignorato.

A seconda del sistema, l'intervallo minimo è di 100 millisecondi.

Se STOP_TIMER viene richiamato per un metodo SUB per il quale attualmente non sia in esecuzione un timer, questo richiamo è ignorato.

Elementi grafici e logici

9

9.1 Linea, linea di separazione, rettangolo, cerchio ed ellisse

9.1.1 Definizione degli elementi grafici

Descrizione

Le linee, le linee di separazione, i rettangoli e le ellissi / i cerchi vengono progettati nel metodo LOAD:

- I rettangoli trasparenti si ottengono sovrapponendo il colore di riempimento al colore di sfondo del sistema.
- Con l'elemento ELLIPSE, un cerchio si costituisce quando l'altezza e la larghezza sono impostate in maniera equivalente.
- Le linee di separazione orizzontali e verticali presentano sempre la larghezza o l'altezza della finestra esatta. Le linee di separazione non danno luogo a barre di scorrimento.
  Se in precedenza per la rappresentazione delle linee di separazione si è utilizzato l'elemento RECT, ad es. RECT(305,0,1,370,0,0,1), lo stesso viene riconosciuto dal supporto alla programmazione e convertito automaticamente in V_SEPARATOR(305,1,"#87a5cd", 1). Ciò vale tanto per le linee di separazione verticali quanto per quelle orizzontali.

Elemento LINE

Programmazione:

Sintassi:	LINE (x1, y1, x2, y2, color, style, tag, visible)
Descrizione:	definizione della linea

Parametri:	x1	Coordinata x del punto iniziale
	y1	Coordinata y del punto iniziale
	x2	Coordinata x del punto finale
	y2	Coordinata y del punto finale
	color	Colore della linea
	style	Stile della linea: 1 = continua 2 = tratteggiata 3 = puntinata 4 = tratteggiata e puntinata
	tag	Tramite questo tag, con le funzioni SHOW_GRAPHIC, HIDE_GRAPHIC e DELETE_GRAPHIC, è possibile mostrare, nascondere ed eliminare un elemento grafico o un gruppo di elementi grafici.
	visible	Elemento grafico visibile (TRUE/FALSE)

Elemento RECT

Programmazione:

Sintassi:	RECT (x, y, w, h, frame color, fill color, style, tag, visible)	
Descrizione:	Definizione del rettangolo	
Parametri:	x	Coordinata x in alto a sinistra
	y	Coordinata y in alto a sinistra
	w	Larghezza
	h	Altezza
	frame color	Colore della cornice
	fill color	Colore di riempimento
	style	Stile della cornice: 1 = continua 2 = tratteggiata 3 = puntinata 4 = tratteggiata e puntinata
	tag	Tramite questo tag, con le funzioni SHOW_GRAPHIC, HIDE_GRAPHIC e DELETE_GRAPHIC, è possibile mostrare, nascondere ed eliminare un elemento grafico o un gruppo di elementi grafici.
	visible	Elemento grafico visibile (TRUE/FALSE)

Elemento ELLISSE

Programmazione:

Sintassi:	ELLIPSE(x, y, w, h, frame color, fill color, style, tag, visible)
Descrizione:	Definizione dell'ellisse o del cerchio

Parametri:	x	Coordinata x in alto a sinistra
	y	Coordinata y in alto a sinistra
	w	Larghezza
	h	Altezza
	frame color	Colore della cornice
	fill color	Colore di riempimento
	style	Stile della cornice: 1 = continua 2 = tratteggiata 3 = puntinata 4 = tratteggiata e puntinata
	tag	Tramite questo tag, con le funzioni SHOW_GRAPHIC, HIDE_GRAPHIC e DELETE_GRAPHIC, è possibile mostrare, nascondere ed eliminare un elemento grafico o un gruppo di elementi grafici.
	visible	Elemento grafico visibile (TRUE/FALSE)

Un valore equivalente di altezza e larghezza dà luogo a un cerchio.

Elemento V_SEPARATOR

Programmazione:

Sintassi:	V_SEPARATOR (x,w,color,pen)	
Descrizione:	Definizione della linea di separazione verticale	
Parametri:	x	Posizione x
	pen width	Spessore della linea
	color	Colore
	style	Stile della linea: 1 = continua 2 = tratteggiata 3 = puntinata 4 = tratteggiata e puntinata
	tag	Tramite questo tag, con le funzioni SHOW_GRAPHIC, HIDE_GRAPHIC e DELETE_GRAPHIC, è possibile mostrare, nascondere ed eliminare un elemento grafico o un gruppo di elementi grafici.
	visible	Elemento grafico visibile (TRUE/FALSE)

Elemento H_SEPARATOR

Programmazione:

Sintassi:	H_SEPARATOR (y,h,color,pen)
Descrizione:	Definizione della linea di separazione orizzontale

Parametri:	y	Posizione y
	pen width	Spessore della linea
	color	Colore
	style	Stile della linea: 1 = continua 2 = tratteggiata 3 = puntinata 4 = tratteggiata e puntinata
	tag	Tramite questo tag, con le funzioni SHOW_GRAPHIC, HIDE_GRAPHIC e DELETE_GRAPHIC, è possibile mostrare, nascondere ed eliminare un elemento grafico o un gruppo di elementi grafici.
	visible	Elemento grafico visibile (TRUE/FALSE)

Ulteriori informazioni

Nella sezione Funzioni grafiche (Pagina 194) sono descritte le funzioni SHOW_GRAPHIC, HIDE_GRAPHIC, CLEAR_BACKGROUND e DELETE_GRAPHIC. Con queste funzioni si può controllare la visualizzazione degli elementi grafici.

9.1.2 Funzioni grafiche

Panoramica

Sono disponibili le seguenti funzioni grafiche:

- CLEAR_BACKGROUND
- DELETE_GRAPHIC
- HIDE_GRAPHIC
- SHOW_GRAPHIC

Le funzioni si possono applicare agli elementi grafici LINE, RECT, ELLIPSE, V_SEPARATOR e H_SEPARATOR (Pagina 191).

Funzione CLEAR_BACKGROUND: eliminazione di elementi grafici

Con la funzione CLEAR_BACKGROUND è possibile cancellare gli elementi grafici.

Sintassi:	CLEAR_BACKGROUND()	
Descrizione:	Eliminazione di elementi grafici. La memoria occupata dagli oggetti grafici viene liberata.	
Parametri:	- nessuno -	

Funzione DELETE_GRAPHIC: eliminazione di un elemento grafico o di un gruppo di elementi grafici

Con la funzione DELETE_GRAPHIC si può eliminare un determinato elemento grafico o un gruppo di elementi grafici. Occorre definire un valore per i rispettivi elementi grafici nel parametro "tag".

- Rispetto alla funzione CLEAR_BACKGROUND si possono eliminare elementi grafici specifici.
- Rispetto alla funzione HIDE_GRAPHIC viene liberata la memoria occupata dagli oggetti grafici.

Sintassi:	**DELETE_GRAPHIC**(tag)	
Descrizione:	Nascondere un elemento grafico o un gruppo di elementi grafici il cui valore nel parametro "tag" è identico.	
Parametri:	tag	Il parametro "tag" può essere definito negli elementi grafici. È anche possibile definire più elementi grafici con lo stesso valore e quindi formare un gruppo di elementi grafici.

Esempio

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
DELETE_GRAPHIC(5)                                    ;i 3 elementi grafici
                                                     del tipo LINE con tag=5
                                                     vengono eliminati
```

Funzione HIDE_GRAPHIC: occultamento di un elemento grafico o di un gruppo di elementi grafici

Con la funzione HIDE_GRAPHIC si può nascondere un determinato elemento grafico o un gruppo di elementi grafici. Occorre definire un valore per i rispettivi elementi grafici nel parametro "tag".

Sintassi:	HIDE_GRAPHIC(tag)	
Descrizione:	Nasconde un elemento grafico o un gruppo di elementi grafici il cui valore nel parametro "tag" è identico.	
Parametri:	tag	Il parametro "tag" può essere definito negli elementi grafici. È anche possibile definire più elementi grafici con lo stesso valore e quindi formare un gruppo di elementi grafici.

Esempio

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
```

```
HIDE_GRAPHIC(5)                                          ;i 3 elementi grafici
                                                         del tipo LINE con tag=5
                                                         vengono nascosti
```

Funzione SHOW_GRAPHIC: visualizzazione di un elemento grafico o di un gruppo di elementi grafici

Con la funzione SHOW_GRAPHIC si può visualizzare un determinato elemento grafico o un gruppo di elementi grafici. Occorre definire un valore per i rispettivi elementi grafici nel parametro "tag".

Sintassi:	SHOW_GRAPHIC(tag)	
Descrizione:	Visualizza un elemento grafico o un gruppo di elementi grafici il cui valore nel parametro "tag" è identico.	
Parametri:	tag	Il parametro "tag" può essere definito negli elementi grafici. È anche possibile definire più elementi grafici con lo stesso valore e quindi formare un gruppo di elementi grafici.

Esempio

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
SHOW_GRAPHIC(5)                                          ;i 3 elementi grafici
                                                         del tipo LINE con tag=5
                                                         vengono visualizzati
```

9.2 Definizione di un array

Definizione

Con l'aiuto di un array, i dati di un tipo di dati omogeneo vengono ordinatamente archiviati nella memoria, in modo tale che diventi possibile l'accesso ai dati mediante un indice.

Descrizione

Gli array possono essere mono o bidimensionali. Un array mono dimensionale viene considerato come uno bidimensionale ad una sola riga o colonna..

Gli array vengono definiti con il codice //A e si concludono con //END. Il numero di righe e di colonne è libero. Un array ha la seguente struttura:

Programmazione

Sintassi:	**//A**(Identificatore) (a/b...) (c/d...) ... //END	
Descrizione:	Definizione di un array	
Parametri:	Identificatore	Nome dell'array
	a, b, c, d	Valori dell'array I valori di tipo STRING devono essere indicati tra doppie virgolette.

Esempio

```
//A(filetto)                    ; Dimensioni/passo/diametro del nocciolo
(0.3 / 0.075 / 0.202)
(0.4 / 0.1   / 0.270)
(0.5 / 0.125 / 0.338)
(0.6 / 0.15  / 0.406)
(0.8 / 0.2   / 0.540)
(1.0 / 0.25  / 0.676)
(1.2 / 0.25  / 0.676)
(1.4 / 0.3   / 1.010)
(1.7 / 0.35  / 1.246)
//END
```

9.2.1 Accesso al valore di un elemento array

Descrizione

Attraverso la proprietà "valore" (identificatore.val) è possibile inoltrare il valore di un accesso array.

L'indice righe (numero di riga dell'array) e l'indice colonne (numero di colonna dell'array) iniziano sempre da 0. Se un indice di riga o di colonna si trova al di fuori dell'array, viene emesso il valore 0 oppure una stringa vuota e la variabile ERR assume il valore TRUE. La variabile ERR risulta TRUE anche quando il criterio di ricerca non è stato trovato.

Programmazione

Sintassi:	Identificatore [Z,[M[,C]]].val oppure Identificatore [Z,[M[,C]]]		
Descrizione:	Accesso a un array mono dimensionale con una sola colonna		
Sintassi:	Identificatore [S,[M[,C]]].val] oppure Identificatore [S,[M[,C]]] oppure		
Descrizione:	Accesso a un array mono dimensionale con una sola riga		
Sintassi:	Identificatore [Z,S,[M[,C]]].val oppure Identificatore [Z,S,[M[,C]]]		
Descrizione:	Accesso a un array bidimensionale		
Parametri:	Identificatore:	Nome dell'array	
	Z:	Valore riga (indice righe o criterio di ricerca)	
	S:	Valore colonna (indice colonne o criterio di ricerca)	
	M:	Modalità di accesso	
		0	Diretta
		1	ricerca riga, colonna diretta
		2	riga diretta, ricerca colonna
		3	ricerca
		4	ricerca indice righe
		5	ricerca indice colonne
	C:	Modalità di confronto	
		0	Il criterio di ricerca deve trovarsi nel campo dei valori della riga o della colonna.
		1	Il criterio di ricerca deve essere trovato in maniera esatta.

Esempio:	`VAR1 = MET_G[REG[3],1,0].VAL`	`;assegnare a Var1 un valore da array MET_G`

Modalità di accesso

- **Modalità di accesso "Diretta"**
  Con la modalità di accesso "Diretta" (M = 0) avviene l'accesso all'array con l'indice di riga in Z e l'indice di colonna in S. La modalità di confronto C non viene elaborata.
- **Modalità di accesso "Ricerca"**
  Nella modalità di accesso M = 1, 2 o 3, la ricerca avviene sempre nella riga 0 o nella colonna 0.

Modo M	Valore riga Z	Valore colonna S	Valore di uscita
0	Indice righe	Indice colonne	Valore dalla riga Z e dalla colonna S
1	Criterio di ricerca: Ricerca nella colonna 0	Indice colonna della colonna da cui viene letto il valore	Valore dalla riga trovata e dalla colonna S
2	Indice della riga dalla quale viene letto il valore restituito	Criterio di ricerca: Ricerca nella riga 0	Valore dalla riga Z e dalla colonna trovata

Modo M	Valore riga Z	Valore colonna S	Valore di uscita
3	Criterio di ricerca: Ricerca nella colonna 0	Criterio di ricerca: Ricerca nella riga 0	Valore dalla riga e dalla colonna trovata
4	Criterio di ricerca: Ricerca nella colonna S	Indice della colonna in cui viene effettuata la ricerca	Indice righe
5	Indice della riga in cui viene effettuata la ricerca	Criterio di ricerca: Ricerca nella riga Z	Indice colonne

Modalità di confronto

Se si utilizza la modalità di confronto C = 0, il contenuto della riga di ricerca o della colonna di ricerca deve essere ordinato in sequenza crescente. Se il criterio di ricerca è minore del primo elemento o maggiore dell'ultimo, viene emesso il valore 0 oppure una stringa vuota e la variabile di errore ERR risulta TRUE.

Se si utilizza la modalità di confronto C = 1, il criterio di ricerca deve poter essere trovato nella riga di ricerca o nella colonna di ricerca. Se non è possibile trovare il criterio di ricerca, viene emesso il valore 0 oppure una stringa vuota e la variabile di errore ERR risulta TRUE.

9.2.2 Esempio: Accesso a un elemento array

Prerequisito

Di seguito vengono definiti due array come prerequisito per i seguenti esempi:

```
//A(filetto)
                (0.3 / 0.075 / 0.202)
                (0.4 / 0.1   / 0.270)
                (0.5 / 0.125 / 0.338)
                (0.6 / 0.15  / 0.406)
                (0.8 / 0.2   / 0.540)
                (1.0 / 0.25  / 0.676)
                (1.2 / 0.25  / 0.676)
                (1.4 / 0.3   / 1.010)
                (1.7 / 0.35  / 1.246)
//END


//A(Array2)
                ("DEF" /     "STG" /     "KDM" )
                (0.3 /       0.075 /     0.202 )
                (0.4 /       0.1 /       0.270 )
                (0.5 /       0.125 /     0.338 )
                (0.6 /       0.15 /      0.406 )
```

```
                    (0.8 /       0.2 /       0.540 )
                    (1.0 /       0.25 /      0.676 )
                    (1.2 /       0.25 /      0.676 )
                    (1.4 /       0.3 /       1.010 )
                    (1.7 /       0.35 /      1.246 )
//END
```

Esempi

- **Modalità di accesso - Esempio 1:**
  in Z si trova il criterio di ricerca. Questo criterio viene sempre ricercato nella colonna 0. Attraverso l'indice righe del criterio trovato, il valore viene restituito dalla colonna S:
  ```
  VAR1 = filetto[0.5,1,1]   ;VAR1 ha il valore 0.125
  ```
  Spiegazione:
  Ricerca il valore 0.5 nella colonna 0 dell'array "Filetto" e restituisce il valore trovato nella colonna 1 della stessa riga.
- **Modalità di accesso - Esempio 2:**
  in S si trova il criterio di ricerca. Questo criterio viene sempre ricercato nella riga 0. Attraverso l'indice colonne del criterio trovato, il valore viene restituito dalla riga Z:
  ```
  VAR1 = ARRAY2[3,"STG",2]   ;VAR1 ha il valore 0.125
  ```
  Spiegazione:
  Ricerca la colonna contenente "STG" nella riga 0 dell'array "Array2". Restituisce il valore della colonna trovata e della riga con l'indice 3.
- **Modalità di accesso - Esempio 3:**
  il criterio di ricerca si trova in Z ed in S. Con il criterio in Z viene ricercato l'indice di riga nella colonna 0 e con il criterio in S l'indice di colonna nella riga 0. Attraverso l'indice righe e l'indice colonne trovato, il valore viene restituito dall'array:
  ```
  VAR1 = ARRAY2[0.6,"STG",3]   ;VAR1 ha il valore 0.15
  ```
  Spiegazione:
  ricerca della riga contenente 0.6 nella colonna 0 dell'array "Array2", ricerca della colonna contenente "STG" nella riga 0 dell'array "Array2". Trasferisce il valore dalla riga e dalla colonna trovate a VAR1.
- **Modalità di accesso - Esempio 4:**
  in Z si trova il criterio di ricerca. In S si trova l'indice della colonna nella quale avviene la ricerca. L'indice righe del criterio trovato viene restituito:
  ```
  VAR1 = filetto[0.125,1,4]   ;VAR1 ha il valore 2
  ```
  Spiegazione:
  ricerca del valore 0.125 nella colonna 1 dell'array "Filetto" e trasferisce l'indice di riga del valore trovato a VAR1.
- **Modalità di accesso - Esempio 5:**
  in Z si trova l'indice della riga nella quale avviene la ricerca. Il criterio di ricerca si trova in S. L'indice colonne del criterio trovato viene restituito:
  ```
  VAR1 = filetto[4,0.2,5,1]   ;VAR1 ha il valore 1
  ```
  Spiegazione:
  ricerca del valore 0.2 nella riga 4 dell'array "Filetto" e trasferisce l'indice di colonna del valore trovato a VAR1. È stata scelta la modalità di confronto 1, poiché i valori della riga 4 non sono sono ordinati in modo crescente.

9.2.3 Richiamo dello stato di un elemento array

Descrizione

Con la proprietà "stato" è possibile verificare se un accesso all'array fornisce un valore corretto.

Programmazione

Sintassi:	Identificatore [Z, S, [M[,C]]]**.vld**
Descrizione:	Questa proprietà è di sola lettura.
Parametri:	Identificatore Nome dell'array
Valore di ritorno:	FALSE = un valore non valido
	TRUE = un valore valido

Esempio

```
DEF MPIT = (R///"MPIT",,"MPIT",""/wr3)
DEF PIT  = (R///"PIT",,"PIT",""/wr3)
PRESS(VS1)
  MPIT = 0.6
  IF MET_G[MPIT,0,4,1].VLD == TRUE
    PIT    = MET_G[MPIT,1,0].VAL
    REG[4] = PIT
    REG[1] = "OK"
  ELSE
    REG[1] = "ERROR"
  ENDIF
END_PRESS
```

9.3 Descrizione tabelle (grid)

Definizione

Al contrario dell'array, i valori di una tabella (grid) vengono aggiornati costantemente. Si tratta di una rappresentazione tabellare di valori che possono provenire da un NC o da un PLC. La tabella è organizzata per colonne e pertanto all'interno di una colonna si trovano sempre variabili dello stesso tipo.

Assegnazione

Il riferimento a una definizione della tabella viene effettuato nella definizione delle variabili:

- La definizione della variabile determina i valori da indicare, mentre la definizione degli elementi della tabella l'aspetto e la disposizione sullo schermo. Le proprietà dei campi di input/output della riga di definizione delle variabili vengono acquisite nella tabella.
- L'area visibile della tabella viene determinata dalla larghezza e altezza del campo di input/output. Se sono presenti più righe o colonne di quante possano essere visualizzate nell'area visibile, è possibile scorrere orizzontalmente e verticalmente.

Identificatore della tabella

Identificatore di una tabella di uguali valori dell'NCK o del PLC, i quali possono essere indirizzati attraverso un blocco di un canale. L'identificatore della tabella viene distinto dai valori limite o dal campo di toggle attraverso il segno %. L'identificatore della tabella può essere seguito, separato da una virgola, dal nome di un file che contiene la definizione della tabella.

Identificatore di una tabella di valori dello stesso tipo ad es. da NCK o PLC. L'identificatore viene specificato nella posizione in cui è progettato il dato per i valori limite o il campo di toggle. L'identificatore della tabella viene introdotta da un carattere % che lo precede, seguito da un nome file separato da una virgola. Questo indica il file in cui è definita la descrizione della tabella. La tabella viene cercata per default nel file di progettazione in cui è progettata la maschera.

Una definizione della tabella si può caricare anche dinamicamente, ad es. con il metodo LOAD, tramite la funzione Load Grid (LG).

Variabile di sistema o utente

Questo parametro rimane vuoto per le tabelle, in quanto le variabili da mostrare vengono indicate nel dettaglio nelle righe di definizione della colonna. La descrizione della tabella può essere disponibile in formato dinamico.

Esempio

Definizione di una variabile Variable MyGridVar che nel campo di I/O (distanza a sinistra: 100, distanza dall'alto: standard, larghezza: 350, altezza: 100) rappresenta una tabella "MyGrid1".

```
DEF MyGridVar=(V/% MyGrid1///////100,,350,100)
```

Vedere anche

Parametri delle variabili (Pagina 91)

Load Grid (LG) (Pagina 163)

9.3.1 Definizione di una tabella (grid)

Descrizione

Il blocco della tabella è costituito da:

- Descrizione dell'intestazione
- 1 ... n descrizioni delle colonne

Programmazione

Sintassi:	**//G**(Identificatore tabella/Tipo tabella/Numero righe/ [Attributo riga fissa],[Attributo colonna fissa])		
Descrizione:	Definizione di una tabella (grid)		
Parametri:	Identificatore della tabella	l'identificatore della tabella viene utilizzato in questo caso senza anteporre il carattere % Può essere impiegato una sola volta in una finestra di dialogo.	
	Tipo di tabella	0 (preimpostazione)	Tabella per dati PLC o utente (dati specifici NCK e per canale)
		1	e altri riservati
	Numero di righe	Numero di righe inclusa la riga di intestazione	
		La riga o la colonna fisse non vengono ruotate (scrolling). Il numero di colonne si ricava dal numero delle colonne progettate.	
	Visualizzazione della riga dei titoli di colonna	-1:	La riga dei titoli di colonna non viene visualizzata
		0:	La riga dei titoli di colonna viene visualizzata (default)
	Numero di colonne fisse	0:	Nessuna colonna fissa
		1 ... n:	Numero di colonne che devono essere sempre visibili, ossia che non devono scorrere in orizzontale

Esempi

`//G(grid1/0/5/-1,2)`	Viene visualizzata la riga dei titoli di colonna e 2 colonne fisse
`//G(grid1/0/5/,1)`	Viene visualizzata la riga dei titoli di colonna e 1 colonna fissa
`//G(grid1/0/5)`	Viene visualizzata la riga dei titoli di colonna e nessuna colonna fissa

9.3.2 Definizione delle colonne

Descrizione

Per le tabelle (grid) si rivela utile utilizzare variabili con indice. Per variabili PLC o NC il numero di indice con uno o più indici è significativo.

I valori visualizzati in una tabella possono essere modificati direttamente dall'utente finale nell'ambito dei diritti stabiliti con gli attributi e all'interno dei limiti eventualmente definiti.

Programmazione

Sintassi:	(Tipo/Valore limite/vuoto/testo completo,titolo colonna/attributi/pagina di help/ Variabile di sistema o utente/Larghezza colonna/offset1, offset2, offset3) Vedere anche Sintassi di progettazione estesa (Pagina 46).	
Descrizione:	Definizione delle colonne	
Parametri:	analogamente alle variabili	
	Tipo	Tipo di dati
	Valori limite	Valore limite MIN, valore limite MAX
	Testo esteso, titolo colonna	
	Attributi	
	Pagina di help	
	Variabile di sistema o utente	Come variabili vanno indicate variabili PLC o NC tra doppie virgolette.
	Larghezza colonna	Indicazione in pixel
	Offset	Le ampiezze incremento da applicare all'indice in questione per riempire la colonna vengono indicate nel parametro di offset associato: • Offset1: ampiezza incremento per il 1° indice • Offset2: ampiezza incremento per il 2° indice • Offset3: ampiezza incremento per il 3° indice

Titolo della colonna dal file di testo

Il titolo della colonna può essere impostato come testo oppure numero del testo (&8xxxx) e non può essere ruotato (scrolling).

Modifica delle proprietà delle colonne

Le proprietà delle colonne modificabili in modo dinamico (scrivibili) sono:

- Valori limite (min, max),
- Titolo della colonna (st)
- Attribute (wr, ac e li)
- Pagina di help (hlp)

La modifica di una proprietà della colonna avviene tramite l'identificatore delle variabili dalla riga di definizione e tramite l'indice della colonna (che inizia con 1).

Esempio: `VAR1[1].st="Colonna 1"`

Per la definizione delle colonne si possono utilizzare gli attributi wr, ac e li.

Vedere anche

Load Grid (LG) (Pagina 163)

9.3.3 Controllo della selezione nella tabella (grid)

Descrizione

Attraverso le proprietà Row e Col è possibile impostare e rilevare l'elemento selezionato all'interno di una tabella:

- Identificatore.**Row**
- Identificatore.**Col**

Programmazione

Ogni cella di una tabella possiede le proprietà Val e Vld.

Per la scrittura e lettura delle proprietà della cella, oltre all'identificatore delle variabili dalla riga di definizione occorre indicare un indice righe e colonne.

Sintassi:	Identificatore[Indice righe, indice colonne].Val oppure Identificatore[Indice righe, indice colonne]
Descrizione:	Proprietà Val
Sintassi:	Identificatore[Indice righe, indice colonne].Vld
Descrizione:	Proprietà Vld

Esempio

```
Var1[2,3].val=1.203
```

Se non viene indicato alcun indice righe e colonne, sono validi gli indici delle celle selezionate, ossia:

```
Var1.Row =2
Var1.Col=3
Var1.val=1.203
```

9.4 Custom Widget

9.4.1 Definizione dei Custom Widget

Descrizione

Con l'ausilio di un Custom Widget vengono progettati nella finestra di dialogo elementi di visualizzazione specifici per l'utente.

Opzione software

Per utilizzare i Custom Widget nelle finestre di dialogo, sono necessarie anche le seguenti opzioni software:

"SINUMERIK Integrate Run MyHMI /3GL" (6FC5800-0AP60-0YB0)

Programmazione

Definizione:	**DEF** (Nome)	
Sintassi:	**(W///"","**(Nome libreria)**.**(Nome della classe)**"/////**a**,**b**,**c**,**d**);**	
Descrizione:	W	Definizione di un Custom Widget
Parametri:	Nome	Nome del Custom Widget, liberamente selezionabile
	Nome libreria	Liberamente selezionabile, nome del file della libreria dll (Windows) oppure so (Linux)
	Nome della classe	Liberamente selezionabile, nome della funzione della classe dalla libreria precedentemente nominata
	a, b, c, d	Posizione e dimensione della progettazione

Esempio

Un Custom Widget viene definito nel seguente modo nella progettazione delle finestre di dialogo:

```
DEF Cus = (W///"","slestestcustomwidget.SlEsTestCustomWidget"/////
20,20,250,100);
```

9.4.2 Struttura della libreria Custom Widget

Descrizione

Essenzialmente la libreria Custom Widget contiene una classe definita. Il nome di questa classe deve essere indicata nella progettazione della finestra di dialogo accanto al nome della libreria. A partire dal nome della libreria, "Run MyScreens" accede ad un file dll omonimo, ad es.:

```
slestestcustomwidget.dll
```

Programmazione

La definizione della classe del file dll dovrebbe avere il seguente aspetto:

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT

class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
....
public slots:
   bool serialize(const QString& szFilePath, bool bIsStoring);
....
}
```

9.4.3 Struttura dell'interfaccia Custom Widget

Descrizione

Per poter visualizzare il Custom Widget nella finestra di dialogo, la libreria viene integrata con un'interfaccia. Questa contiene macrodefinizioni con cui "Run MyScreens" può avviare il Custom Widget. L'interfaccia è presente sotto forma di un file cpp. Il nome del file è liberamente selezionabile, ad es.:
`sleswidgetfactory.cpp`

Programmazione

L'interfaccia viene definita nel seguente modo:

```
#include "slestestcustomwidget.h"     ; Il file di header del Custom Widget corri-
                                      spondente viene inserito all'inizio del file
....
//Makros                              ; Le macrodefinizioni non vengono modificate
....
WIDGET_CLASS_EXPORT(SlEsTestCustom-   ; Il Custom Widget corrispondente viene di-
Widget)                               chiarato alla fine del file
```

Esempio

Contenuto del file sleswidgetfactory.cpp per un Custom Widget con il nome della classe "SlEsTestCustomWidget":

```
#include <Qt/qglobal.h>
```

```
#include "slestestcustomwidget.h"

////////////////////////////////////////////////////////////////////////
// MAKROS FOR PLUGIN DLL-EXPORT - DO NOT CHANGE
////////////////////////////////////////////////////////////////////////

#ifndef Q_EXTERN_C
#ifdef __cplusplus
#define Q_EXTERN_C    extern "C"
#else
#define Q_EXTERN_C    extern
#endif
#endif

#define SL_ES_FCT_NAME(PLUGIN) sl_es_create_ ##PLUGIN
#define SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( IMPLEMENTATION , PARAM) \
    { \
    IMPLEMENTATION *i = new PARAM; \
    return i; \
    }

#ifdef Q_WS_WIN
#   ifdef Q_CC_BOR
#       define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C __declspec(dllexport) void* \
    __stdcall SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#   else
#       define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C __declspec(dllexport) void* SL_ES_FCT_NAME(PLUGIN) \
    (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#   endif
#else
#   define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C void* SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#endif

#define WIDGET_CLASS_EXPORT(CLASSNAME)  \
    EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(CLASSNAME,CLASSNAME(pParent))

////////////////////////////////////////////////////////////////////////
// FOR OEM USER - please declare here your widget classes for export
```

```
////////////////////////////////////////////////////////////////////////

WIDGET_CLASS_EXPORT(SlEsTestCustomWidget)
```

9.4.4 Interazione fra il Custom Widget e la finestra di dialogo - scambio dati automatico

I Custom Widget interagiscono con le finestre di dialogo e possono visualizzare o manipolare i valori.

Condizioni

Uno scambio dati automatico avviene alle condizioni seguenti:

Condizione	Direzione
All'avvio o alla decompilazione di una finestra di dialogo	Finestra di dialogo → Custom Widget
All'esecuzione del comando GC per la generazione di richiami di cicli	Custom Widget → Finestra di dialogo

Programmazione

Le seguenti definizioni sono necessarie per le interazioni:

Ampliamento della progettazione della finestra di dialogo

Definizione:	**DEF** (Variabile)	
Sintassi:	**(**(tipo)**//5/"","**(variabile)**","/wr2/)**	
Tipo di variabile:	Tipo	Campo di immissione standard (nessuna griglia o toggle) con qualsiasi tipo di dati (no W)
Parametri:	Variabile	Denominazione a piacere di una variabile per lo scambio dati
Modo di introduzione:	wr2	lettura e scrittura

Esempio

```
DEF CUSVAR1 = (R//5/"","CUSVAR1",""/wr2/)
```

Ampliamento della definizione della classe

Nella definizione della classe del Custom Widget deve essere creato un QProperty il cui nome è identico alla variabile selezionata della progettazione della finestra di dialogo, ad es.:

```
Q_PROPERTY(double CUSVAR1 READ cusVar1 WRITE setCusVar1);
```

Esempio

La definizione della classe del file dll dovrebbe avere il seguente aspetto:

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT


class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
 Q_OBJECT
 Q_PROPERTY(double CUSVAR1  READ cusVar1    WRITE setCusVar1);
....
....
}
```

9.4.5 Interazione fra Custom Widget e finestra di dialogo - scambio dati manuale

Oltre allo scambio dati automatico è possibile anche uno scambio dati manuale. Lo scambio avviene in modo dinamico, ossia nel tempo di esecuzione della finestra di dialogo. Le azioni possibili sono le seguenti:

- Le proprietà del Custom Widget possono essere lette e scritte.
- I metodi del Custom Widget possono essere richiamati dalla progettazione Run MyScreens.
- A un determinato segnale del Custom Widget è possibile reagire e quindi richiamare sottoprogrammi (SUB) nella progettazione Run MyScreens.

9.4.5.1 Lettura e scrittura di proprietà

Descrizione

Per leggere e scrivere le proprietà del Custom Widget sono disponibili le funzioni ReadCWProperties e WriteCWProperties nella progettazione Run MyScreens.

Programmazione

Sintassi:	**ReadCWProperty**("nome variabile", "nome proprietà")	
Descrizione:	Lettura della proprietà di un Custom Widget	
Parametri:	Nome della variabile	Nome della variabile della finestra di dialogo a cui è assegnato un Custom Widget
	Nome proprietà	Nome della proprietà da leggere del Custom Widget
Valore di ritorno:	Valore attuale della proprietà del Custom Widget	

Sintassi:	**WriteCWProperty**("nome variabile", "nome proprietà", "valore")	
Descrizione:	Scrittura della proprietà di un Custom Widget	
Parametri:	Nome della variabile	Nome della variabile della finestra di dialogo a cui è assegnato un Custom Widget
	Nome proprietà	Nome della proprietà da scrivere del Custom Widget
	Valore	Valore che deve essere scritto nella proprietà del Custom Widget

Esempi

Esempio 1:

Lettura della proprietà "MyStringVar" del Custom Widget collegata alla variabile della finestra di dialogo "MyCWVar1" e assegnazione del valore nel registro 7.

Dichiarazione della classe del Custom Widget:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(QString MyStringVar
              READ myStringVar
              WRITE setMyStringVar);
   ...
```

Progettazione della finestra di dialogo:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   REG[7]=ReadCWProperty("MyCWVar1", "MyStringVar")
END_PRESS
```

Esempio 2:

Scrittura del risultato del calcolo "3 + sin(123.456)" nella proprietà "MyRealVar" del Custom Widget collegato alla variabile della finestra di dialogo "MyCWVar1".

Dichiarazione della classe del Custom Widget:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(double MyRealVar
              READ myRealVar
              WRITE setMyRealVar);
```

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
   ...
```

Progettazione della finestra di dialogo:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   WriteCWProperty("MyCWVar1", "MyRealVar", 3 + sin(123.456))
END_PRESS
```

9.4.5.2 Esecuzione di un metodo del Custom Widget

Descrizione

Per eseguire i metodi del Custom Widget, è disponibile la funzione CallCWMethod nella progettazione Run MyScreens.

Il metodo del Custom Widget da richiamare deve avere al massimo 10 parametri di trasferimento.

Sono supportati i seguenti formati dati dei parametri di trasferimento:

- bool
- uint
- int.
- double
- QString
- QByteArray

Programmazione

Sintassi:	**CallCWMethod**("nome variabile", "nome metodo[, argomento 0][, argomento 1 ... [,argomento 9]")
Descrizione:	Esecuzione di un metodo CustomWidget

Parametri:	Nome della variabile	Nome della variabile della finestra di dialogo a cui è assegnato un Custom Widget
	Nome metodo	Nome del metodo Custom Widget da richiamare
	Argomento 0 - 9	Parametri di trasferimento per il metodo Custom Widget Formati dati supportati: vedere sopra **Nota:** I parametri di trasferimento vengono sempre trasmessi "ByVal", il che significa che viene sempre trasferito solo il valore e non ad es. il riferimento a una variabile.
Valore di ritorno:	Valore di ritorno del metodo Custom Widget Sono supportati i seguenti formati dati dei parametri di trasferimento: • void • bool • uint • int. • double • QString • QByteArray **Nota**: Anche se il formato dati del valore di ritorno del metodo Custom Widget è "void", questo deve essere assegnato formalmente ad es. a una variabile.	

Esempio

Dichiarazione della classe del Custom Widget:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   public slots:
       void myFunc1(int nValue, const QString& szString, double dValue);
...
```

Progettazione della finestra di dialogo:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
DEF MyStringVar1 = (S)
DEF MyRealVar = (R)

PRESS(VS3)
   REG[9] = CallCWMethod("MyCWVar1", "myFunc1", 1+7, MyStringVar1, sin(MyRealVar) -
8)
END_PRESS
```

Nota

Il Custom Widget deve implementare il metodo "serialize". Qui si ha la possibilità di scrivere i dati interni di un Custom Widget
in un file predefinito o di ripristinarli a partire da questo file. Ciò si rende soprattutto necessario quando, con la maschera "Run MyScreens" aperta, si passa ad un altro settore operativo e poi si ritorna a quello precedente. Altrimenti i dati interni andrebbero perduti al momento di visualizzarli nuovamente.

Sintassi:	public slots: bool **serialize**(const QString& szFilePath, bool bIsStoring);	
Descrizione:	Lettura o scrittura dei dati e degli stati interni da o in un file	
Parametri:	szFilePath	Nome del file con indicazione completa del percorso in/da cui vanno letti/scritti i dati e gli stati interni del Custom Widget. Il file stesso deve eventualmente creare il Custom Widget.
	bIsStoring	TRUE = scrittura FALSE = lettura

Esempio

```
bool SlEsTestCustomWidget::serialize(const QString& szFilePath, bool bIsStoring)
{
   QFileInfo fi(szFilePath);
   bool bReturn = false;
   QDir dir;
   if (dir.mkpath(fi.canonicalPath()))
   {
      QFile fileData(szFilePath);
      QIODevice::OpenMode mode;

      if (bIsStoring)
      {
         mode = QIODevice::WriteOnly;
      }
      else
      {
         mode = QIODevice::ReadOnly;
      }

      if (fileData.open(mode))
      {
         QDataStream streamData;
         streamData.setDevice(&fileData);

         if (bIsStoring)
         {
            streamData << m_nDataCount << m_dValueX;
```

```
bool SlEsTestCustomWidget::serialize(const QString& szFilePath, bool bIsStoring)
          }
          else
          {
             streamData >> m_nDataCount >> m_dValueX;
          }

          streamData.setDevice(0);
             fileData.flush();
             fileData.close();
          bReturn = true;
       }
    }
    return bReturn;
}
```

9.4.5.3 Reazione a un segnale Custom Widget

Descrizione

In Run MyScreens è possibile reagire a un determinato segnale (invokeSub()) del Custom Widget e quindi richiamare un sottoprogramma (SUB).

Per il trasferimento dei valori (Custom Widget-Signal -> SUB) esistono 10 variabili globali, le cosiddette SIGARG, che nella progettazione sono confrontabili ai registri (REG). Qui vengono memorizzati i valori trasferiti con il segnale Custom Widget.

Sono supportati i seguenti formati dati dei parametri di trasferimento:

- bool
- uint
- int.
- double
- QString
- QByteArray

Programmazione

Richiamo del sottoprogramma:

Sintassi:	`void invokeSub(const QString& rszSignalName, const QVariantList& rvntList);`
Descrizione:	Segnale Custom Widget con il quale viene richiamato un sottoprogramma Run MyScreens.

Parametri:	rszSignalName	Nome del sottoprogramma Run MyScreens da richiamare
	rvntList	Array QVariantList per il trasferimento di parametri che vengono memorizzati nel parametro globale SIGARG e sono a disposizione nella progettazione. Dimensioni massime: 10 elementi Formati dati supportati: vedere sopra **Nota:** I parametri di trasferimento vengono sempre trasmessi "ByVal", il che significa che viene sempre trasferito solo il valore e non ad es. il riferimento a una variabile.

Sottoprogramma da richiamare:

Sintassi:	`SUB(on_<nome variabile>_<nome segnale>)`
	`…`
	`END_SUB`	

Descrizione:	Reazione a un segnale Custom Widget	
Parametri:	Nome variabile	Nome della variabile della finestra di dialogo a cui è assegnato un Custom Widget.
	Nome del segnale	Nome del segnale Custom Widget
	SIGARG 0 - 9	Parametri di trasferimento per il metodo Custom Widget. Formati dati supportati: Vedere sopra **Nota**: I parametri di trasferimento vengono sempre trasmessi "ByVal", il che significa che viene sempre trasferito solo il valore e non ad es. il riferimento a una variabile.

Esempio

Dichiarazione della classe del Custom Widget:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
   Q_OBJECT
signals:
   void invokeSub(const QString& szSubName, const QVariantList& vntList);
…
}
```

Classe CustomWidget:

```

QVariantList vntList;
vntList << 123.456;
emit invokeSub("MySub", vntList);
```

Progettazione della finestra di dialogo:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
SUB(on_MyCWVar1_MySub)
```

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
   DEBUG("SUB(on_MyCWVar1_MySub) was called with parameter: """ << SIGARG[0] <<
""""")
END_SUB
```

Risultato "easyscreen_log.txt":

```
[10:22:40.445] DEBUG: SUB(on_MyCWVar1_MySub) was called with parameter: "123.456"
```

9.5 SlEsGraphCustomWidget

9.5.1 SlEsGraphCustomWidget

Allgemein

Mit Hilfe von SlEsGraphCustomWidget können Sie geometrische Objekte (Punkt, Linie, Rechteck, Rechteck mit abgerundeten Ecken, Ellipse, Kreisbogen, Text) und Kurven aus Stützpunkten (z. B. Messwerte, Verlauf) darstellen.

Die Objekte werden in einer oder mehreren sogenannten Konturen organisiert. Diese können dann einzeln oder auch kombiniert zur Anzeige gebracht oder geleert, angewählt und gelöscht werden.

Die Objekte werden mit Hilfe von Funktionen der aktuell angewählten Kontur hinzugefügt und auch in dieser Reihenfolge gezeichnet. Möchten Sie Objekte mit einer bestimmten Farbe zeichnen, so müssen Sie vor dem Hinzufügen die entsprechende Zeichenfarbe setzen. Alle nachfolgend hinzugefügten Objekte werden dann mit dieser Zeichenfarbe gezeichnet. Neben der Zeichenfarbe können Sie auch die Zeichenbreite und den Zeichenstil beeinflussen. Bei den geschlossenen Objekten Rechteck, Rechteck mit abgerundeten Ecken und Ellipse können Sie vor dem Hinzufügen der Objekte eine Füllfarbe setzen.

Jede Kontur kann in verschiedene Modi gebracht werden:

- Normale Anzeige von allen Objekttypen.
- Punkte können zu einer sogenannten Polyline verbunden und so als eine Kurve/Graph dargestellt werden.
- Die Fläche zwischen Punkten, Linien oder Kreisbögen bis zur X-Achse kann mit der jeweils aktuellen Füllfarbe eingefärbt werden. Diese Möglichkeit können Sie zum Beispiel zur Visualisierung von Restmaterial beim Drehen nutzen.
  Werden in diesem Modus Punkte zusätzlich als Polyline angezeigt, wird die gesamte Fläche unterhalb der Kurve bis zur X-Achse ausgefüllt. Die Punkte, Linien und Kreisbögen können sich in allen vier Quadranten befinden.

Mit einem speziellen Cursor-Modi können Sie eine oder mehrere Konturen "ablaufen". Der Cursor wird dafür auf das erste oder letzte Punkt-Objekt einer Kontur gesetzt oder ausgehend von der aktuellen Cursorposition innerhalb der Kontur ein Punkt-Objekt vorwärts oder zurück bewegt.

Nach Bedarf kann eine zweite Y-Achse (rechts) mit eigener Skalierung eingeblendet werden. Diese ist durch einen Offset und einem Faktor an die erste Y-Achse (links) gekoppelt.

Mittels Suchfunktion kann anhand einer vorgegebenen X-Koordinate ein vorher einer Kontur hinzugefügter Punkt gefunden und falls gewünscht der Cursor darauf gesetzt werden.

Konturen können Sie auch als Ringpuffer mit einstellbarer Größe konfigurieren.

Durch die Serialisierung kann der aktuelle Zustand des SlEsGraphCustomWidget in binärer Form in einer Datei abgelegt und auch wiederhergestellt werden.

Das SlEsGraphCustomWidget kann über die Geste "Pan" (verschieben der View) und die Gesten "Pinch"/"Spread" (zoomen in/aus der View) bedient werden.

Per Mausbedienung kann die View verschoben (linke Maustaste + Move) und gezoomt (Mausrad) werden.

Die Anzeige wird aus Performance-Gründen nicht automatisch aktualisiert. Je nach Anwendungsfall können Sie durch Aufruf der entsprechenden Funktion die Aktualisierung anstoßen.

Beispiel

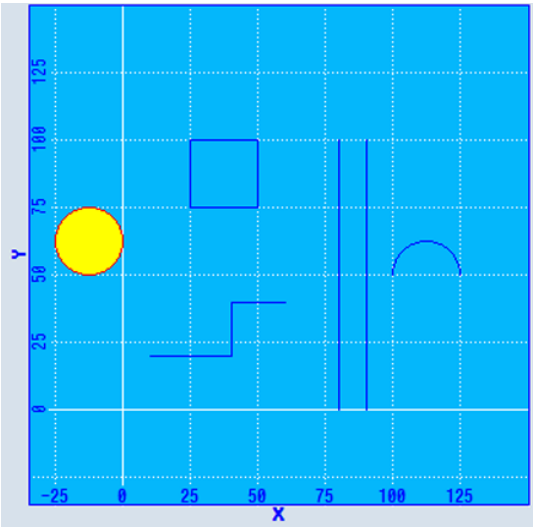


Figura 9-1 SlEsGraphCustomWidget - Beispiel

```
//M(MyGraphSampleMask/"SlEsGraphCustomWidget Sample")
DEF MyGraphVar = (W///,"slesgraphcustomwidget.SlEsGraphCustomWidget"/////10,10,340,340/0,0,0,0)
VS1=("Add objects",,se1)
LOAD
   WRITECWPROPERTY("MyGraphVar", "AxisNameX", "X")
   WRITECWPROPERTY("MyGraphVar", "AxisNameY", "Y")
   WRITECWPROPERTY("MyGraphVar", "ScaleTextOrientationYAxis", 2)
   WRITECWPROPERTY("MyGraphVar", "KeepAspectRatio", TRUE)

   REG[0]= CALLCWMETHOD("MyGraphVar", "addContour", "MyContour", TRUE)
   REG[0]= CALLCWMETHOD("MyGraphVar", "showContour", "MyContour")
```

```
//M(MyGraphSampleMask/"SlEsGraphCustomWidget Sample")
   REG[0]= CALLCWMETHOD("MyGraphVar", "setView", -35, -35, 150, 150)
END_LOAD

PRESS(VS1)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 10, 20, 40, 20)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 40, 20, 40, 40)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 40, 40, 60, 40)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 80, 0, 80, 100)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 90, 0, 90, 100)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addRect", 25, 100, 50, 75)
   REG[0]= CALLCWMETHOD("MyGraphVar", "setPenColor", "#ff0000");red
   REG[0]= CALLCWMETHOD("MyGraphVar", "setFillColor", "#ffff00");yellow
   REG[0]= CALLCWMETHOD("MyGraphVar", "addCircle", -12.5, 62.5, 12.5)
   REG[0]= CALLCWMETHOD("MyGraphVar", "setFillColor");off/default
   REG[0]= CALLCWMETHOD("MyGraphVar", "setPenColor");off/default
   REG[0]= CALLCWMETHOD("MyGraphVar", "addArc", 100, 37.5, 125, 62.5, 0, 180)
   REG[0]= CALLCWMETHOD("MyGraphVar", "update")
END_PRESS
```

9.5.2 Hinweise zur Performance

Abhängig von der eingesetzten Hardware und Grundauslastung des Systems müssen Sie bei Verwendung des SlEsGraphCustomWidgets die folgenden Richtwerte beachten. Neben der eingesetzten Hardware und Grundauslastung variieren die Werte abhängig von der Projektierung der SIEsGraphCustomWidgets.

Beachten Sie diese Richtwerte um die Reaktionsfähigkeit und Stabilität des Gesamtsystems nicht zu beeinträchtigen.

- Anzahl der zu einem Zeitpunkt gleichzeitig projektierten SlEsGraphCustomWidgets (z. B. maximal 1)
- Anzahl der projektierten Konturen (z. B. maximal 6)
- Anzahl der projektierten Grafikobjekte je Kontur (z. B. maximal 1.000)
- Häufigkeit der angeforderten Aktualisierungen (z. B. maximal 500 ms)

Nota

Die Anzeige ist nicht echtzeitfähig.

9.5.3 Properties lesen und schreiben

Definizione

Le properties descritte nel capitolo seguente vengono lette con la funzione ReadCWProperty() e impostate con WriteCWProperty().

Esempi

Lettura della property "CursorX" del SlEsGraphCustomWidget collegato tramite la variabile di visualizzazione "MyGraphVar". Il risultato viene scritto nel registro 0.

```
REG[0] = ReadCWProperty("MyGraphVar", "CursorX")
```

Scrittura del valore "MyFirstContour" nella property "SelectedContour" del SlEsGraphCustomWidget collegato tramite la variabile di visualizzazione "MyGraphVar".

```
WriteCWProperty("MyGraphVar ", "SelectedContour", "MyFirstContour")
```

9.5.4 Properties

Panoramica

Property	Descrizione
AxisNameX	Denominazione degli assi - asse X
AxisNameY	Denominazione degli assi - asse Y
AxisNameY2	Denominazione degli assi - secondo asse Y (destro)
AxisY2Visible	Visualizza/nasconde secondo asse Y (destro)
AxisY2Offset	Offset secondo asse Y (destro)
AxisY2Factor	Fattore secondo asse Y (destro)
ScaleTextEmbedded	Posizionamento dei testi di scala
ScaleTextOrientationYAxis	Orientamento dei testi di scala dell'asse Y
KeepAspectRatio	Conservare il rapporto tra altezza e larghezza
SelectedContour	Nome del profilo selezionato corrente
BackColor	Colore di sfondo
ChartBackColor	Colore di sfondo dell'area di disegno
ForeColor	Colore di primo piano dei testi e colore predefinito del tratto.
ForeColorY2	Colore di primo piano per i testi del secondo asse Y (destro)
GridColor	Colore delle linee della griglia
GridColorY2	Colore delle linee orizzontali della griglia del secondo asse Y (destro)
CursorColor	Colore del puntatore a croce
ShowCursor	Visualizza/nasconde il cursore
CursorX	Posizione X del cursore
CursorY	Posizione Y del cursore

Property	Descrizione
CursorY2	Posizione Y cursore riferito al secondo asse Y (destro)
CursorStyle	Tipo di rappresentazione del cursore
ViewMoveZoomMode	Comportamento in seguito ad azioni di zoom e spostamento mediante gesti

AxisNameX – Denominazione degli assi, asse X

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisNameX**") WriteCWProperty(GraphVarName, "**AxisNameX**", Value)	
Descrizione:	legge o imposta l'identificatore dell'asse X. Se non viene indicato alcun identificatore, viene utilizzato il posto disponibile per l'area di disegno.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (QString)
	Value	Valore da impostare (QString)

AxisNameX – Denominazione degli assi, asse Y

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisNameY**") WriteCWProperty(GraphVarName, "**AxisNameY**", Value)	
Descrizione:	Legge o imposta l'identificatore dell'asse Y. Se non viene indicato alcun identificatore, viene utilizzato il posto disponibile per l'area di disegno.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (QString)
	Value	Valore da impostare (QString)

AxisY2Visible – Visualizza/nasconde il secondo asse Y (destro)

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Visible**") WriteCWProperty(GraphVarName, "**AxisY2Visible**", Value)	
Descrizione:	Visualizza/nasconde il secondo asse Y (destro)	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE o FALSE

AxisY2Offset – Offset secondo asse Y (destro)

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Offset**") WriteCWProperty(GraphVarName, "**AxisY2Offset**", Value)
Descrizione:	Offset del secondo asse Y (destro) riferito al primo asse Y (sinistro). Vedere l'esempio relativo alla Property AxisY2Factor.

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (double)
	Value	Valore da impostare (double)

AxisY2Factor – Fattore del secondo asse Y (destro)

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Factor**") WriteCWProperty(GraphVarName, "**AxisY2Factor**", Value)	
Descrizione:	Fattore del secondo asse Y (destro) riferito al primo asse Y (sinistro).	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (double)
	Value	Valore da impostare (double)

Insieme alla Property "AxisY2Offset" è possibile visualizzare un secondo asse Y (destro) con una scala propria. La scala è accoppiata tramite un offset e un fattore al primo asse Y (a sinistra).

Formula per la conversione da Y2 a Y:

Y = Y2 / fattore - offset

Formula per la conversione da Y a Y2:

Y2 = (Y + offset) * fattore

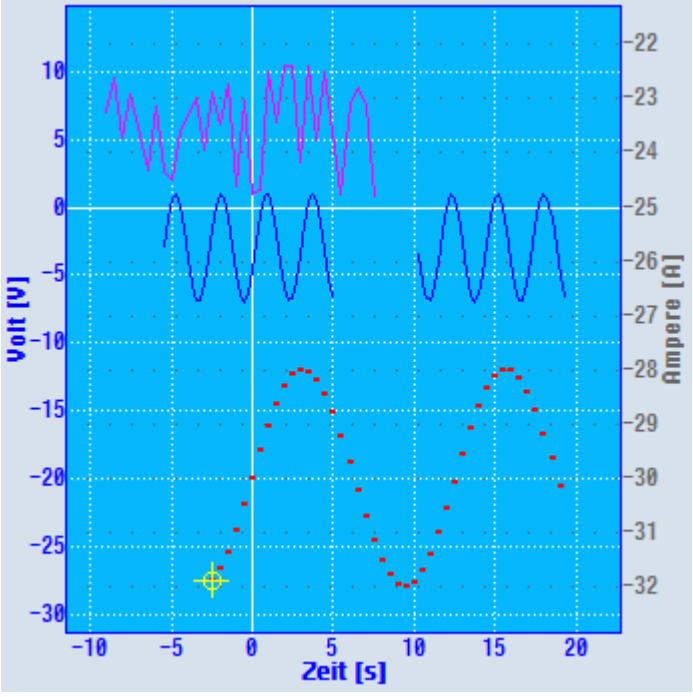


Figura 9-2 Esempio con secondo asse Y con scalatura propria

Esempio

Y: 0 ... 200 deve corrispondere a Y2: -25 ... 25.

- Offset: -100
- Fattore: 0,25

Esempio di calcolo:

Y2 = -30

Y = -30 / 0,25 – (-100) = -20

Esempio di calcolo:

Y = 0

Y2 = (0 + (-100)) * 0,25 = -25

L'asse X è uguale per entrambi i sistemi di coordinate.

I colori possono essere impostati con setForeColorY2() e setGridColorY2() (vedere Colori).

La denominazione degli assi è impostata con setAxisNameY2() (vedere Denominazione degli assi).

ScaleTextEmbedded - Posizionamento dei testi di scala

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**ScaleTextEmbedded**") WriteCWProperty(GraphVarName, "**ScaleTextEmbedded**", Value)	
Descrizione:	Con questa Property si definisce sei i testi di scala vengono posizionati all'interno o all'esterno dell'area di disegno.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE o FALSE

Esempio

Testi di scala all'esterno dell'area di disegno:

ScaleTextEmbedded = FALSE

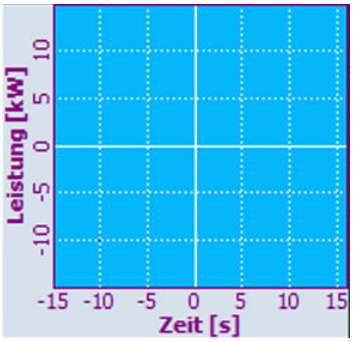


Figura 9-3 Testi di scala all'esterno dell'area di disegno

Testi di scala all'interno dell'area di disegno:

ScaleTextEmbedded = TRUE

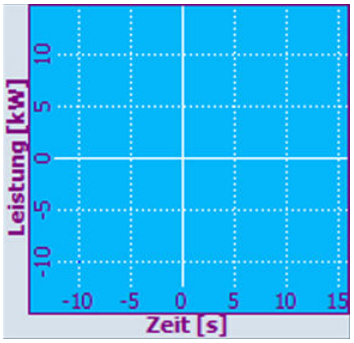


Figura 9-4 Testi di scala all'interno dell'area di disegno

ScaleTextOrientationYAxis - Orientamento dei testi di scala dell'asse Y

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**ScaleTextOrientationYAxis**") WriteCWProperty(GraphVarName, "**ScaleTextOrientationYAxis**", Value)	
Descrizione:	Con questa funzione i testi di scala dell'asse Y sono disposti con orientamento verticale.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (int)
	Value	Valore da impostare (int): 1 (= orizzontale) or 2 (= verticale)

Esempio

Testi di scala all'interno dell'area di disegno:

- ScaleTextEmbedded = TRUE

Orientamento del testo orizzontale:

- ScaleTextOrientationYAxis = 1

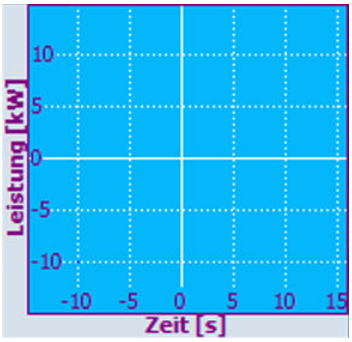


Figura 9-5 Testi di scala all'interno dell'area di disegno, orientamento del testo orizzontale

Orientamento del testo verticale:

- ScaleTextOrientationYAxis = 2

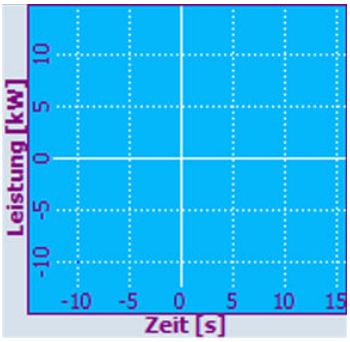


Figura 9-6 Testi di scala all'interno dell'area di disegno, orientamento del testo verticale

Testi di scala all'esterno dell'area di disegno:

- ScaleTextEmbedded = FALSE

Orientamento del testo orizzontale:

- ScaleTextOrientationYAxis = 1

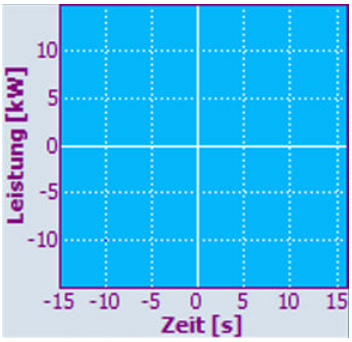


Figura 9-7 Testi di scala all'esterno dell'area di disegno, orientamento del testo orizzontale

Orientamento del testo verticale:

- ScaleTextOrientationYAxis = 2

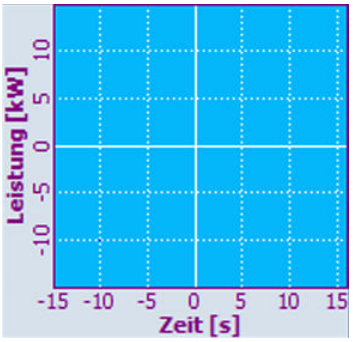


Figura 9-8 Testi di scala all'esterno dell'area di disegno, orientamento del testo verticale

KeepAspectRatio – Conservare il rapporto tra altezza e larghezza

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**KeepAspectRatio**") WriteCWProperty(GraphVarName, "**KeepAspectRatio**", Value)	
Descrizione:	Con questa Property si definisce se la vista definita con setView() deve essere orientata, adattata o ampliata automaticamente in modo che il rapporto tra altezza e larghezza della tra asse X e asse Y resti invariato. Questa proprietà è rilevante in particolare per la visualizzazione di figure geometriche. In questo modo un cerchio o un quadrato, ad es., vengono visualizzati come tali e non deformati in un'ellisse o un rettangolo.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SIEsGraphCustomWidget
	Return Value	Valore letto delle Properties (bool)
	Return Value	Valore da impostare (bool): TRUE o FALSE

Esempio

```
setView(-15, -15, 15, 15)
setFillColor("#A0A0A4")
addCircle(0, 0, 5)
KeepAspectRatio = TRUE
```

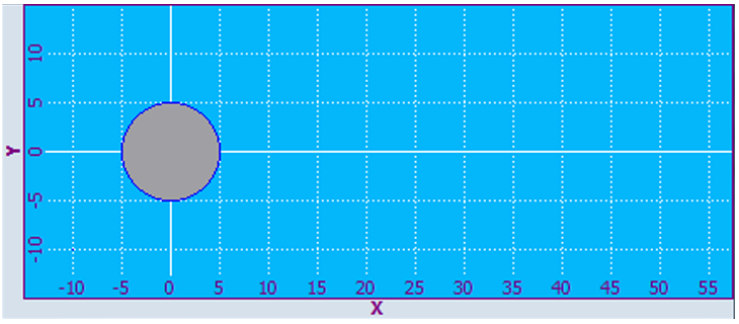


Figura 9-9 Il rapporto tra altezza e larghezza dell'asse X rispetto all'asse Y resta invariato

`KeepAspectRatio =` **FALSE**

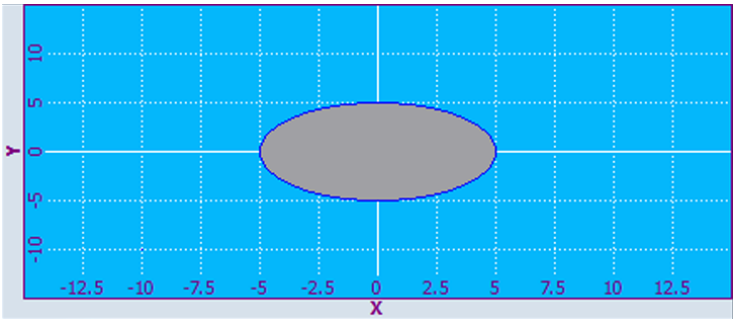


Figura 9-10 Rapporto tra altezza e larghezza dell'asse X rispetto all'asse Y deformato

SelectedContour – Nome del profilo selezionato corrente

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, " **SelectedContour**") WriteCWProperty(GraphVarName, " **SelectedContour**", Value)	
Descrizione:	Leggo o imposta la selezione del profilo corrente.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (QString)
	Value	Valore da impostare (QString)

Nota

Per poter eseguire operazioni su un profilo, ad es. per inserire oggetti grafici, occorre dapprima selezionarlo.

BackColor – Colore di sfondo

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, " **BackColor**") WriteCWProperty(GraphVarName, " **BackColor**", Value)	
Descrizione:	Colore di sfondo della dicitura dell'asse e, in funzione della Property ScaleTextInsideChartArea, anche dei testi di scala.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Colore letto delle Properties (QString)
	Value	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

ChartBackColor – Colore di sfondo dell'area di disegno

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**ChartBackColor**") WriteCWProperty(GraphVarName, "**ChartBackColor**", Value)	
Descrizione:	Colore di sfondo dell'area di disegno.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Colore letto delle Properties (QString)
	Value	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

ForeColor - Colore di primo piano della dicitura e colore predefinito del tratto

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**ForeColor**") WriteCWProperty(GraphVarName, "**ForeColor**", Value)	
Descrizione:	Colore di primo piano dei testi e colore predefinito del tratto.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Colore letto delle Properties (QString)
	Value	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

ForeColorY2 - Colore di primo piano per la dicitura del secondo asse Y (destro)

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**ForeColorY2**") WriteCWProperty(GraphVarName, "**ForeColorY2**", Value)	
Descrizione:	Colore di primo piano dei testi del secondo asse Y (destro).	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Colore letto delle Properties (QString)
	Value	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

GridColor - Colore delle linee della griglia

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**GridColor**") WriteCWProperty(GraphVarName, "**GridColor**", Value)	
Descrizione:	Colore delle linee della griglia.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Colore letto delle Properties (QString)
	Value	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

GridColorY2 - Colore delle linee orizzontali della griglia del secondo asse Y (destro)

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**GridColorY2**") WriteCWProperty(GraphVarName, "**GridColorY2**", Value)	
Descrizione:	Colore delle linee orizzontali della griglia del secondo asse Y (destro).	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Colore letto delle Properties (QString)
	Value	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

CursorColor - Colore del cursore

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorColor**") WriteCWProperty(GraphVarName, "**CursorColor**", Value)	
Descrizione:	Colore del cursore.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Colore letto delle Properties (QString)
	Value	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

ShowCursor – Visualizza/nascondi cursore

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, " **ShowCursor**") WriteCWProperty(GraphVarName, " **ShowCursor**", Value)	
Descrizione:	Visualizza/nasconde il cursore.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE o FALSE

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

CursorX – Posizione X del cursore

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorX**") WriteCWProperty(GraphVarName, "**CursorX**", Value)	
Descrizione:	Posizione X del cursore.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (double)
	Value	Valore da impostare (double)

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

CursorY – Posizione Y del cursore

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorY**") WriteCWProperty(GraphVarName, "**CursorY**", Value)	
Descrizione:	Posizione Y del cursore.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (double)
	Value	Valore da impostare (double)

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

CursorY2 – Posizione Y del cursore riferita al secondo asse Y (destro)

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorY2**") WriteCWProperty(GraphVarName, "**CursorY2**", Value)	
Descrizione:	Posizione Y del cursore riferita al secondo asse Y (destro).	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (double)
	Value	Valore da impostare (double)

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

CursorStyle – Tipo di rappresentazione del cursore

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorStyle**") WriteCWProperty(GraphVarName, "**CursorStyle**", Value)
Descrizione:	Tipo di rappresentazione del cursore.

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (int)
	Value	Valore da impostare (int): 0 (= croce) 1 (= puntatore a croce) 2 (= linea verticale) 3 (= linea orizzontale) 4 (= linea orizzontale e verticale)

ViewMoveZoomMode – Comportamento in seguito ad azioni di zoom e spostamento mediante gesti

Sintassi:	ReturnValue = ReadCWProperty(GraphVarName, "**ViewMoveZoomMode**") WriteCWProperty(GraphVarName, "**ViewMoveZoomMode**", Value)	
Descrizione:	Con i gesti è possibile modificare la vista corrente dei SlEsGraphCustomWidgets. Con questa Property si definisce in quale modo deve essere ammessa questa operazione. Ad esempio, in determinati casi applicativi può essere utile poter spostare e ingrandire la vista solo in orizzontale, ad es. per la visualizzazione dell'andamento dei valori di misura.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Valore letto delle Properties (int)
	Value	Valore da impostare (int): 0 (= off) 1 (= solo orizzontale) 2 (= solo verticale) 3 (= orizzontale e verticale)

9.5.5 Funktionen

Le funzioni indicate di seguito possono essere richiamate con la funzione CallCWMethod().

Esempio

Impostazione della vista del SlEsGraphCustomWidget collegato tramite la variabile di visualizzazione "MyGraphVar".

Il risultato viene scritto nel registro 0.

```
REG[0] = CallCWMethod("MyGraphVar", "setView", -8, -5, 11- 20)
```

Panoramica

Funzione	Descrizione
setView	Impostazione del sistema di coordinate
setMaxContourObjects	Impostazione del numero massimo di oggetti di un profilo (buffer ad anello)
addContour	Aggiunta di un profilo
showContour	Visualizzazione del profilo
hideContour	Impostazione per nascondere il profilo
hideAllContours	Impostazione per nascondere tutti i profili
removeContour	Eliminazione di un profilo
clearContour	Cancellazione degli oggetti grafici di un profilo
fitViewToContours	Adattamento automatico della vista
fitViewToContour	Adattamento automatico della vista
findX	Ricerca in un profilo
setPolylineMode	Rappresentazione dei punti di un profilo come poli-linea/curva
setIntegralFillMode	Rappresentazione dei punti di un profilo come poli-linea/curva
repaint, update	Aggiorna la visualizzazione
addPoint	Aggiunta di un punto in un profilo
addLine	Aggiunta di una linea in un profilo
addRect	Aggiunta di un rettangolo in un profilo
addRoundedRect	Aggiunta di un rettangolo con angoli arrotondati in un profilo.
addEllipse	Aggiunta di un'ellisse in un profilo
addCircle	Aggiunta di un cerchio in un profilo
addArc	Aggiunta di un arco di cerchio in un profilo
addText	Aggiunta di un testo in un profilo
setPenWidth	Impostazione della larghezza del tratto
setPenStyle	Impostazione dello stile del tratto
setPenColor	Impostazione del colore del tratto
setFillColor	Impostazione del colore di riempimento
setCursorPosition	Posizionamento del cursore
setCursorPositionY2	Posizionamento del cursore con riferimento al secondo asse Y (destro)
setCursorOnContour	Posizionamento del cursore sul profilo
moveCursorOnContourBegin	Posizionamento del cursore sul primo oggetto grafico di un profilo
moveCursorOnContourEnd	Posizionamento del cursore sull'ultimo oggetto grafico di un profilo
moveCursorOnContourNext	Posizionamento del cursore sull'oggetto grafico successivo di un profilo
serialize	Salvataggio/ripristino dello stato attuale

setView – Impostazione del sistema di coordinate

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setView**", x1, y1, x2, y2)	
Descrizione:	Con questa funzione si definiscono le dimensioni del sistema di coordinate che deve essere visualizzato all'interno del SlEsGraphCustomWidget. A questo proposito vedere anche la Property KeepAspectRatio.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x1	Margine sinistro (double)
	y1	Margine superiore (double)
	x2	Margine destro (double)
	y2	Margine inferiore (double)

Nota

La funzione aggiorna automaticamente la visualizzazione.

Esempio

```
setView(-8, -5, 11, 20)
AxisNameX = "X"
AxisNameY = "Y"
ScaleTextOrientationYAxis = 1
ScaleTextEmbedded = false
KeepAspectRatio = false
update
```

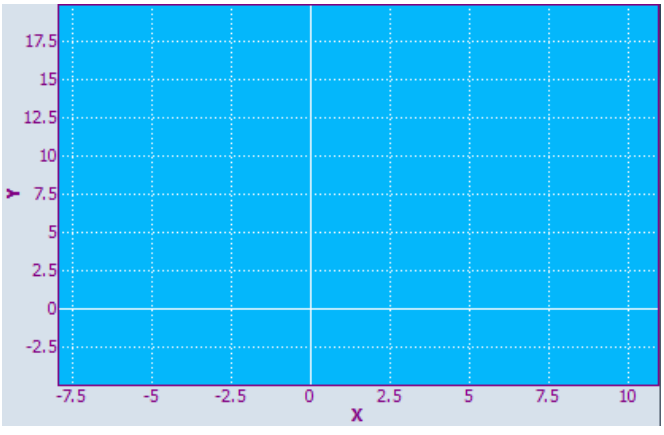


Figura 9-11 Esempio - setView

setMaxContourObjects – Impostazione del numero massimo di oggetti di un profilo (buffer ad anello)

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setMaxContourObjects**", Value) ReturnValue = CallCWMethod(GraphVarName, "**setMaxContourObjects**", Value, ContourName)	
Descrizione:	Con questa funzione si definiscono le dimensioni del buffer ad anello di un profilo. Se il numero degli oggetti aggiunti nel profilo supera questo limite, viene eliminato l'oggetto meno recente.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	Value	Numero massimo di oggetti grafici (uint)
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

Nota

La funzione aggiorna automaticamente la visualizzazione.

addContour – Aggiunta di un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName) ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName, Selected) ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName, Selected, AssignedY2)	
Descrizione:	Con questa funzione si crea un nuovo profilo. Opzionalmente si può assegnare il profilo al sistema di coordinate del secondo asse Y (destro).	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.
	Selected	Selezionare il profilo (bool): TRUE o FALSE
	AssignedY2	Assegnare il profilo al secondo asse Y (destro) (bool): TRUE o FALSE

showContour – Visualizzazione del profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**showContour**") ReturnValue = CallCWMethod(GraphVarName, "**showContour**", ContourName)
Descrizione:	Questa funzione permette di rendere visibile un profilo.

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

hideContour – Impostazione per nascondere un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**hideContour**") ReturnValue = CallCWMethod(GraphVarName, "**hideContour**", ContourName)	
Descrizione:	Questa funzione permette di rendere invisibile un profilo.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

showAllContours – Impostazione per rendere visibili tutti i profili

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**showAllContours**")	
Descrizione:	Questa funzione permette di rendere visibili tutti i profili.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito

hideAllContours – Impostazione per rendere invisibili tutti i profili

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**hideAllContours**")	
Descrizione:	Questa funzione permette di rendere invisibili tutti i profili.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito

removeContour – Impostazione per rendere invisibile il profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**removeContour**") ReturnValue = CallCWMethod(GraphVarName, "**removeContour**", ContourName)
Descrizione:	Elimina un profilo.

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

clearContour – Cancellazione degli oggetti grafici di un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**clearContour**") ReturnValue = CallCWMethod(GraphVarName, "**clearContour**", ContourName)	
Descrizione:	Cancella tutti gli oggetti grafici contenuti in un profilo.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

fitViewToContours – Adattamento automatico della vista

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContours**") ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContours**", OnlyVisible)	
Descrizione:	Con questa funzione, la vista viene adattata automaticamente ai profili. In funzione del parametro di riferimento si definisce se devono essere visibili solo i profili visibili o tutti i profili.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	OnlyVisible	Valore da impostare (bool): TRUE o FALSE

fitViewToContour – Adattamento automatico della vista

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContour**", ContourName)	
Descrizione:	Con questa funzione, la vista viene adattata automaticamente in modo che sia visibile solo un determinato profilo.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString)

findX – Ricerca in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**findX**", x) ReturnValue = CallCWMethod(GraphVarName, "**findX**", x, ContourName) ReturnValue = CallCWMethod(GraphVarName, "**findX**", x, ContourName, SetCursor)	
Descrizione:	Con queste funzioni è possibile ricercare in un profilo un punto definito in precedenza con una coordinata X specifica. Come risultato si ottiene la coordinata Y. Opzionalmente si può anche posizionare subito il cursore su questo punto.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (QString): se l'operazione è corretta, viene restituito il valore Y corrispondente
	x	Valore X da ricercare (double)
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato
	SetCursor	Impostazione del cursore (bool): TRUE o FALSE

Nota

Se si imposta il cursore con questa funzione, la visualizzazione viene aggiornata automaticamente.

setPolylineMode – Rappresentazione dei punti di un profilo come poli-linea/curva

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setPolylineMode**", SetPoly)	
Descrizione:	Tutti i punti aggiunti nel profilo corrente mediante richiamo di questa funzione vengono collegati a formare una poli-linea/curva. Questa funzione è specifica del profilo e può essere attivata o disattivata sezione per sezione.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	SetPoly	PolylineMode (bool): TRUE = on

Esempio

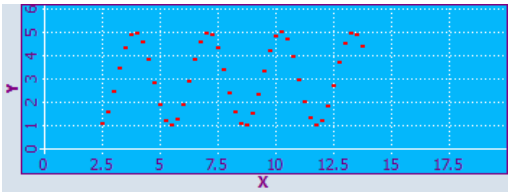


Figura 9-12 Esempio - PolylineMode: Off

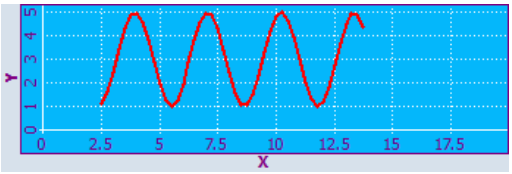


Figura 9-13 Esempio - PolylineMode: on

setIntegralFillMode – Riempimento delle aree tra punti, linee e archi di cerchio e l'asse X

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setIntegralFillMode**", SetIntFill)	
Descrizione:	Le aree comprese tra i punti e gli archi di cerchio e l'asse X vengono riempite con il colore di riempimento corrente (indipendentemente dal fatto che i punti siano +Y o –Y). Questa funzione è specifica del profilo e può essere attivata o disattivata sezione per sezione.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SIEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	SetIntFill	IntegralFillMode (bool): TRUE = on

Esempio

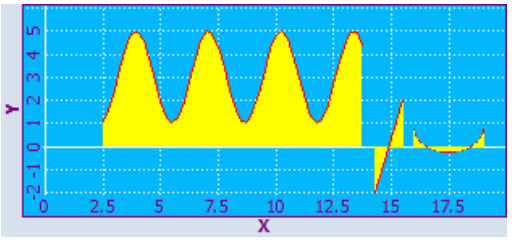


Figura 9-14 Esempio - setIntegralFillMode: on

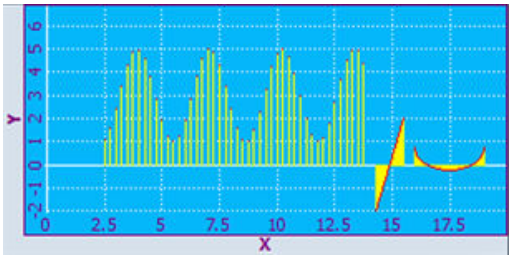


Figura 9-15 Esempio - setIntegralFillMode: Off

repaint, update – Aggiornamento della vista

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**repaint**") ReturnValue = CallCWMethod(GraphVarName, "**update**")	
Descrizione:	Con queste funzioni si può aggiornare manualmente la visualizzazione.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito

- update()
  La funzione aggiorna la vista, a condizione che l'aggiornamento del Widget non sia disattivato e che il Widget non sia nascosto. La funzione è ottimizzata in modo che le prestazioni dell'applicazione restino mantenute e che i ripetuti aggiornamenti non provochino sfarfallii.
- repaint()
  La funzione aggiorna la vista immediatamente dopo essere stata richiamata. Pertanto, la funzione repaint va utilizzata soltanto se è indispensabile un aggiornamento immediato, ad esempio per le animazioni.

Si consiglia di lavorare sempre con la funzione update(). Internamente, il SlEsGraphCustomWidget funziona sempre con la funzione update(), ad es. per setView().

addPoint – Aggiunta di un punto in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addPoint**", x, y) ReturnValue = CallCWMethod(GraphVarName, "**addPoint**", x, y, DummyPoint)	
Descrizione:	Aggiunta di un punto nel profilo correntemente selezionato. Inoltre è possibile impostare un punto fittizio, ossia che non viene visualizzato, ma che può essere posizionato a tale scopo con il cursore.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x	Coordinata x (double)
	y	Coordinata y (double)
	DummyPoint	Il punto viene considerato come invisibile (bool): TRUE = sì

addLine – Aggiunta di una linea in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addLine**", x1, y1, x2, y2)
Descrizione:	Aggiunta di una linea nel profilo correntemente selezionato

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x1	Coordinata x del punto iniziale (double)
	y1	Coordinata y del punto iniziale (double)
	x2	Coordinata x del punto finale (double)
	y2	Coordinata y del punto finale (double)

addRect – Aggiunta di un rettangolo in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addRect**", x1, y1, x2, y2)	
Descrizione:	Aggiunta di un rettangolo nel profilo correntemente selezionato.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x1	Margine sinistro (double)
	y1	Margine sinistro (double)
	x2	Margine destro (double)
	y2	Margine inferiore (double)

addRoundedRect – Aggiunta di un rettangolo con angoli arrotondati in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addRoundedRect**", x1, y1, x2, y2, r)	
Descrizione:	Aggiunta di un rettangolo con angoli arrotondati nel profilo correntemente selezionato.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x1	Margine sinistro (double)
	y1	Margine superiore (double)
	x2	Margine superiore (double)
	y2	Margine superiore (double)
	r	Raggio (double)

addEllipse – Aggiunta di un'ellisse in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addEllipse**", x1, y1, x2, y2, radius)
Descrizione:	Aggiunta di un'ellisse nel profilo correntemente selezionato.

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x1	Margine sinistro (double)
	y1	Margine superiore (double)
	x2	Margine destro (double)
	y2	Margine inferiore (double)
	radius	Raggio (double)

addCircle – Aggiunta di un cerchio in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addCircle**", x, y, radius)	
Descrizione:	Aggiunta di un cerchio nel profilo correntemente selezionato.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x	Margine sinistro (double)
	y	Margine superiore (double)
	radius	Raggio (double)

addArc – Aggiunta di un arco di cerchio in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addArc**", x1, y1, x2, y2, StartAngle, SpanAngle	
Descrizione:	Aggiunta di un arco di cerchio nel profilo correntemente selezionato.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x1	Margine sinistro (double)
	y1	Margine superiore (double)
	x2	Margine destro (double)
	y2	Margine inferiore (double)
	StartAngle	Angolo iniziale in gradi (double)
	SpanAngle	Angolo di apertura in gradi (double)

addText – Aggiunta di un testo in un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**addText**", x, y, Text)
Descrizione:	Aggiunta di un testo nel profilo correntemente selezionato.

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x	Margine sinistro (double)
	y	Margine superiore (double)
	Testo	Testo (QString)

setPenWidth – Impostazione della larghezza del tratto

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setPenWidth**") ReturnValue = CallCWMethod(GraphVarName, "**setPenWidth**", width)	
Descrizione:	Definisce la larghezza del tratto degli oggetti inseriti nel profilo corrente a partire dall'istante in cui viene richiamata la funzione.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	width	Larghezza del tratto (double). Se non viene definita una larghezza del tratto, viene impostata la larghezza del tratto predefinita.

setPenStyle – Impostazione dello stile del tratto

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setPenStyle**") ReturnValue = CallCWMethod(GraphVarName, "**setPenStyle**", style)	
Descrizione:	Definisce lo stile del tratto degli oggetti inseriti nel profilo corrente a partire dall'istante in cui viene richiamata la funzione.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	style	1 : linea continua (___) 2 : linea tratteggiata (_ _) 3 : linea a puntini 4 : linea-punto-linea (_._) 5 : linea-punto-punto (_..)

setPenColor – Impostazione del colore del tratto

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setPenColor**") ReturnValue = CallCWMethod(GraphVarName, "**setPenColor**", Color)
Descrizione:	Definisce il colore del tratto degli oggetti inseriti nel profilo corrente a partire dall'istante in cui viene richiamata la funzione.

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	Color	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

setFillColor – Impostazione del colore di riempimento

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setFillColor**") ReturnValue = CallCWMethod(GraphVarName, "**setFillColor**", Color)	
Descrizione:	Definisce il colore di riempimento degli oggetti inseriti nel profilo corrente a partire dall'istante in cui viene richiamata la funzione.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	Color	Valore da impostare (QString) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

setCursorPosition – Posizionamento del cursore

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorPosition**", x, y)	
Descrizione:	Imposta il cursore nella posizione preimpostata.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x	Coordinata x (double)
	y	Coordinata y (double)

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

setCursorPositionY2Cursor – Posizionamento del cursore con riferimento al secondo asse Y (destro)

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorPositionY2**", x, y2)	
Descrizione:	Imposta il cursore nella posizione preimpostata con riferimento al secondo asse Y (destro).	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	x	Coordinata x (double)
	y2	Coordinata Y riferita al secondo asse Y (destro) (double)

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

setCursorOnContour – Posizionamento del cursore sul profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorOnContour**") ReturnValue = CallCWMethod(GraphVarName, "**setCursorOnContour**", Contour-Name)	
Descrizione:	Posiziona il cursore sull'ultima posizione del cursore nell'ambito di un profilo.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

Esempio

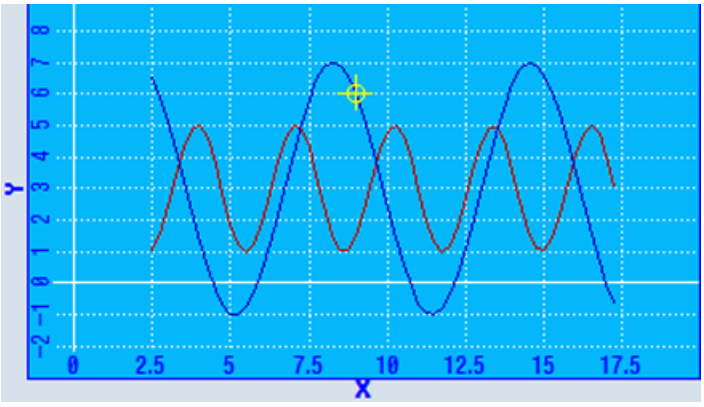


Figura 9-16 Esempio - setCursorOnContour

moveCursorOnContourBegin – Posizionamento del cursore sul primo oggetto grafico di un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBegin**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBegin**", ContourName)
Descrizione:	A partire dalla posizione corrente del cursore nell'ambito di un profilo, con questa funzione è possibile spostare il cursore al primo punto-oggetto di un profilo.

Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

moveCursorOnContourEnd – Posizionamento del cursore sull'ultimo oggetto grafico di un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourEnd**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourEnd**", ContourName)	
Descrizione:	A partire dalla posizione corrente del cursore nell'ambito di un profilo, con questa funzione è possibile spostare il cursore all'ultimo punto-oggetto di un profilo.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

moveCursorOnContourNext – Posizionamento del cursore sull'oggetto grafico successivo di un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourNext**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourNext**", ContourName)	
Descrizione:	A partire dalla posizione corrente del cursore nell'ambito di un profilo, con questa funzione è possibile spostare il cursore al successivo punto-oggetto di un profilo.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

moveCursorOnContourBack – Posizionamento del cursore sull'oggetto grafico precedente di un profilo

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBack**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBack**", ContourName)	
Descrizione:	A partire dalla posizione corrente del cursore nell'ambito di un profilo, con questa funzione è possibile spostare il cursore al precedente punto-oggetto di un profilo.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	ContourName	Nome del profilo (QString). Se non viene indicato alcun profilo, il richiamo si riferisce automaticamente al profilo correntemente selezionato.

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

serialize – Salvataggio/ripristino dello stato corrente

Sintassi:	ReturnValue = CallCWMethod(GraphVarName, "**serialize**", FilePath, IsStoring)	
Descrizione:	Con questa funzione è possibile scrivere in formato binario e ripristinare lo stato e il contenuto corrente del SlEsGraphWidget in un file o un DataStream.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = riuscito
	FilePath	Percorso completo con nome del file (QString)
	IsStoring	Salvataggio/ripristino (bool): TRUE = salvataggio

Nota

Questa funzione aggiorna automaticamente la visualizzazione.

9.5.6 Signale

Il segnale ViewChanged descritto qui di seguito può essere rilevato nella progettazione per attivare la reazione corrispondente.

Panoramica

Funzione	Descrizione
ViewChanged	Modifica della vista

ViewChanged – modifica della vista

Sintassi:	SUB(on_<GraphVarName>**_ViewChanged**) ... END_SUB	
Descrizione:	Se la vista cambia, viene emesso il segnale "ViewChanged". A questa situazione si può reagire nella progettazione con un metodo SUB specifico. I parametri SIGARG vengono impostati di conseguenza.	
Parametri:	GraphVarName	Nome della variabile di visualizzazione che contiene un SlEsGraphCustomWidget
	SIGARG[0]	Margine sinistro (double)
	SIGARG[1]	Margine superiore (double)
	SIGARG[2]	Margine destro (double)
	SIGARG[3]	Margine inferiore (double)

Esempio

```
DEF MyGraphVar = (W///,"slesgraphcustomwidget.SlEsGraphCustomWidget"/////
10,10,340,340/0,0,0,0)
```

```
SUB(on_MyGraphVar_ViewChanged
     DLGL("Current view: " << SIGARG[0] << ", " << SIGARG[1] << ", " <<   SIGARG[2]
<< ", " << SIGARG[3]
END_SUB
```

9.6 SIEsTouchButton

9.6.1 SIEsTouchButton

Informazioni generali

Con Run MyScreens è possibile creare in modo semplice applicazioni funzionalmente complesse. In particolare per i comandi Touch si possono progettare pulsanti liberamente posizionabili (TouchButtons). Per la progettazione dei TouchButtons valgono le seguenti proprietà:

- Le due possibili variazioni di stato del pulsante "clicked" e "checked" possono essere catturate nella progettazione per eseguire le azioni corrispondenti.
- I TouchButton possono essere controllati tramite Single-Touch, Multi-Touch, mouse o tastiera.
- Il TouchButton viene rappresentato in base alla risoluzione attuale. Questo vale anche per il font e per le immagini visualizzate.
- Il TouchButton dispone di due stili di visualizzazione, "Softkey Look&Feel" e "Specifico dell'utente". Nello stile di visualizzazione "Softkey Look&Feel", la rappresentazione del TouchButton corrisponde a quella dei softkey Operate. Per entrambi gli stili di visualizzazione sono disponibili funzioni come, ad esempio, la scalatura di immagini.
- I TouchButton vengono realizzati come CustomWidget e messi a disposizione.

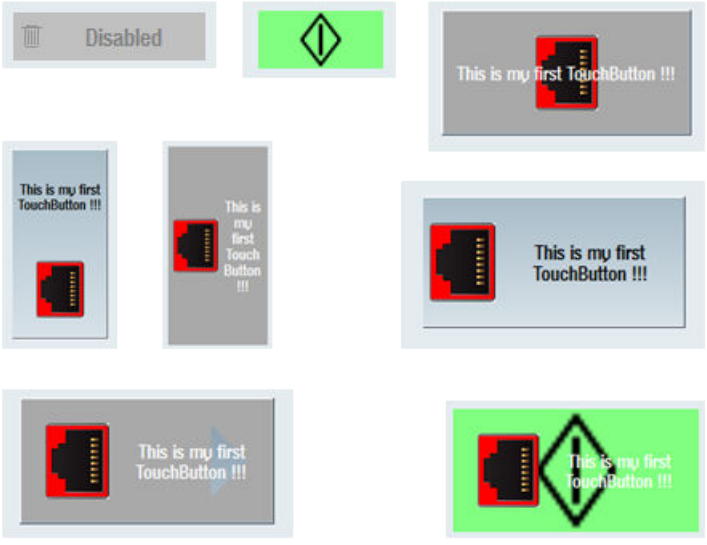


Figura 9-17 Esempi di TouchButton

Esempio

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
```

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
LOAD
   WRITECWPROPERTY("MyTB1", "text", "This is my first TouchButton !!!")
   WRITECWPROPERTY("MyTB1", "textPressed", "This is my first TouchButton (pressed)!!!")
   WRITECWPROPERTY("MyTB1", "picture", "dsm_remove_trashcan_red.png") WRITECWPROPERTY("MyTB1", "pic-
tureAlignment", "left")
   WRITECWPROPERTY("MyTB1", "scalePicture", FALSE)
   WRITECWPROPERTY("MyTB1", "picturePressed", "slsu_topology_empty_round_slot.png") WRITECWPROPER-
TY("MyTB1", "picture", "slsu_topology_empty_slot_left_error.png")
END_LOAD
```

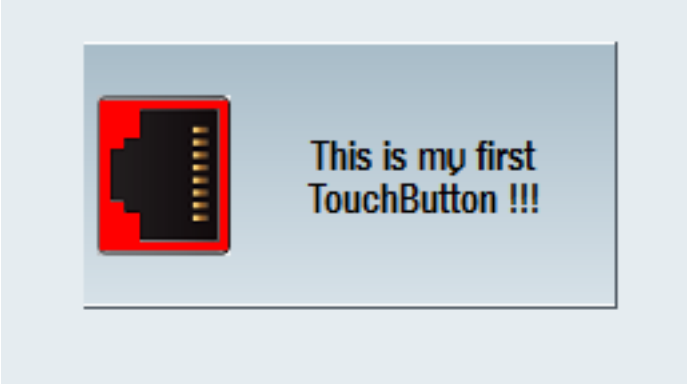


Figura 9-18 Esempio "This is my first TouchButton !!!"

9.6.2 Properties lesen und schreiben

Descrizione

Le properties descritte nel capitolo seguente vengono lette con la funzione ReadCWProperty() e impostate con WriteCWProperty().

Esempi

Lettura della property "Text" del SIEsTouchButton collegato tramite la variabile di visualizzazione "MyTouchButton". Il risultato viene scritto nel registro 0.

```
REG[0] = ReadCWProperty("MyTouchButton", "Text")
```

Scrittura del valore "sk_ok.png" nella property "Picture" del SIEsTouchButton collegato tramite la variabile di visualizzazione "MyTouchButton".

```
WriteCWProperty("MyTouchButton", "Picture", "sk_ok.png")
```

9.6.3 Properties

Panoramica

Property	Descrizione
ButtonStyle	Stile di visualizzazione del TouchButton
Flat	Visualizzazione piana o 3D
Enabled	Attivazione/disattivazione operabilità
Checkable	Attivazione/disattivazione funzionalità toggle
Checked	TouchButton già innestato (checked)
ShowFocusRect	Visualizzazione del rettangolo di attivazione quando il TouchButton riceve lo stato attivo.
Picture	Immagine da visualizzare quando il TouchButton è nello stato normale e non premuto
PicturePressed	Testo da visualizzare quando il TouchButton è rappresentato premuto o innestato (checked)
PictureAlignment PictureAlignmentString	Allineamento dell'immagine
ScalePicture	L'immagine viene scalata (estesa o compressa)
PictureKeepAspectRatio	Comportamento di estensione dell'immagine
Testo	Testo di visualizzazione normale
TextPressed	Testo visualizzato quando il TouchButton è premuto
textAlignment textAlignmentString	Allineamento del testo
TextAlignedToPicture	Orientamento del testo in funzione dell'immagine - attivazione/disattivazione
BackgroundPicture	Immagine di sfondo da visualizzare
BackgroundPictureAlignment BackgroundPictureAlignmentString	Allineamento dell'immagine di sfondo
ScaleBackgroundPicture	L'immagine di sfondo viene scalata (estesa o compressa)
BackgroundPictureKeepAspectRatio	Comportamento di estensione dell'immagine
BackColor	Colore di sfondo quando il TouchButton è nello stato normale e non premuto
BackColorChecked	Colore di sfondo quando il TouchButton è innestato (checked)
BackColorPressed	Colore di sfondo quando il TouchButton viene premuto momentaneamente
BackColorDisabled	Colore di sfondo quando il TouchButton non è azionabile
TextColor	Colore del testo quando il TouchButton è nello stato normale e non premuto
TextColorChecked	Colore del testo quando il TouchButton è innestato (checked)
TextColorPressed	Colore del testo quando il TouchButton viene premuto momentaneamente
TextColorDisabled	Colore del testo quando il TouchButton non è azionabile

ButtonStyle – Stile di visualizzazione

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ButtonStyle**") WriteCWProperty(TouchButtonVarName, "**ButtonStyle**", Value)	
Descrizione:	Legge/imposta lo stile di visualizzazione del TouchButton.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (int)
	Value	Valore da impostare (int) 0 = softkey Look&Feel (impostazione predefinita) 1 = specifico dell'utente

Con l'impostazione "0 = Softkey Look&Feel" il TouchButton si presenta e si comporta esattamente come un softkey. Viene tenuto in considerazione anche lo Skin impostato attualmente – vedere il dato macchina di visualizzazione 9112 = $MM_HMI_SKIN.

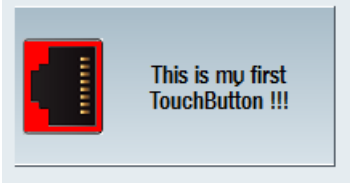


Figura 9-19 ButtonStyle - Softkey Look&Feel

Con l'impostazione "1 = specifico dell'utente" è possibile definire il colore del testo e dello sfondo a seconda dello stato del TouchButton. Permette inoltre di realizzare l'effetto 3D e la scalatura automatica di immagini e di influenzare le distanze dai bordi.

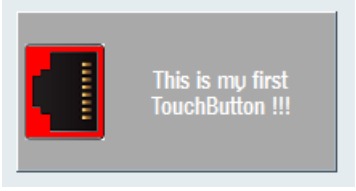


Figura 9-20 ButtonStyle - Specifico dell'utente

Flat – Tipo di rappresentazione

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Flat**") WriteCWProperty(TouchButtonVarName, "**Flat**", Value)	
Descrizione:	Legge/imposta se il TouchButton deve avere una rappresentazione piana (impostazione predefinita) o 3D. **Nota:** questa property è disponibile solo se è attivato lo stile di visualizzazione (ButtonStyle) "1 = Specifico dell'utente".	
	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
Parametri:	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE (valore predefinito) o FALSE

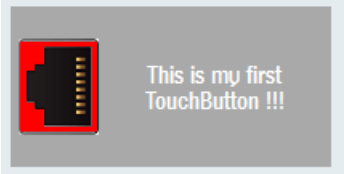


Figura 9-21 Tipo di rappresentazione piana

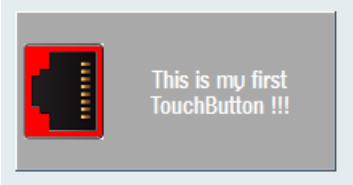


Figura 9-22 Tipo di rappresentazione 3D

Enabled – Operabilità del TouchButton

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Enabled**") WriteCWProperty(TouchButtonVarName, "**Enabled**", Value)	
Descrizione:	Legge/imposta se il TouchButton deve essere azionabile (= TRUE) o non azionabile (FALSE).	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE (valore predefinito) o FALSE

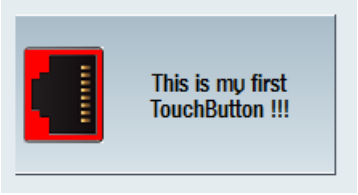


Figura 9-23 TouchButton azionabile

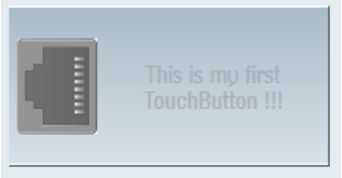


Figura 9-24 TouchButton non azionabile

Nota

Se si fa clic su un TouchButton inattivo, viene emesso il segnale "clickedDisabled". Il segnale può essere valutato e ad es. può essere emesso un messaggio che spiega il motivo per il quale il TouchButton e la funzione ad esso correlata non sono attualmente disponibili. L'immagine visualizzata o l'immagine di sfondo viene rappresentata automaticamente in scala di grigi nello stato disattivato.

Checkable – Attivazione/disattivazione funzionalità toggle

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Checkable**") WriteCWProperty(TouchButtonVarName, "**Checkable**", Value)	
Descrizione:	Legge/imposta la funzionalità toggle del TouchButton.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE o FALSE (valore predefinito)

Nota

Normalmente, una volta rilasciato il TouchButton torna al suo stato normale (analogamente a un pulsante). Se al contrario la funzionalità toggle del TouchButton è attivata (TRUE), dopo la pressione questo rimane inizialmente in posizione premuta. Se viene nuovamente azionato, torna allo stato normale (analogamente a un interruttore).

Vedere anche la Property "Checked".

Checked – Stato toggle attuale

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Checked**") WriteCWProperty(TouchButtonVarName, "**Checked**", Value)	
Descrizione:	Legge/imposta lo stato toggle attuale del TouchButton.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE o FALSE (valore predefinito)

Se la funzione toggle viene impostata a "TRUE" tramite la property "Checkable", si può leggere o impostare lo stato toggle attuale con la property "Checked".

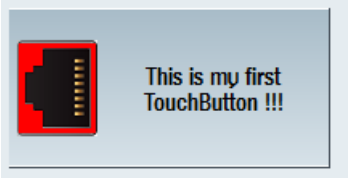


Figura 9-25 Unchecked

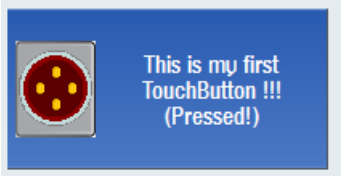


Figura 9-26 Checked

Per i due stati possono essere visualizzati testo e immagine.

Nota

Vedere anche la Property "Checkable".

ShowFocusRect – Visualizzazione del rettangolo di attivazione

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ShowFocusRect**") WriteCWProperty(TouchButtonVarName, "**ShowFocusRect**", Value)	
Descrizione:	Legge/imposta se il TouchButton deve visualizzare un rettangolo di attivazione appena riceve lo stato attivo.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE o FALSE (valore predefinito)

Il colore del riquadro di attivazione viene impostato automaticamente con l'impostazione del colore Operate, nell'esempio seguente "arancione".

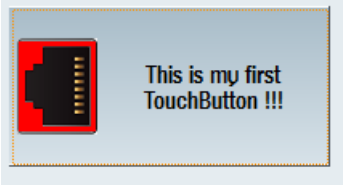


Figura 9-27 ShowFocusRect

Picture – Immagine visualizzata

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Picture**") WriteCWProperty(TouchButtonVarName, "**Picture**", Value)	
Descrizione:	Legge/imposta l'immagine da visualizzare quando il TouchButton è in posizione di riposo (non premuto).	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (String)
	Value	Valore da impostare (String)

Come valore viene specificato il nome file, ad es. "sk_ok.png". Il TouchButton determina automaticamente il file di immagine corretto dalla directory di risoluzione attuale (vedere il capitolo Utilizzo di immagini/grafica (Pagina 64)).

L'immagine visualizzata viene rappresentata automaticamente in scala di grigi nello stato disattivato.

Nota

Vedere anche le Properties "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PicturePressed – Immagine visualizzata in stato premuto

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PicturePressed**") WriteCWProperty(TouchButtonVarName, "**PicturePressed**", Value)	
Descrizione:	Legge/imposta l'immagine da visualizzare quando il TouchButton è premuto o innestato.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (String)
	Value	Valore da impostare (String)

Se non viene specificato un valore (stringa vuota " "), il TouchButton visualizza in questo stato l'immagine impostata con la property "Picture".

Come valore viene specificato il nome file, ad es. "sk_ok.png". Il TouchButton determina automaticamente il file di immagine corretto dalla directory di risoluzione attuale (vedere il capitolo Utilizzo di immagini/grafica (Pagina 64)).

L'immagine visualizzata viene rappresentata automaticamente in scala di grigi nello stato disattivato.

Nota

Vedere anche le Properties "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PictureAlignment – Allineamento dell'immagine

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureAlignment**") WriteCWProperty(TouchButtonVarName, "**PictureAlignment**", Value)	
Descrizione:	Legge/imposta l'orientamento dell'immagine visualizzata sul TouchButton.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (int)
	Value	Valore da impostare (int): 1 = sinistra (impostazione predefinita) 2 = destra 32 = in alto 64 = in basso 128 = centrato

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).
Vedere anche le Properties "Text", "TextAlignedToPicture", "PictureAlignmentString", "ScalePicture", "PictureKeepAspectRatio".

PictureAlignmentString – Orientamento dell'immagine

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureAlignment-String**") WriteCWProperty(TouchButtonVarName, "**PictureAlignmentString**", Value)	
Descrizione:	Legge/imposta l'allineamento dell'immagine visualizzata. Questa property ha lo stesso significato della property "PictureAlignment". Il valore non viene trasferito qui in formato numerico (int), bensì come stringa.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (String)
	Value	Valore da impostare (String): "Left" = sinistra (impostazione predefinita) "Right" = destra "Top" = in alto "Bottom" = in basso "Center" = centrato

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

Vedere anche le Properties "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

ScalePicture - Scalatura dell'immagine visualizzata

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ScalePicture**") WriteCWProperty(TouchButtonVarName, "**ScalePicture**", Value)	
Descrizione:	Legge/imposta se il TouchButton deve scalare l'immagine da visualizzare a seconda dell'allineamento.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE o FALSE (valore predefinito)

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

PictureKeepAspectRatio – Scalatura dell'immagine visualizzata mantenendo le proporzioni

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureKeepAspectRatio**") WriteCWProperty(TouchButtonVarName, "**PictureKeepAspectRatio**", Value)	
Descrizione:	Legge/imposta se nella scalatura (property "ScalePicture" = "TRUE") devono essere mantenute o meno le proporzioni dell'immagine visualizzata.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE (valore predefinito) o FALSE

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

Text – Testo visualizzato

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Text**") WriteCWProperty(TouchButtonVarName, "**Text**", Value)	
Descrizione:	Legge/imposta il testo visualizzato quando il TouchButton è in posizione di riposo (non premuto).	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (String)
	Value	Valore da impostare (String)

I testi vengono troncati automaticamente ai limiti delle parole nel rettangolo disponibile.

Le interruzioni di riga possono essere forzate dal sistema solo se il testo visualizzato proviene da un file di lingua.

Nota

Vedere il capitolo Testi dipendenti dalla lingua (Pagina 273).

TextPressed – Testo visualizzato nello stato premuto

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextPressed**") WriteCWProperty(TouchButtonVarName, "**TextPressed**", Value)	
Descrizione:	Legge/imposta il testo visualizzato quando il TouchButton è premuto o innestato.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (String)
	Value	Valore da impostare (String)

I testi vengono troncati automaticamente ai limiti delle parole nel rettangolo disponibile.

Le interruzioni di riga possono essere forzate dal sistema solo se il testo visualizzato proviene da un file di lingua.

Nota

Vedere il capitolo Testi dipendenti dalla lingua (Pagina 273).

TextAlignment – Allineamento del testo

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextAlignment**") WriteCWProperty(TouchButtonVarName, "**TextAlignment**", Value)	
Descrizione:	Legge/imposta l'allineamento del testo sul TouchButton.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (int)
	Value	Valore da impostare (int): 1 = sinistra 2 = destra 32 = in alto 64 = in basso 128 = centrato (impostazione predefinita) (vedere gli esempi seguenti)

Figura 9-28 TextAlignment - sinistra

Figura 9-29 TextAlignment - destra

Figura 9-30 TextAlignment - in alto

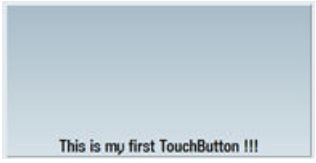


Figura 9-31 TextAlignment - sinistra

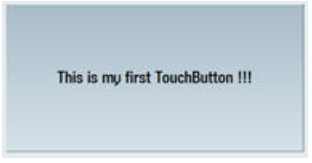


Figura 9-32 TextAlignment - centrato

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

Vedere anche le Properties "Text", "TextAlignmentString", "TextAlignedToPicture".

TextAlignmentString – Allineamento del testo

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextAlignment-String**") WriteCWProperty(TouchButtonVarName, "**TextAlignmentString**", Value)
Descrizione:	Legge/imposta l'allineamento del testo. Questa property ha lo stesso significato della property "TextAlignment". Il valore non viene trasferito qui in formato numerico (int), bensì come stringa.

Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (String)
	Value	Valore da impostare (String): "Left" = sinistra "Right" = destra "Top" = in alto "Bottom" = in basso "Center" = centrato (impostazione predefinita)

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

Vedere anche le Properties "Text", "TextAlignment", "TextAlignedToPicture".

TextAlignedToPicture – Allineamento del testo relativamente all'immagine

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "TextAlignedToPicture") WriteCWProperty(TouchButtonVarName, "**TextAlignedToPicture**", Value)	
Descrizione:	Legge/imposta se il testo visualizzato deve essere posizionato relativamente all'immagine. Se qui si imposta FALSE, il testo viene visualizzato centrato sul TouchButton.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE (valore predefinito) o FALSE

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).
Vedere anche le Properties "Text", "TextPressed", "Picture", "PicturePressed".

BackColor – Colore di sfondo

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, " **BackColor**") WriteCWProperty(TouchButtonVarName, " **BackColor**", Value)	
Descrizione:	Legge/imposta il colore di sfondo quando il TouchButton è in posizione di riposo (non premuto).	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Colore letto delle Properties (String)
	Value	Valore da impostare (String) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

Nota

Questa property è disponibile solo se è attivato lo stile di visualizzazione "1 – Specifico dell'utente".

BackColorChecked – Colore di sfondo nello stato innestato

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorChecked**") WriteCWProperty(TouchButtonVarName, " **BackColorChecked**", Value)	
Descrizione:	Legge/imposta il colore di sfondo quando il TouchButton si trova in stato innestato (funzione toggle). Vedere le properties "Checked", "Checkable"	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Colore letto delle Properties (String)
	Value	Valore da impostare (String) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

Nota

Questa property è disponibile solo se è attivato lo stile di visualizzazione "1 – Specifico dell'utente".

Nota

Vedere anche le Properties "Checked", "Checkable".

BackColorPressed – Colore di sfondo nello stato premuto

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorPressed**") WriteCWProperty(TouchButtonVarName, " **BackColorPressed**", Value)	
Descrizione:	Legge/imposta il colore di sfondo quando il TouchButton è nello stato premuto.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Colore letto delle Properties (String)
	Value	Valore da impostare (String) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

Nota

Questa property è disponibile solo se è attivato lo stile di visualizzazione "1 – Specifico dell'utente".

BackColorDisabled – Colore di sfondo nello stato disattivato

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorDisabled**") WriteCWProperty(TouchButtonVarName, " **BackColorDisabled**", Value)	
Descrizione:	Legge/imposta il colore di sfondo quando il TouchButton è nello stato disattivato.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Colore letto delle Properties (String)
	Value	Valore da impostare (String) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

Nota

Questa property è disponibile solo se è attivato lo stile di visualizzazione "1 – Specifico dell'utente".

Nota

Vedere anche la property "Enabled".

TextColor – Colore del testo

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColor**") WriteCWProperty(TouchButtonVarName, " **TextColor**", Value)	
Descrizione:	Legge/imposta il colore del testo quando il TouchButton è in posizione di riposo (non premuto).	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Colore letto delle Properties (String)
	Value	Valore da impostare (String) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

Nota

Questa property è disponibile solo se è attivato lo stile di visualizzazione "1 – Specifico dell'utente".

TextColorChecked – Colore del testo nello stato innestato

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorChecked**") WriteCWProperty(TouchButtonVarName, " **TextColorChecked**", Value)
Descrizione:	Legge/imposta il colore del testo quando il TouchButton si trova in stato innestato (funzione end).

Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Colore letto delle Properties (String)
	Value	Valore da impostare (String) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

Nota

Questa property è disponibile solo se è attivato lo stile di visualizzazione "1 – Specifico dell'utente".

Nota

Vedere anche le Properties "Checked", "Checkable".

TextColorPressed – Colore del testo nello stato premuto

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorPressed**") WriteCWProperty(TouchButtonVarName, "**TextColorPressed**", Value)	
Descrizione:	Legge/imposta il colore del testo quando il TouchButton è nello stato premuto.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Colore letto delle Properties (String)
	Value	Valore da impostare (String) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

Nota

Questa property è disponibile solo se è attivato lo stile di visualizzazione "1 – Specifico dell'utente".

TextColorDisabled – Colore del testo nello stato disattivato

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorDisabled**") WriteCWProperty(TouchButtonVarName, "**TextColorDisabled**", Value)	
Descrizione:	Legge/imposta il colore del testo quando il TouchButton è nello stato disattivato.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Colore letto delle Properties (String)
	Value	Valore da impostare (String) come valore RGB in formato "#RRGGBB", ad es. "#04B7FB"

Nota

Questa property è disponibile solo se è attivato lo stile di visualizzazione "1 – Specifico dell'utente".

Nota

Vedere anche la property "Enabled".

BackgroundPicture – Immagine di sfondo

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPicture**") WriteCWProperty(TouchButtonVarName, "**BackgroundPicture**", Value)	
Descrizione:	Legge/imposta l'immagine di sfondo permanente, ossia indipendente dallo stato, che deve essere visualizzata.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (String)
	Value	Valore da impostare (String)

Come valore viene specificato il nome file, ad es. "sk_ok.png". Il TouchButton determina automaticamente il file di immagine corretto dalla directory di risoluzione attuale (vedere il capitolo Utilizzo di immagini/grafica (Pagina 64)).

L'immagine di sfondo viene visualizzata automaticamente in scala di grigi nello stato disattivato.

Nota

Vedere il capitolo Utilizzo di immagini/grafica (Pagina 64).

Vedere anche le Properties "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureAlignment – Orientamento dell'immagine di sfondo

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureAlignment**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureAlignment**", Value)
Descrizione:	Legge/imposta l'allineamento dell'immagine di sfondo.

Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (int)
	Value	Valore da impostare (int): 1 = sinistra 2 = destra 32 = in alto 64 = in basso 128 = centrato (impostazione predefinita)

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

Vedere anche le Properties "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureAlignmentString – Orientamento dell'immagine di sfondo

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureAlignmentString**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureAlignmentString**", Value)	
Descrizione:	Legge/imposta l'allineamento dell'immagine di sfondo. Questa property ha lo stesso significato della property "BackgroundPictureAlignment". Il valore non viene trasferito qui in formato numerico (int), bensì come stringa.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (int)
	Value	Valore da impostare (int): "Left" = sinistra "Right" = destra "Top" = in alto "Bottom" = in basso "Center" = centrato (impostazione predefinita)

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

Vedere anche le Properties "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

ScaleBackgroundPicture – Scalatura dell'immagine di sfondo

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ScalePicture**") WriteCWProperty(TouchButtonVarName, "**ScalePicture**", Value)	
Descrizione:	Legge/imposta se il TouchButton deve scalare l'immagine da visualizzare a seconda dell'allineamento.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE o FALSE (valore predefinito)

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

Vedere anche le Properties "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureKeepAspectRatio – Scalatura dell'immagine di sfondo mantenendo le proporzioni

Sintassi:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureKeepAspectRatio**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureKeepAspectRatio**", Value)	
Descrizione:	Legge/imposta se nella scalatura (property "ScaleBackgroundPicture" = "TRUE") devono essere mantenute o meno le proporzioni dell'immagine visualizzata.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Valore letto delle Properties (bool)
	Value	Valore da impostare (bool): TRUE (valore predefinito) o FALSE

Nota

Vedere il capitolo Posizionamento e allineamento di immagine e testo (Pagina 271).

9.6.4 Funktionen

Le funzioni indicate di seguito possono essere richiamate con la funzione CallCWMethod().

Esempio

Impostazioni delle distanze del SIEsTouchButton collegato tramite la variabile di visualizzazione "MyTouchButton".

Il risultato viene scritto nel registro 0.

```
REG[0] = CallCWMethod("MyTouchButton", "setMargins", 20, 20, 20, 20,
20)
```

Panoramica

Funzione	Descrizione
setMargins	Impostazione delle distanze
serialize	Salvataggio/ripristino dello stato attuale

setMargins – Impostazione delle distanze

Sintassi:	ReturnValue = CallCWMethod(TouchButtonVarName, "**setMargins**", left, top, right, bottom, center) ReturnValue = CallCWMethod(TouchButtonVarName, "**setMargins**", left, top, right, bottom, center, marginAlignment)	
Descrizione:	Questa funzione permette di impostare i margini e la distanza tra immagine e testo. Se viene specificato un valore con "-1", vale la distanza predefinita del sistema. Se il valore è maggiore o uguale a "0", il valore viene impostato come margine.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Errorcode (bool): TRUE = riuscito
	left	Margine sinistro (int)
	top	Margine superiore (int)
	right	Margine destro (int)
	bottom	Margine inferiore (int)
	center	Distanza tra immagine e testo (int), se l'impostazione Alignment non è "center"
	marginAlignment	Opzionale: determina se il margine deve avere effetto, oltre che per l'immagine visualizzata, anche per il testo visualizzato (bool), TRUE (impostazione predefinita)

serialize – Salvataggio/ripristino dello stato corrente

Sintassi:	ReturnValue = CallCWMethod(TouchButtonVarName, "**serialize**", FilePath, IsStoring)	
Descrizione:	Questa funzione consente di scrivere in formato binario e ripristinare ove necessario lo stato attuale e il contenuto del SIEsTouchButton in un file o DataStream. Questa funzione aggiorna automaticamente la visualizzazione.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SIEsTouchButton
	Return Value	Errorcode (bool): TRUE = riuscito
	FilePath	Percorso completo con nome del file (QString)
	IsStoring	Salvataggio/ripristino (bool): TRUE = salvataggio

Nota

La funzione non viene utilizzata direttamente dall'addetto alla progettazione!

9.6.5 Signale

I segnali descritti di seguito possono essere rilevati nella progettazione per attivare la reazione corrispondente.

Panoramica

Funzione	Descrizione
clickedDisabled	Azionamento di un TouchButton disattivato
clicked	È stato eseguito un clic o un tocco
checked	È stata effettuata un'azione di toggle

- Se un TouchButton è azionabile e del tipo non toggle, quando viene premuto viene inviato il segnale "clicked".
- Se un TouchButton è azionabile e del tipo toggle, quando viene premuto viene inviata la seguente sequenza di segnali:
  "checked" → "clicked"
- Se un TouchButton non è azionabile, quando viene premuto viene inviato il segnale "clickedDisabled".

Tutti i segnali citati in precedenza vengono sempre inviati dopo una manovra operativa, ossia dopo aver rilasciato il pulsante del mouse o la barra spaziatrice oppure dopo aver toccato lo schermo Multitouch.

Di conseguenza, il SIEsTouchButton ha una pura funzione di interruttore, ossia non consente di realizzare la funzionalità di un tasto.

Nota

Nel caso di comandi Multitouch, tenere presente che viene inviato un clic solo dopo che un tocco (pressione e rilascio entro circa 0,7 secondi) è stato completamente riconosciuto. Se già alla semplice pressione venisse eseguita un'azione, sarebbe ad es. impossibile uno scorrimento/spostamento della ScrollArea retrostante oppure potrebbero verificarsi facilmente manovre errate indesiderate.

clicked – È stato eseguito un clic su un TouchButton

Sintassi:	SUB(on_<TouchButtonVarName>_clicked) ... END_SUB	
Descrizione:	Una sequenza completa di pressione e rilascio di un TouchButton azionabile genera il segnale "clicked". Si consiglia di lavorare prevalentemente con questo segnale.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SlEsTouchButton
	SIGARG[0]	Fornisce lo stato toggle del TouchButton (bool)

Esempio

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_clicked)
   DLGL("checked: " << SIGARG[0])
END_SUB
```

checked – Un TouchButton è stato commutato

Sintassi:	SUB(on_<TouchButtonVarName>_checked) ... END_SUB	
Descrizione:	Un TouchButton toggle è stato premuto, per cui il suo stato è cambiato.	
Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SlEsTouchButton
	SIGARG[0]	Fornisce lo stato toggle del TouchButton (bool)

Esempio

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_toggled)
   DLGL("toggled: " << SIGARG[0])
END_SUB
```

clickedDisabled – È stato eseguito un clic su un TouchButton non azionabile

Sintassi:	SUB(on_<TouchButtonVarName>_clickedDisabled) ... END_SUB
Descrizione:	Una sequenza completa di pressione e rilascio di un TouchButton non azionabile genera il segnale "clickedDisabled".

Parametri:	TouchButtonVarName	Nome della variabile di visualizzazione che contiene un SlEsTouchButton
	SIGARG[0]	Fornisce lo stato toggle del TouchButton (bool)

Esempio

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_clickedDisabled)
   DLGL("checkedDisabled: " << SIGARG[0])
END_SUB
```

9.6.6 Posizionamento e allineamento di immagine e testo

Posizionamento di immagini

Con l'allineamento (Alignment) preimpostato si posiziona un'immagine nel seguente modo:

- In un primo tempo viene determinata l'immagine adatta alla risoluzione corrente in base alla rispettiva directory della risoluzione.
- Dopodiché il rettangolo della superficie (ClientArea) viene ridotto delle distanze dai margini impostate (MarginArea).
  La MarginArea può essere influenzata con la funzione "setMargins":
  - setMargins(-1, -1, -1, -1, -1)

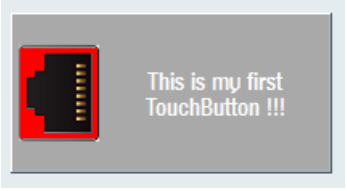


Figura 9-33 setMargins(-1, -1, -1, -1, -1)

  - setMargins(0, 0, 0, 0, 0)

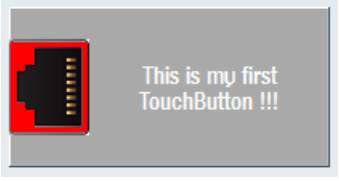


Figura 9-34 setMargins(0, 0, 0, 0, 0)

  - setMargins(20, 20, 20, 20, 20)

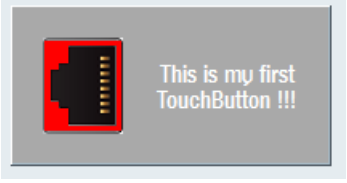


Figura 9-35 setMargins(20, 20, 20, 20, 20)

Esempio di allineamento "a sinistra"

La superficie viene dimezzata orizzontalmente per cui l'immagine da visualizzare può occupare al massimo la metà della superficie.

L'immagine viene quindi tracciata nel seguente modo:

- Property "scalePicture": FALSE
  L'immagine viene tracciata in questo punto senza scalatura. Accertarsi che l'immagine di base non sia troppo grande e che possa essere rappresentata per intero all'interno del TouchButton.

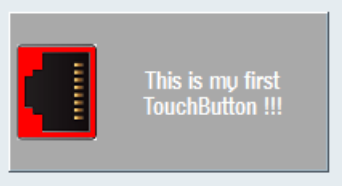


Figura 9-36 scalePicture FALSE

- Property "scalePicture": TRUE
  L'immagine viene scalata (estesa o compressa) finché non rientra esattamente nella metà sinistra del TouchButton. La property "pictureKeepAspectRatio" viene considerata come segue:
  - Property "pictureKeepAspectRatio": FALSE
    L'immagine viene scalata in orizzontale e in verticale finché non rientra esattamente nella metà sinistra del TouchButton. Non viene più mantenuto il rapporto tra altezza e larghezza dell'immagine originale.

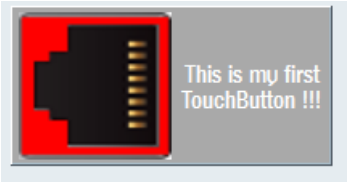


Figura 9-37 scalePicture - pictureKeepAspectRatio FALSE

  - Property "pictureKeepAspectRatio": TRUE
    L'immagine viene scalata con le dimensioni maggiori possibili nella metà sinistra del TouchButton tenendo conto delle proporzioni dell'immagine originale.

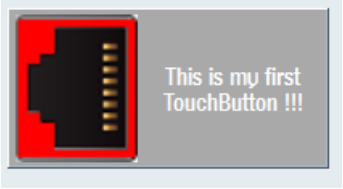


Figura 9-38 scalePicture - pictureKeepAspectRatio TRUE

Nota

Per gli orientamenti "a destra", "in alto" e "in basso" vale lo stesso principio.

Con l'allineamento "centrato" si tenta di adattare l'immagine all'area completa della MarginArea in funzione delle properties "scalePicture" e "pictureKeepAspectRatio".

Posizionamento del testo

A seconda della property "TextAlignedToPicture" il testo viene posizionato nel seguente modo:

- Property "textAlignedToPicture": FALSE
  Nella MarginArea il testo viene visualizzato centrato verticalmente e orizzontalmente.

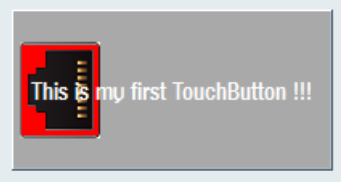


Figura 9-39 textAlignedToPicture FALSE

- Property "textAlignedToPicture": TRUE
  Il testo viene visualizzato centrato orizzontalmente e verticalmente nello spazio rimanente della MarginArea una volta sottratta l'area dell'immagine tracciata.

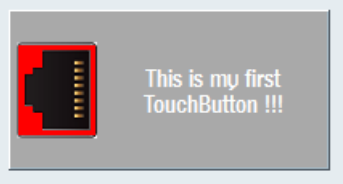


Figura 9-40 textAlignedToPicture TRUE

9.6.7 Testi dipendenti dalla lingua

Il SIEsTouchButton non supporta una funzione autonoma per le lingue straniere. Tuttavia è possibile realizzare una dipendenza dalla lingua, come illustrato nell'esempio che segue:

Esempio

easyscreen.ini:

```
[LANGUAGEFILES]LngFile03 = user.txt
```

user_eng.txt:

```
85000 0 0 "This is my first Touchbutton !!!"
```

user_deu.txt:

```
85000 0 0 "Questo è il mio primo Touchbutton !!!"
```

File di progettazione:

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
LOAD
```

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
   WRITECWPROPERTY("MyTB1", "text", $85000)
END_LOAD
LANGUAGE
   WRITECWPROPERTY("MyTB1", "text", $85000)
END_LANGUAGE
```

I testi vengono troncati automaticamente ai limiti delle parole nel rettangolo disponibile. Un'interruzione di riga forzata è possibile solo se il testo da visualizzare proviene da un file della lingua. In questo caso, occorre inserire un "%n" nel punto corrispondente del testo (vedere l'esempio che segue).

user_eng.txt:

```
85001 0 0 "This is%nmy%nfirst%nTouchButton !!!"
```

File di progettazione:

```
WRITECWPROPERTY("MyTB1", "text", $85001)
```

Risultato

Figura 9-41 Esempio di interruzione di riga nel testo dipendente dalla lingua

Settore operativo Custom 10

10.1 Come attivare il settore operativo "Custom"

Attivazione del settore operativo "Custom"

Il settore operativo "Custom" non è attivato al momento della consegna.

1. Copiare prima il file "slamconfig.ini" dalla [cartella di sistema siemens]/cfg alla [cartella di sistema oem]/cfg
   oppure alla [cartella di sistema addon]/cfg o alla [cartella di sistema user]/cfg.
2. Per attivare il settore operativo "Custom", occorre immettere:
   ```
   [Custom]
   Visible=True
   ```

Risultato

Dopo l'attivazione il softkey per il settore operativo "Custom" si trova nel menu principale (F10) nella barra di avanzamento del menu su HSK4 (= impostazione predefinita).

Il settore operativo "Custom" mostra una finestra vuota sopra l'intero settore operativo con un titolo progettabile. Tutti i softkey orizzontali e verticali sono progettabili.

10.2 Come progettare il softkey per "Custom"

Progettazione del softkey per il settore operativo "Custom"

Nel file "slamconfig.ini" vengono progettate la dicitura e la posizione del softkey per il settore operativo "Custom".

Per la progettazione del softkey di accesso esistono le seguenti possibilità:

1. Per sostituire la dicitura del softkey con un **testo dipendente dalla lingua**, occorre immettere i seguenti dati nella sezione [Custom]:
   ```
   TextId=MY_TEXT_ID
   TextFile=mytextfile
   TextContext=mycontext
   ```
   In questo esempio il softkey mostra il testo dipendente dalla lingua che è stato memorizzato con il testo "MY_TEXT_ID" nel file di testo mytextfile_xxx.qm sotto "MyContext" (xxx sta qui per il codice della lingua).
2. Per sostituire la dicitura del softkey con un **testo indipendente dalla lingua**, occorre immettere i seguenti dati nella sezione [Custom]:
   ```
   TextId=HELLO
   TextFile=<empty>
   TextContext=<empty>
   ```
   In questo esempio il softkey per il settore operativo "Custom" mostra in ogni lingua il testo "HELLO".
3. Oltre al testo sul softkey può anche essere visualizzato **un pittogramma**.
   A questo scopo occorre immettere i seguenti dati nella sezione [Custom]:
   ```
   Picture=mypicture.png
   ```
   Il softkey visualizza quindi il pittogramma del file mypicture.png. Grafici e bitmap sono memorizzati nel seguente percorso: [cartella di sistema oem]/ico/ico<Risoluzione>. Va utilizzata la directory corrispondente alla risoluzione del display.
4. È inoltre possibile impostare **la posizione**del softkey. A questo scopo sono disponibili i seguenti dati nella sezione [Custom]:
   ```
   SoftkeyPosition=12
   ```
   La posizione 12 è l'impostazione predefinita. Ciò corrisponde a HSK4 nella barra di avanzamento del menu del settore operativo. La posizione 1-8 corrisponde a HSK1 ... HSK8 nella barra di menu, la posizione 9-16 corrisponde a HSK1 ... HSK8 nella barra di avanzamento del menu.

10.3 Come progettare il settore operativo "Custom"

Progettazione del softkey per il settore operativo "Custom"

Per progettare il settore operativo, occorrono i file "easyscreen.ini" e "custom.ini". I modelli per entrambi i file si trovano nella directory: [cartella di sistema siemens]/templates/cfg.

1. Copiare prima i file nella [cartella di sistema oem]/cfg ed apportare qui le modifiche.
2. Nel file "easyscreen.ini" è già contenuta una riga di definizione per il settore operativo "Custom":
   ```
   ;StartFile02 = area := Custom, dialog := SlEsCustomDialog,
   startfile := custom.com
   ```
   Il ";" all'inizio della riga rappresenta il carattere di commento. La riga è quindi commentata e pertanto non attiva. Per modificare questo, occorre cancellare il ";".
   Con l'attributo "startfile" in questa riga si definisce che quando si seleziona il settore operativo "Custom" il dato rimanda al file di progetto "custom.com".

3. Creare il **file di progetto "custom.com"** nella [cartella di sistema oem]/proj. Qui è contenuta la progettazione che viene creata analogamente al file aeditor.com del settore operativo "Programma". I softkey di accesso progettati vengono visualizzati nel settore operativo "Custom".
4. Nel file "custom.ini" si progetta il **testo indipendente dalla lingua** per la riga del titolo della finestra di dialogo.
   A questo scopo nel modello è presente il seguente dato:
   ```
   [Header]
   Text=Custom
   ```
   Questo testo può essere sostituito con un testo specifico dell'utente.
5. Per progettare la **pagina iniziale** del settore operativo "Custom", nel modello è presente il seguente dato:
   ```
   [Picture]
   Picture=logo.png
   ```
   Logo.png è il nome della pagina iniziale che viene visualizzata nella finestra di dialogo iniziale del settore operativo "Custom". Qui si può visualizzare ad es. il logo di una ditta o un'altra immagine. Il file deve essere salvato nella directory della risoluzione corrispondente: [cartella di sistema oem]/ico/ ...
6. Per visualizzare direttamente una determinata maschera "Run MyScreens" quando compare per la prima volta il settore operativo "Custom", il modello contiene la voce seguente:
   ```
   ; Mask shown with startup of area "custom"
   [Startup]
   ;Startup = Mask:=MyCustomStartupMask, File:=mycustommasks.com
   ```
   Un caso applicativo tipico è, ad es., l'avvio di SINUMERIK Operate direttamente con una determinata maschera "Run MyScreens".
   A questo scopo si deve impostare come segue nel file di configurazione "systemconfiguration.ini", sezione "[miscellaneous]", la chiave "startuparea":
   ```
   [miscellaneous]
   ;name of the area to be shown at system startup
   startuparea = Custom
   ```

10.4 Esempio di programmazione per il settore "Custom"

Esempio

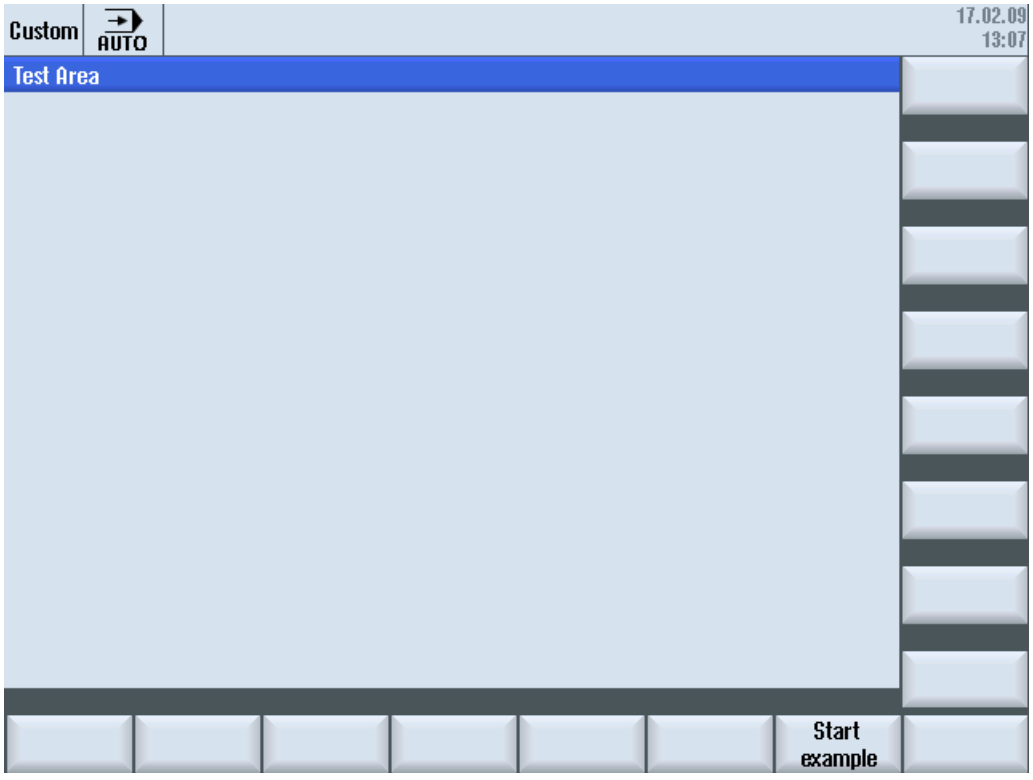


Figura 10-1 Esempio con softkey "Start example"

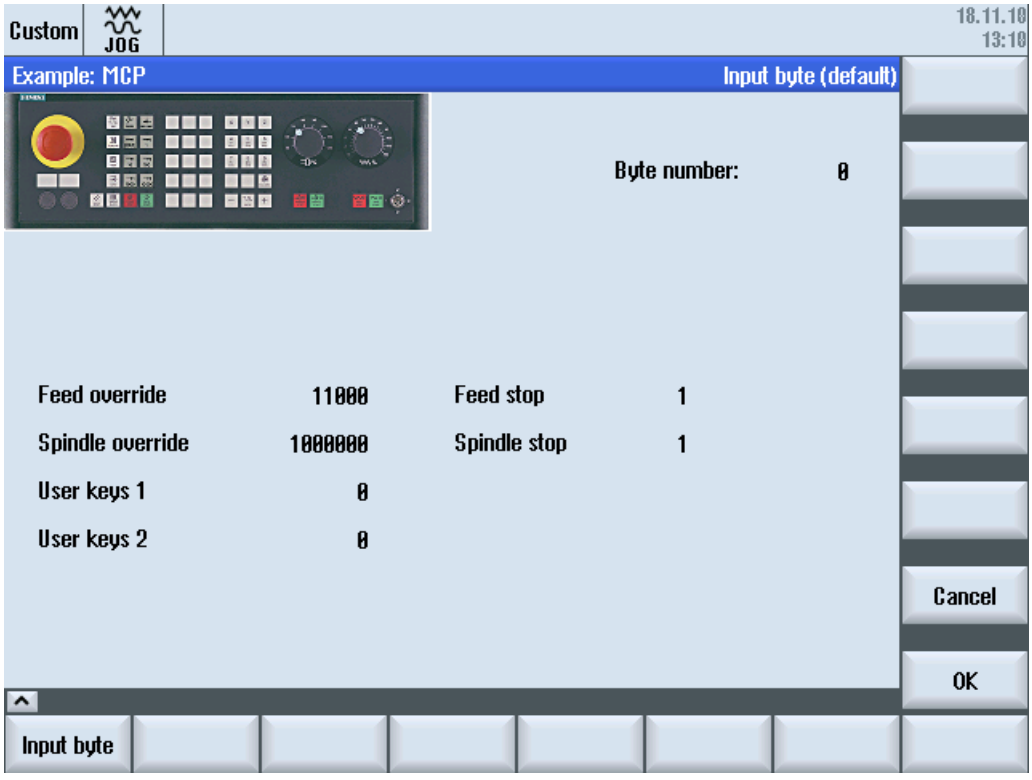


Figura 10-2 Esempio con bitmap e campi di testo

Panoramica file

Sono necessari i seguenti file:

- "custom.com"
- "easyscreen.ini"

Programmazione

Contenuto del file "custom.com":

Nota

Il file grafico "mcp.png" incorporato vale solo a titolo di esempio. Se si desidera riprogrammare questo esempio, va sostituito con un file grafico disponibile.

```
//S(Start)
   HS7=("Start example", se1, ac7)
   PRESS(HS7)
      LM("Maske4")
   END_PRESS
```

```
//END
//M(Maske4/"Example: MCP"/"mcp.png")
   DEF byte=(I/0/0/"Input byte=0 (default)","Byte number:",""/wr1,li1///380,40,100/480,40,50)
   DEF Feed=(IBB//0/"","Feed override",""/wr1//"EB3"/20,180,100/130,180,100), Axistop=(B//0/"","Feed
stop",""/wr1//"E2.2"/280,180,100/380,180,50/100)
   DEF Spin=(IBB//0/"","Spindle override",""/wr1//"EB0"/20,210,100/130,210,100), spinstop=(B//
0/"","Spindle stop",""/wr1//"E2.4"/280,210,100/380,210,50/100)
   DEF custom1=(IBB//0/"," User keys 1",""/wr1//"EB7.7"/20,240,100/130,240,100)
   DEF custom2=(IBB//0/"","User keys 2",""/wr1//"EB7.5"/20,270,100/130,270,100)
   DEF By1
   DEF By2
   DEF By3
   DEF By6
   DEF By7

   HS1=("Input byte", SE1, AC4)
   HS2=("")
   HS3=("")
   HS4=("")
   HS5=("")
   HS6=("")
   HS7=("")
   HS8=("")
   VS1=("")
   VS2=("")
   VS3=("")
   VS4=("")
   VS5=("")
   VS6=("")
   VS7=("Cancel", SE1, AC7)
   VS8=("OK", SE1, AC7)

   PRESS(VS7)
      EXIT
   END_PRESS

   PRESS(VS8)
      EXIT
   END_PRESS

   LOAD
      By1=1
      By2=2
      By3=3
      By6=6
```

```
      By7=7
   END_LOAD

   PRESS(HS1)
      Byte.wr=2
   END_PRESS

   CHANGE(Byte)
      By1=byte+1
      By2=byte+2
      By3=byte+3
      By6=byte+6
      By7=byte+7
      Feed.VAR="EB"<<By3
      Spin.VAR="EB"<<Byte
      Custom1.VAR="EB"<<By6
      Custom2.VAR="EB"<<By7
      Axisstop.VAR="E"<<By2<<".2"
      Spinstop.VAR="E"<<By2<<".4"
      Byte.wr=1
   END_CHANGE

   CHANGE(Axis stop)
      IF Axistop==0
         Axistop.BC=9
      ELSE
         Axistop.BC=11
      ENDIF
   END_CHANGE

   CHANGE(Spin stop)
      IF Spinstop==0
         Spinstop.BC=9
      ELSE
         Spinstop.BC=11
      ENDIF
   END_CHANGE
//END
```

Selezione di finestre di dialogo 11

11.1 Selezione di finestre di dialogo tramite softkey PLC

Progettazione

Descrizione della procedura:

- nel file "systemconfiguration.ini" esiste una sezione [keyconfiguration]. La voce specifica un'azione per un softkey speciale del PLC.
- Come azione viene indicato un numero. Se il numero è maggiore o uguale a 100, si tratta di un richiamo "Run MyScreens".
- Per definire l'azione da eseguire, nel file "easyscreen.ini" deve essere creata una sezione il cui nome si ottiene dal nome del settore operativo e dal nome della finestra di dialogo (vedere la voce in [keyconfiguration] → Area:=..., Dialog:=...) → [_] → ad es. [AreaParameter_SlPaDialog]
- In questa sezione vengono definiti i numeri di azione (specificati nel file "systemconfiguration.ini" → vedere Action:=...). Qui si tratta di due istruzioni:
  1. LS("barra_softkey1","param.com") ... caricamento di una barra dei softkey
  2. LM("maschera1","param.com") ... caricamento di una maschera

Selezione di barre dei softkey tramite softkey del PLC

In "Run MyScreens" è possibile selezionare barre dei softkey "Run MyScreens" e finestre di dialogo "Run MyScreens" tramite softkey del PLC. A questo scopo occorre che l'attributo "action", da specificarsi nella progettazione dei relativi softkey del PLC, abbia un valore maggiore o uguale a 100.

La progettazione dei softkey del PLC avviene nel file "systemconfiguration.ini" nella sezione [keyconfiguration]:

```
[keyconfiguration]
KEY75.1 = Area:=area, Dialog:=dialog, Screen:=screen, Action:= 100,
```

La progettazione dei comandi LM e LS, che devono essere eseguiti con i corrispondenti softkey del PLC, avviene nel file "easyscreen.ini" e in particolare in sezioni il cui nome si compone in base allo schema seguente:

`[areaname_dialogname]`	La prima parte del nome "areaname" definisce il settore operativo, la seconda parte "dialogname" definisce la finestra di dialogo per la quale valgono i comandi progettati in questa sezione.

`[AreaParameter_SlPaDialog]`
`100.screen1 = LS("Softkey1","param.com")`
`101.screen3 = LM("Maschera1","param.com")`
	Devono essere utilizzati i nomi assegnati nel file "systemconfiguration.ini" per il settore operativo e la finestra di dialogo. L'indicazione della finestra di dialogo è opzionale.
	In particolare, può essere omessa per i settori operativi che vengono implementati tramite un'unica finestra di dialogo, vedere l'esempio a lato.
	Se nel settore operativo AreaParameter, implementato con la finestra di dialogo SlPaDialog, viene visualizzato "screen1", alla comparsa di "action" con il valore 100 viene eseguito il comando "LS("Softkey1","param.com")".

`action.screen=comando`	I due attributi "action" e "screen" indicano chiaramente quando viene eseguito il comando specificato.
	L'indicazione di "screen" è opzionale.
	I comandi consentiti sono:
	`LM (LoadMask)`
	`LS (LoadSoftkeys)`

11.1.1 Richiamo delle maschere dei cicli Run MyScreens nell'editor di programma

Campo d'impiego

Tramite i tasti PLC si possono aprire direttamente le maschere dei cicli Run MyScreens nell'editor di programma.

Presupposto

Un ciclo tecnologico può essere selezionato solo quando l'editor di programma è aperto. Il ciclo tecnologico parametrizzato tramite la maschera viene inserito nella posizione in cui si trova il cursore nell'editor di programma.

Procedura

1. Creare il richiamo del ciclo Run MyScreens tramite gli hardkey (Pagina 286) PLC.
2. Per poter integrare correttamente la maschera RunMyScreens nell'editor di programma, è necessario espandere la progettazione dei PLC Keys come descritto nel seguente esempio:
   Esempio di una progettazione completa di Keys:
   systemconfiguration.ini
   ```
   [keyconfiguration]
   KEY101.0 = action:=209, runmyscreenmode:=HandledByDialog
   ```
   easyscreen.ini
   ```
   [AreaProgramEdit_SlProgramEdit]
   209 = LM("MASK_E_DR_O1_MAIN","e_dr_o1.com")
   ```
3. Tramite l'interfaccia PLC "DB19.DBW24" si può verificare il feedback della maschera dei cicli Run MyScreens aperta.
   Esempio:
   In questo esempio, quando viene aperta la maschera Run MyScreens il valore "9876" viene scritto nell'interfaccia PLC:
   Progettazione con PLC-ID=9876:
   //M(MASK_E_DR_O1_MAIN/$85305////XG1,FA2,TA7//"drilling.txt"/**9876**)
   oppure
   //M{ MASK_E_DR_O1_MAIN, HD=$85305, LANGFILELIST=" drilling.txt", **PLC_ID=9876**}
   Vedere anche: Manuale di guida alle funzioni SINUMERIK 840D sl Funzioni di base

Stato dell'editor di programma sull'interfaccia OA

Lo stato dell'editor di programma si può controllare con la variabile CAP locale "/Hmi/StepEditorInfo".

Nome della variabile:	/Hmi/StepEditorInfo
Descrizione della variabile:	Informazioni dell'editor di programma che è attivo al momento
La stringa contiene le seguenti informazioni	
[fileName]	Percorso del programma
[channel]	Numero del canale
[technology]	Tecnologia
[appearance]	Codice del programma (ShopMill/ShopTurn/codice G)
[state]	Stato per StepEditor: • EditorClosed: L'editor è chiuso (ad es. HMI non nel settore operativo Programma) • EditEnabled: Modifiche consentite • EditDisabled: Modifiche non consentite (ad es. readOnly o posizione cursore non valida)

11.2 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"
HK5	ALARM	Selezione settore operativo "Diagnostica", finestra di dialogo "Lista allarmi"
HK6	CUSTOM	Selezione del settore operativo "Custom"
HK7 [1)]	MENU SELECT	Selezione "Menu principale"
HK8 [1)]	MENU FUNCTION	Selezione del settore operativo "Function"
HK9 [1)]	MENU USER	Selezione del settore operativo "User"

1) solo per 828D

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 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:

```
; Hardkey PLC (KEY50-KEY254)
[keyconfiguration]
KEY50.0 = name := AreaMachine, dialog := SlMachine
KEY51.0 = name := AreaParameter, dialog := SlParameter
KEY52.0 = name := AreaProgramEdit, dialog := SlProgramEdit
KEY53.0 = name := AreaProgramManager, dialog := SlPmDialog
```

```
KEY54.0 = name := AreaDiagnosis, dialog := SlDgDialog
KEY55.0 = name := AreaStartup, dialog := SlSuDialog
KEY56.0 = name := Custom, dialog := SlEsCustomDialog
```

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

11.3 Selezione di finestre di dialogo tramite NC

Comando MMC in HMI Operate

I comandi MMC possono essere utilizzati come descritto di seguito:

1. **Definizione di comandi MMC**
   Nel file standard "systemconfiguration.ini" sono presenti le seguenti combinazioni:
   address:=MCYCLES --> command:=LM
   address:=CYCLES --> command:=PICTURE_ON
   Questa distinzione è necessaria per distinguere tra cicli di misura e altri cicli. Questo significa che:
   - LM vale sempre per i cicli di misura, PICTURE_ON sempre per i cicli non di misura
   - I nuovi comandi MMC definiti non possono essere denominati PICTURE_ON e LM
2. **Licenza "Run MyScreens"**
   Tutte le finestre di dialogo aperte da "Run MyScreen", tranne i cicli di misura, sono coperte dalla licenza "Run MyScreens" e quindi solo un numero ristretto di finestre può essere utilizzato senza licenza.
   Esempio di richiamo con "test.com" (Run MyScreens):

```
g0 f50
MMC("CYCLES,PICTURE_ON,test.com,maschera1","A")
m0
MMC("CYCLES,PICTURE_OFF","N")
M30
```

Esempi di maschere di ciclo 12

12.1 Esempi di maschere di ciclo

Con l'installazione di SINUMERIK Operate vengono memorizzati anche esempi di maschere di ciclo create con Run MyScreens.

Gli esempi relativi a Run MyScreens sono disponibili nei seguenti percorsi:

[cartella di sistema Siemens]		
...\siemens\sinumerik\hmi\template\easyscreen\		
...		
	standard\	Figure di help, PNG
	x3d\	Figure di help / animazioni
...		
	Milling\	Lavorazioni di fresatura
	Turning\	Lavorazione di tornitura
...		
	ita\\	Descrizione in italiano
	eng\	Descrizione in inglese

Nell'interfaccia HMI gli esempi si possono trovare nel settore operativo Messa in servizio:

Softkey orizzontale Dati di sistema → Dati HMI → Modelli → Esempi → Easyscreen → ... (vedere sopra).

Descrizione sintetica degli esempi

- Lavorazioni di fresatura
  - Esempio Foratura: qui è possibile abbinare una posizione
  - Esempio Misura pezzo: con il dialogo aperto, si può generare qui un richiamo di ciclo mediante NC-Start ed elaborarlo
  - Esempio Contapezzi: in questo esempio viene mostrata la gestione di GUD
- Lavorazione di tornitura
  - Attrezzature: queste comprendono una contropunta, un caricatore di barre, un raccoglitore pezzi
  - Esempio Misura pezzo: con il dialogo aperto, si può generare qui un richiamo di ciclo mediante NC-Start ed elaborarlo
  - Esempio Foratura: qui è possibile abbinare una posizione

Progettazione nel Sidescreen 13

Per la visualizzazione delle progettazioni Run MyScreens nel Sidescreen vi sono le seguenti possibilità:

1. Visualizzazione di una progettazione Run MyScreens in una pagina Sidescreen (separata):
   La progettazione Run MyScreens occupa l'intero spazio della pagina Sidescreen.
2. Visualizzazione di più progettazioni Run MyScreens in una Sidescreen Page (separata):
   Le singole progettazioni Run MyScreens vengono visualizzate sovrapposte nella Sidescreen Page, in cosiddetti elementi Sidescreen. Gli elementi Sidescreen e quindi anche le progettazioni Run MyScreens possono essere sfogliate in verticale.
3. Aggiunta di una o più progettazioni Run MyScreens a una Sidescreen Page esistente:
   Ciò significa che la progettazione Run MyScreens viene aggiunta a una Sidescreen Page contenuta della dotazione di fornitura SIEMENS. Questo caso coincide con il punto 2 tranne che per la pagine Sidescreen interessata.
4. Aggiunta di una o più progettazioni Run MyScreens a un elemento Sidescreen:
   Le singole progettazioni Run MyScreens vengono visualizzate affiancate in cosiddetti widget Sidescreen e possono essere sfogliate in orizzontale.

13.1 Visualizzazione di una progettazione Run MyScreens in una Sidescreen Page

Per la visualizzazione di una progettazione Run MyScreens in una Sidescreen Page (separata) occorre aggiungere una nuova Sidescreen Page alla configurazione Sidescreen esistente ("slsidescreeen.ini"). In questa Sidescreen Page viene quindi visualizzata o aggiunta la progettazione Run MyScreens.

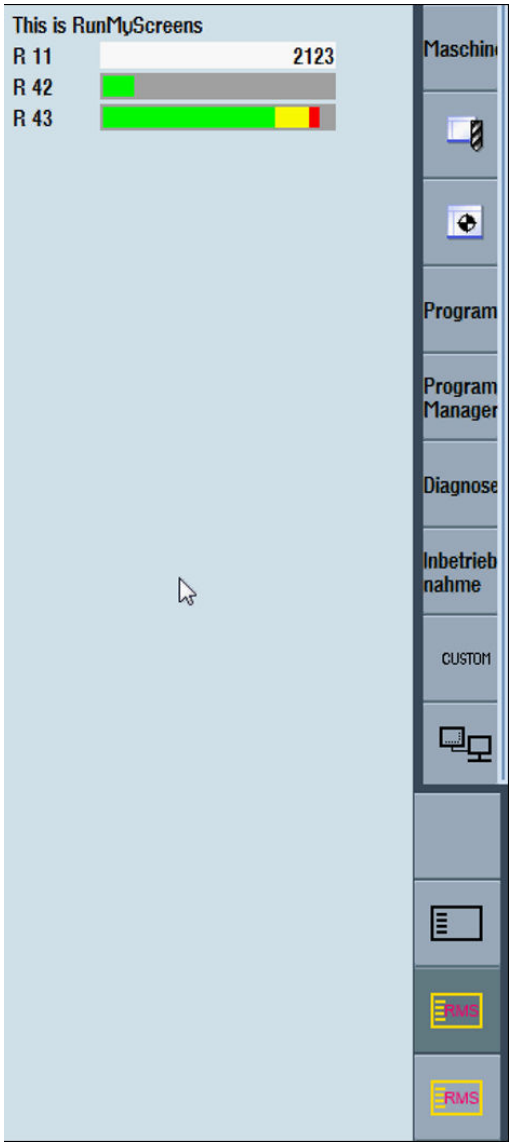


Figura 13-1 **Visualizzazione di una progettazione Run MyScreens in una Sidescreen Page**

Procedura

Aggiungere la seguente voce al file "slsidescreen.ini":

```
[Sidescreen]
PAGE100= name:=RMSPage,
implementation:=sleseasyscreen.SlEsSideScreenPage
```

- Il numero della Page, qui "100", va calcolato in base alle Sidescreen Page esistenti. Il numero della Page deve essere ≥ 100. Così si evitano possibili conflitti con le Pages SIEMENS.
- Il nome della Page, qui "RMSPage", si può scegliere liberamente.
- L'indicazione per l'implementazione della pagina, qui "sleseasyscreen.SlEsSideScreenPage", non deve essere modificata.

Nel passo successivo si configura la progettazione Run MyScreens da eseguire. A questo scopo creare una nuova sezione il cui nome contenga il nome della nuova pagina creata:

```
[Page_RMSPage]
Icon=rmspage.png
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
PROPERTY003= name:=focusable, type:=bool, value:="true"
```

- Il nome della sezione viene formato aggiungendo al prefisso "Page_" il nome della pagina, in questo caso "RMSPage".
- In Icon immettere il nome del file, in questo caso "rmspage.png". Il file contiene l'icona del pulsante per la selezione della pagina. Salvare il file nella directory /user/sinumerik/hmi/ico/ico640 o /oem/sinumerik/hmi/ico/ico640 ab.
- Immettere in Property maskPath il nome del file che contiene la progettazione Run MyScreens, in questo caso "sidescreenmask.com".
- Immettere nella Property maskName il nome della maschera Run MyScreens che deve essere visualizzata (qui "Mask").
- Nella Property focusable impostare l'area di impostazione attiva. Solo quando questo attributo ha il valore "true", la Page diventa attiva per le immissioni. Questo attributo è necessario per le Pages che contengono elementi in cui si devono immettere dei dati.

Salvare "slsidescreen.ini" nella directory /oem/sinumerik/hmi/cfg o /user/sinumerik/hmi/cfg.

Dopo il primo avvio dell'HMI la progettazione Run MyScreens è disponibile.

Esempio

Esempio di una configurazione completa in "slsidescreen.ini":

```
[Sidescreen]
PAGE005= name:=RMSPage,
implementation:=sleseasyscreen.SlEsSideScreenPage
[Page_RMSPage]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

13.2 Visualizzazione di più progettazioni Run MyScreens in una pagina Sidescreen

Per la visualizzazione di più progettazioni Run MyScreens in una pagina Sidescreen (separata), è necessario configurare, oltre alla pagina Sidescreen, i cosiddetti elementi del Sidescreen. Questo elemento serve alla visualizzazione delle progettazioni Run MyScreens.

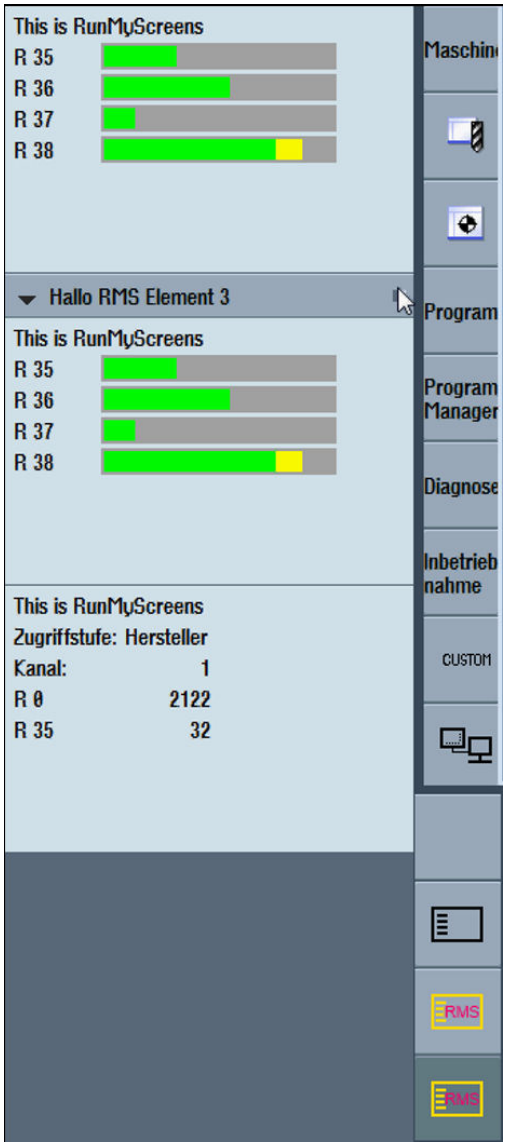


Figura 13-2 **Visualizzazione di più progettazioni Run MyScreens in una pagina Sidescreen**

Procedura

Aggiungere le seguenti voci al file "slsidescreen.ini":

```
[Sidescreen]
PAGE100= name:=RMSPage, implementation:=SlSideScreenPage
```

- Il numero della Page, qui "100", va calcolato in base alle Sidescreen Page esistenti. Il numero della Page deve essere ≥ 100. Così si evitano possibili conflitti con le Pages SIEMENS.
- Il nome della Page, qui "RMSPage", si può scegliere liberamente.
- L'indicazione per l'implementazione della Page, qui "SlSideScreenPage", non deve essere modificata.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement001,
implementation:=sleseasyscreen.SlEsSideScreenElement
ELEMENT002= name:=RMSElement002,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Alla Page sono assegnati due elementi Sidescreen: RMSElement001 e RMSElement002. I nomi degli elementi possono essere scelti liberamente.
- L'indicazione per l'implementazione degli elementi, qui "sleseasyscreen.SlEsSideScreenElement", non deve essere modificata.

Al termine occorre ancora assegnare agli elementi Sidescreen le progettazioni Run MyScreens da visualizzare in questi elementi:

```
[Element_RMSElement001]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask001"
…
[Element_RMSElement002]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask002"
```

Salvare "slsidescreen.ini" nella directory /oem/sinumerik/hmi/cfg o /user/sinumerik/hmi/cfg.

Dopo il primo avvio dell'HMI la progettazione Run MyScreens è disponibile.

13.3 Aggiunta di progettazioni Run MyScreens in una Sidescreen Page

La procedura per l'integrazione di una o più progettazioni Run MyScreens in una Sidescreen Page esistente è identica a quella descritta nel capitolo Visualizzazione di più progettazioni Run MyScreens in una pagina Sidescreen (Pagina 294). L'unica differenza consiste nel fatto che gli elementi Sidescreen per la visualizzazione delle progettazioni Run MyScreens vengono aggiunti a una Sidescreen Page già esistente.

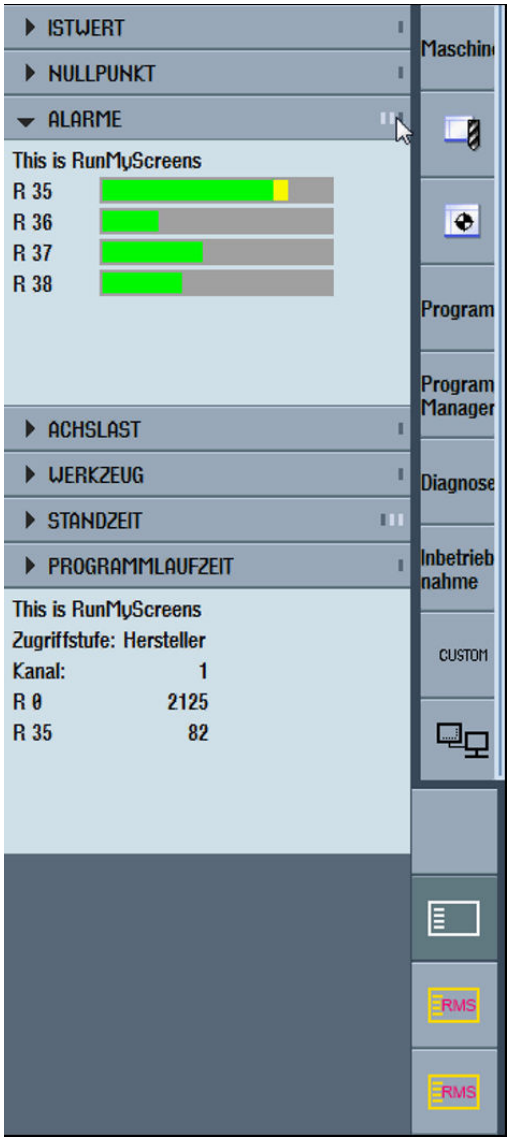


Figura 13-3 **Aggiunta di progettazioni Run MyScreens in una Sidescreen Page**

Nota

Di tutte Sidescreen Pages contenute nella dotazione di fornitura SIEMENS, solo la WidgetsPage (vedere il file "slsidescreen.ini") può essere ampliata con progettazioni Run MyScreens.

Procedura

Aggiungere le seguenti voci al file "slsidescreen.ini":

Per confermare la progettazione Run MyScreen aggiungere un elemento corrispondente nella sezione [Page_WidgetsPage].

```
[Page_WidgetsPage]
ELEMENT100= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Per i nuovi elementi, usare solo un intervallo numerico ≥ 100. Così si evitano possibili conflitti con le gli elementi di visualizzazione SIEMENS. L'elemento RMSElement viene inserito nella WidgetsPage in base alla numerazione sotto gli elementi SIEMENS.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask001"
```

13.4 Aggiunta di progettazioni Run MyScreens a un elemento Sidescreen

La base della procedura di esempio descritta di seguito è una pagina Sidescreen con un elemento Sidescreen. Nell'elemento Sidescreen vengono visualizzate due progettazioni Run MyScreens affiancate.

Configurare una Sidescreen Page con un elemento Sidescreen come descritto di seguito. Assegnare all'elemento Sidescreen due widget Sidescreen con le relative progettazioni Run MyScreens.

Procedura

Aggiungere le seguenti voci al file "slsidescreen.ini":

```
[Sidescreen]
PAGE100= name:=RMSPage, implementation:=SlSideScreenPage
```

- L'indicazione per l'implementazione, qui "SlSideScreenPage", non deve essere modificata.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement, implementation:= SlSideScreenElement
```

- L'indicazione per l'implementazione, qui "SlSideScreenPage", non deve essere modificata.

```
[Element_RMSElement]
TextId=RMS_ELEMENT_TITLE
TextFile=rmstexts
TextContext=rmstexts
TextContext=rmstexts
WIDGET001= name:=RMSWidget001,
implementation:=sleseasyscreen.SlEsSideScreenWidget
```

```
WIDGET002= name:=RMSWidget002,
implementation:=sleseasyscreen.SlEsSideScreenWidget
```

- In caso di utilizzo dell'implementazione SlSideScreenElement per l'elemento Sidescreen (vedere la sezione [Page_RMSPage]), l'elemento Sidescreen possiede una riga del titolo in cui può comparire un testo.
  Questo testo viene progettato con le voci TextId, TextFile e TextContext:
  - TextId: Id del testo da visualizzare
  - TextFile: File in cui è salvato il testo
  - TextContext: Contesto relativo al testo all'interno di questo file

  Per ogni lingua occorre creare un file separato.
  Il nome di questi file viene creato in base al seguente formato:
  <TextFile>_<suffisso_lingua>.ts, ad es. rmstexts_deu.ts per l'esempio precedente.
  Se non si specifica un file e un contesto (TextFile=<empty> e TextContext=<empty>), come testo viene visualizzato l'Id di testo. Come Id di testo si può specificare una stringa qualsiasi.
  Il file di testo (ad es. rmstexts_deu.ts) è strutturato come segue:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TS>
<TS>
 <context>
 <name>rmstexts</name>
 <message>
  <source>My Application</source>
 <translation>Mia applicazione</translation>
 </message>
 </context>
</TS>
```

  Salvare il file nella directory /oem/sinumerik/hmi/lng o /user/sinumerik/hmi/lng. Al successivo avvio dell'HMI il file viene convertito nel formato file richiesto al runtime.
- I nomi dei widget assegnati all'elemento, qui "RMSWidget001" e "RMSWidget002", possono essere scelti liberamente.
- L'implementazione del widget "sleseasysscreen.SlEsSideScreenWidget" non deve essere modificata.

```
[Widget_RMSWidget001]
PROPERTY001= name:=maskPath, type:=QString, value:="mymask001.com"
PROPERTY002= name:=maskName, type:=QString, value:="MyMask001"

[Widget_RMSWidget002]
PROPERTY001= name:=maskPath, type:=QString, value:="mymask002.com"
PROPERTY002= name:=maskName, type:=QString, value:="MyMask002"
```

- I nomi delle sezioni per la configurazione dei widget Sidescreen vengono creati aggiungendo al prefisso "Widget" il nome del widget che deve essere configurato in questa sezione.
- Immettere in Property maskPath il nome del file che contiene la progettazione Run MyScreens, in questo caso "mymask001.com" o "mymask002.com".
- Immettere nella Property maskName il nome della maschera Run MyScreens che deve essere visualizzata, in questo caso "MyMask001" o "MyMask002".

13.5 Avvertenze sull'esecuzione delle progettazioni Run MyScreens nel Display-Manager

Rispettare le seguenti avvertenze:

- Le funzioni LS, LM, DLGL, EXIT, EXITLS non sono disponibili nel Sidescreen durante l'esecuzione di progettazioni Run MyScreens.
- Non è possibile richiedere le dimensioni della maschera nel blocco LOAD.
- I testi sono sempre visualizzati con il font utilizzato in SINUMERIK Operate alla risoluzione di 640x480 pixel.
- Le posizioni progettate degli elementi di comando vengono convertite (ad es. estensione).
- Le dimensioni della finestra di dialogo vengono adattate automaticamente.
- Se le progettazioni Run MyScreens vengono eseguite in una Sidescreen Page, è disponibile l'intera superficie della Page per la visualizzazione della progettazione. In caso di esecuzione in elementi o widget Sidescreen, la superficie disponibile per la visualizzazione viene preimpostata dal sistema in funzione del contesto.
- L'intestazione e i softkey non sono progettabili.
- Tutti i messaggi di debug o di errore vengono scritti sequenzialmente nel file di testo "easyscreen_log.txt". Non esiste un identificativo specifico che indica da quale progettazione proviene un messaggio.
- Lo stato di una progettazione visualizzata nel Sidescreen viene annullato quando si disattiva la visualizzazione della progettazione.

Nota

L'integrazione di progettazioni Run MyScreens nel Sidescreen fa sì che venga eseguita contemporaneamente più di una progettazione Run MyScreens. Per preservare la funzionalità dell'intero sistema, fare attenzione – a seconda dell'hardware impiegato – al risultante carico di sistema aggiuntivo.

Progettazione in Display Manager 14

14.1 Visualizzazione delle progettazioni Run MyScreens nel Sidescreen

Per la visualizzazione di progettazioni Run MyScreens in Display Manager è disponibile la finestra di dialogo SISideScreenDialog. A seconda dello spazio disponibile questa finestra di dialogo può visualizzare una o più Sidescreen Pages (vedere IM9 capitolo 3.13). Più Pages 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 Page ha un proprio file di configurazione. I nomi di questi file di configurazione vengono specificati durante la progettazione della finestra di dialogo nel file "ystemconfiguration.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.

Esempio

```
[dialogs]
DLG500= name:=RMSApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false, cmdline:="-sidescreen1
rmspage1.ini –sidescreen2 rmspage2.ini –spacing 3"
```

L'istanza della finestra di dialogo SlSideScreenDialog creata in questo esempio ha il nome RMSApp. Comprende 2 Pages/colonne. Le due Pages/colonne vengono configurate nei file "rmspage1.ini" e "rmspage2.ini". Tra le due colonne vi è una distanza di 3 pixel.

Nota

I nomi dei file con le configurazioni delle Pages, qui "rmspage1.ini" e "rmspage2.ini", possono essere scelti liberamente.

L'uso di progettazioni Run MyScreens in Display Manager avviene come descritto nel capitolo Progettazione nel Sidescreen (Pagina 291). L'unica differenza consiste nel fatto che l'integrazione delle progettazioni non avviene nel file "slsidescreen.ini", bensì nei file previsti per la configurazione delle Pages esistenti.

Per l'integrazione di progettazioni Run MyScreens in Display Manager di SINUMERIK Operate sono disponibili 2 varianti. Queste vengono descritte nei capitoli seguenti.

14.2 Integrazione in SISideScreenDialog

Per l'Integrazione della progettazione Run MyScreens occorre creare l'istanza della finestra di dialogo SlSideScreenDialog nel file "systemconfiguration.ini" e creare il file per l'integrazione della progettazione Run MyScreens.

L'esempio che segue mostra l'integrazione di una progettazione Run MyScreens. La progettazione Run MyScreens occupa l'intera area della finestra di dialogo SlSideScreenDialog, ossia è presente una sola Page/colonna.

Procedura

Dapprima creare un'istanza della finestra di dialogo SlSideScreenDialog nel file "systemconfiguration.ini" nella sezione `[dialogs]`. L'istanza viene chiamata "RMSApp".

```
[dialogs]
DLG500= name:=RMSApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false,
cmdline:="-sidescreen1 rmsapp.ini"
```

Quindi configurare o integrare la progettazione Run MyScreens nel file "rmsapp.ini":

```
[Sidescreen]
PAGE001= name:=RMSPage, implementation:=SlSideScreenPage
```

- Creare una Page con il nome "RMSPage".

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Sulla Page RMSPage creare un elemento con il nome RMSElement. Questo elemento serve alla visualizzazione della progettazione Run MyScreens.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

- Immettere in Property maskPath il nome del file che contiene la progettazione Run MyScreens, in questo caso "sidescreenmask.com".
- Immettere nella Property maskName il nome della maschera Run MyScreens che deve essere visualizzata (qui "Mask").

Creare i file "rmsapp.ini" e "systemconfiguration.ini" (vedere sopra) nella directory /oem/sinumerik/hmi/cfg o /user/sinumerik/hmi/cfg ab.

Dopo il primo avvio dell'HMI la progettazione Run MyScreens è disponibile.

14.3 Integrazione in SIWidgetsApp

Di seguito viene descritta l'integrazione di una progettazione Run MyScreens nell'applicazione SlWidgetsApp contenuta nella dotazione di fornitura SIEMENS.

A seconda della Page/colonna dell'applicazione SlWidgetsApp in cui la progettazione Run MyScreens deve essere integrata, occorre espandere il file "sldmwidgets1.ini" o il file "sldmwidgets2.ini".

L'esempio che segue mostra l'integrazione di una progettazione Run MyScreens nella Page/ colonna sinistra dell'applicazione SIWidgetsApp.

Procedura

Espandere il file "sldmwidgets1.ini" come segue:

```
[Sidescreen]
PAGE001= name:=WidgetsPage, implementation:=SlSideScreenPage
[Page_WidgetsPage]
ELEMENT100= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Creare un elemento con il nome "RMSElement" per la visualizzazione della progettazione Run MyScreens.

Nota

Per i nuovi elementi, usare solo un intervallo numerico ≥ 100. Così si evitano possibili conflitti con le gli elementi di visualizzazione SIEMENS. L'elemento RMSElement viene inserito nella WidgetsPage in base alla numerazione sotto gli elementi SIEMENS.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

- Immettere in Property maskPath il nome del file che contiene la progettazione Run MyScreens, in questo caso "sidescreenmask.com".
- Immettere nella Property maskName il nome della maschera Run MyScreens che deve essere visualizzata, in questo caso "Mask".

Salvare il file "sldmwidgets1.ini" nella directory /oem/sinumerik/hmi/cfg o /user/sinumerik/hmi/cfg.

Dopo il primo avvio dell'HMI la progettazione Run MyScreens è disponibile.

14.4 Avvertenze sull'esecuzione delle progettazioni Run MyScreens in Display Manager

Rispettare le seguenti avvertenze:

- Le funzioni LS, LM, DLGL, EXIT, EXITLS non sono disponibili in Display Manager durante l'esecuzione di progettazioni Run MyScreens.
- Non è possibile richiedere le dimensioni della maschera nel blocco LOAD.
- I testi sono sempre visualizzati con il font utilizzato in SINUMERIK Operate alla risoluzione di 640x480 pixel.
- Le posizioni progettate degli elementi di comando vengono convertite (ad es. estensione).

- Se le progettazioni Run MyScreens vengono eseguite in una Sidescreen Page, è disponibile l'intera superficie della Page per la visualizzazione della progettazione. In caso di esecuzione in elementi o widget Sidescreen, la superficie disponibile per la visualizzazione viene preimpostata dal sistema in funzione del contesto.
- Tutti i messaggi di debug o di errore vengono scritti sequenzialmente nel file di testo easyscreen_log.txt. Non esiste un identificativo specifico che indica da quale progettazione proviene un messaggio.
- Lo stato di una progettazione visualizzata in Display Manager viene annullato quando si disattiva la visualizzazione della progettazione.

Nota

L'integrazione di progettazioni Run MyScreens in Display Manager fa sì che venga eseguita contemporaneamente più di una progettazione Run MyScreens. Per preservare la funzionalità dell'intero sistema, fare attenzione – a seconda dell'hardware impiegato – al risultante carico di sistema aggiuntivo.

Liste di riferimento

A

A.1 Elenchi dei softkey di accesso

A.1.1 Elenco dei softkey di accesso per Tornitura

Settore operativo Programma per Tornitura

HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK7	HSK8
Edit	**Foratura**	**Tornitura**	**Tornitura del profilo**	**Fresatura**	**Varie**	**Simulazione**	**Selezione NC**
HSK9	HSK10	HSK11	HSK12	HSK13	HSK14	HSK15	HSK16
--	Retta Cerchio (solo in Shop-Turn)	--	--	Misura pezzo	Misura utensile	**OEM**	--

Tornitura

Nelle seguenti tabelle vengono rappresentati i possibili softkey di accesso della tecnologia Tornitura. Le assegnazioni dei singoli softkey di accesso possono variare da sistema a sistema. I softkey OEM indicati sono consentiti per "Run MyScreens".

Softkey di accesso programGUIDE (codice G):

	Foratura	**Tornitura**	**Tornitura del profilo**		**Fresatura**		**Varie**	
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**	
VSK1	Centratura	Sgrossatura	Profilo	--	Fresatura a spianare	Profilo	Impostazioni	High Speed Settings
VSK2	Foratura alesatura	Gola	Sgrossatura	--	Tasca	Interpolazione	Orientamento piano	Assi paralleli
VSK3	Foratura profonda	Scarico	Sgrossat. mat. res.	--	Perno poliedrico	Preforatura	Orientamento utensile	--
VSK4	Alesatura	Filetto	Troncatura	--	Cava	Tasca	--	--
VSK5	Filetto	Troncatura	Troncat. mat. res.	--	Fresatura di filetti	Mat. res. tasca	--	--
VSK6	**OEM**	--	Tornitura con troncatura	--	Incisione	Perno	Sottoprogramma	--
VSK7	Posizioni	**OEM**	Torn.tron mat. res.	**OEM**	**OEM**	Mat. res. perno	--	**OEM**
VSK8	Ripetere posizione	--	>>	<<	Fresatura del profilo	<<	>>	<<

Softkey di accesso per ShopTurn:

	Foratura	**Tornitura**	**Tornitura del profilo**		**Fresatura**		**Varie**		
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**		**HSK10**
VSK1	Foratura in asse	Sgrossatura	Nuovo profilo	--	Fresatura a spianare	Nuovo profilo	Impostazioni	High Speed Settings	Attrezzo
VSK2	Centratura	Gola	Sgrossatura	--	Tasca	Interpolazione	Orientamento piano	Assi paralleli	Retta
VSK3	Foratura alesatura	Scarico	Sgrossat. mat. res.	--	Perno poliedrico	Preforatura	Orientamento utensile	Ripetiz. programma	Centro del cerchio
VSK4	Foratura profonda	Filetto	Troncatura	--	Cava	Tasca	Contromandrino	--	Raggio cerchio
VSK5	Filetto	Troncatura	Troncat. mat. res.	--	Fresatura di filetti	Mat. res. tasca	Trasformazioni	--	Polari
VSK6	**OEM**	--	Tornitura con troncatura	--	Incisione	Perno	Sottoprogramma	--	Distacco/ accostam.
VSK7	Posizioni	**OEM**	Torn.tron mat. res.	**OEM**	**OEM**	Mat. res. perno	--	**OEM**	--
VSK8	Ripetere posizione	--	>>	<<	Fresatura del profilo	<<	>>	<<	--

A.1.2 Elenco dei softkey di accesso per Fresatura

Settore operativo Programma per Fresatura

HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK7	HSK8
Edit	**Foratura**	**Fresatura**	**Fresatura del profilo**	**Tornitura** (solo con codice G)	**Varie**	**Simulazione**	**Selezione NC**
HSK9	HSK10	HSK11	HSK12	HSK13	HSK14	HSK15	HSK16
--	Retta Cerchio (solo in ShopMill)	--	--	Misura pezzo	Misura utensile	**OEM**	--

Fresatura

Nelle seguenti tabelle vengono rappresentati i possibili softkey di accesso della tecnologia Fresatura. Le assegnazioni dei singoli softkey di accesso possono variare da sistema a sistema. I softkey OEM indicati sono consentiti per "Run MyScreens".

Softkey di accesso programGUIDE (codice G):

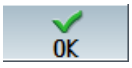
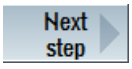
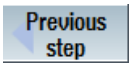
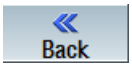
	Foratura	**Fresatura**	**Fresatura del profilo**		**Tornitura**		**Varie**	
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**	
VSK1	Centratura	Fresatura a spianare	Profilo	--	Sgrossatura	Profilo	Impostazioni	--
VSK2	Foratura alesatura	Tasca	Interpolazione	--	Gola	Sgrossatura	Orientamento piano	Assi paralleli
VSK3	Foratura profonda	Perno poliedrico	Preforatura	--	Scarico	Sgrossat. mat. res.	Orientamento utensile	--
VSK4	Alesatura	Cava	Tasca	--	Filetto	Troncatura	High Speed Settings	--
VSK5	Filetto	Fresatura di filetti	Mat. res. tasca	--	Troncatura	Troncat. mat. res.	--	--
VSK6	**OEM**	Incisione	Perno	--	--	Tornitura con troncatura	Sottoprogramma	--
VSK7	Posizioni	**OEM**	Mat. res. perno	**OEM**	**OEM**	Torn.tron mat. res.	--	**OEM**
VSK8	Ripetere posizione	--	>>	<<	Tornitura del profilo	<<	>>	<<

Softkey di accesso per ShopMill:

	Foratura	**Fresatura**	**Fresatura del profilo**		**Varie**		**Retta cerchio**
	HSK2	**HSK3**	**HSK4**		**HSK6**		**HSK10**
VSK1	Centratura	Fresatura a spianare	Nuovo profilo	--	Impostazioni	--	Attrezzo
VSK2	Foratura alesatura	Tasca	Interpolazione	--	Orientamento piano	Assi paralleli	Retta
VSK3	Foratura profonda	Perno poliedrico	Preforatura	--	Orientamento utensile	Ripetiz. programma	Centro del cerchio
VSK4	Alesatura	Cava	Tasca	--	High Speed Settings	--	Raggio cerchio
VSK5	Filetto	Fresatura di filetti	Mat. res. tasca	--	Trasformazioni	--	Elicoide
VSK6	**OEM**	Incisione	Perno	--	Sottoprogramma	--	Polari
VSK7	Posizioni	**OEM**	Mat. res. perno	**OEM**	--	**OEM**	--
VSK8	Ripetere posizione	--	>>	<<	>>	<<	--

A.2 Elenco dei softkey predefiniti

Tabella A-1 Softkey predefiniti

Nome	Softkey
SOFTKEY_OK	
SOFTKEY_CANCEL	
SOFTKEY_APPLY	
SOFTKEY_MORE	
SOFTKEY_BACK	
SOFTKEY_ASSISTANT_NEXT	
SOFTKEY_ASSISTANT_PREVIOUS	
SOFTKEY_NAV_BACK	

A.3 Lista dei livelli di accesso

I diversi livelli di accesso hanno il seguente significato:

Grado di protezione	Interblocco con	Campo
ac0	riservati a Siemens	
ac1	Password	Costruttore della macchina
ac2	Password	Service
ac3	Password	Utente
ac4	Interruttore a chiave posizione 3	Programmatore attrezzista
ac5	Interruttore a chiave posizione 2	Operatore qualificato
ac6	Interruttore a chiave posizione 1	Operatore addestrato
ac7	Interruttore a chiave posizione 0	Operatore istruito

A.4 Elenco dei colori

Colori di sistema

Per la progettazione delle finestre di dialogo è disponibile una tabella dei colori unitaria (sottoinsieme dei relativi colori standard). Per un elemento (testo, campo di immissione, sfondo, ecc.) è possibile selezionare uno dei seguenti colori compresi tra 0 e 133.

In alternativa ai colori predefiniti è possibile specificare i colori anche come valori RGB ("#RRGGBB").

Indice	Pittogramma	Colore	Descrizione del colore
1		nero	
2		arancione	
3		verde scuro	
4		grigio chiaro	
5		grigio scuro	
6		blu	
7		rosso	
8		marrone	
9		giallo	
10		bianco	
126		nero	Colore del testo di un campo di I/O attualmente selezionato
127		arancione chiaro	Colore di sfondo di un campo di I/O attualmente selezionato
128		arancione	Colore di sistema campo attivo
129		grigio chiaro	Colore di sfondo
130		blu	Colore intestazione (attivo)
131		nero	Colore testo intestazione (attivo)

Indice	Pittogramma	Colore	Descrizione del colore
132		turchese	Colore di sfondo di un campo di toggle
133		azzurro	Colore di sfondo della casella di riepilogo

A.5 Lista degli identificativi delle lingue nel nome file

Lingue supportate

Lingue standard:

Lingua	Abbreviazione nel nome file
Cinese semplificato	chs
Tedesco	deu
Inglese	eng
Francese	fra
italiano	ita
Spagnolo	esp

Altre lingue:

Lingua	Abbreviazione nel nome file
Cinese tradizionale	cht
Danese	dan
Finlandese	fin
Indonesiano	ind
Giapponese	jpn
Coreano	kor
Malese	msl
Olandese	nld
Polacco	plk
Portoghese (brasiliano)	ptb
Rumeno	rom
Russo	rus
Svedese	sve
Slovacco	sky
Sloveno	slv
Tailandese	tha
Ceco	csy
Turco	trk

Lingua	Abbreviazione nel nome file
Ungherese	hun
Vietnamita	vit

A.6 Elenco delle variabili di sistema accessibili

Bibliografia

Libretto di descrizione parametri Variabili di sistema, /PGAsl/

A.7 Comportamento all'apertura della finestra di dialogo (attributo CB)

La seguente tabella fornisce una rappresentazione generale di come avviene il richiamo del metodo CHANGE.

Per l'attributo CB vale:

CB0

Il metodo CHANGE viene avviato con la visualizzazione della maschera se la variabile ha istantaneamente un valore valido (ad es. mediante preassegnazione o variabile NC/PLC).

CB1 (default)

Il metodo CHANGE non viene avviato esplicitamente con la visualizzazione della maschera. Anche se la variabile ha una variabile NC/PLC progettata viene naturalmente richiamato il metodo CHANGE.

Condizione				Reazione
Tipo	**Variabile di sistema o utente**	**Preassegnazione**	**Esecuzione del metodo Change**	
Campo I/O	Sì	Sì	Sì	Grazie alla variabile NC/PLC progettata risulta sempre almeno un richiamo automatico del metodo CHANGE con il valore attuale della variabile NC/PLC. La preassegnazione di CB non ha alcun effetto.
			No	
		No	Sì	
			No	
	No	Sì	Sì	Il metodo CHANGE viene richiamato con il valore della preassegnazione.
			No	Il metodo CHANGE non viene richiamato.
		No	Sì	Nessun richiamo perché non è presente alcun valore valido per richiamare un metodo CHANGE.
			No	

Condizione				**Reazione**
Tipo	**Variabile di sistema o utente**	**Preassegnazione**	**Esecuzione del metodo Change**	
Toggle	Sì	Sì	Sì	Grazie alla variabile NC/PLC progettata risulta sempre almeno un richiamo automatico del metodo CHANGE con il valore attuale della variabile NC/PLC. La preassegnazione di CB non ha alcun effetto.
			No	
		No	Sì	
			No	
	No	Sì	Sì	Il metodo CHANGE viene richiamato con il valore della preassegnazione.
			No	Il metodo CHANGE non viene richiamato.
		No (per impostazione predefinita, la preassegnazione avviene con il primo valore della lista di toggle)	Sì	Il metodo CHANGE viene richiamato come preassegnazione con il primo valore della lista di toggle.
			No	Il metodo CHANGE non viene richiamato.

Suggerimenti utili

B

B.1 Suggerimenti di carattere generale

- Utilizzare possibilmente la variante BTSS per le variabili di sistema (variabili $).
  Motivo:
  Così facendo si evita in certi casi la dispendiosa procedura di risoluzione dei nomi.
  Esempio:
  Anziché "$R[10]" è meglio usare "/Channel/Parameter/R[u1,10]".
- Evitare l'impostazione ciclica (equivalente), ad es. della proprietà HLP di maschere e variabili. Confrontare il valore da immettere in precedenza con il valore corrente e impostarlo solo in caso che i due differiscano.
  Motivo:
  Evitare la complessa ricerca delle pagine di help dipendenti dalla risoluzione.
  Esempio:

```
DEF MyVar=(R3///,"X1",,"mm"/WR1//"$AA_IM[0]")
CHANGE(MyVar)
   IF MyVar.VAL < 100
      HLP="mypic1.png"
   ELSE
      HLP="mypic2.png"
   ENDIF
END_CHANGE
```

Raccomandazione:

```
CHANGE(MyVar)
   IF MyVar.VAL < 100
      IF HLP <> "mypic1.png"
         HLP="mypic1.png"
      ENDIF
   ELSE
      IF HLP <> "smypic2.png"
         HLP="mypic2.png"
      ENDIF
   ENDIF
END_CHANGE
```

- Sostituire più funzioni RNP() successive con una funzione MRNP().
  Sostituire più funzioni RDOP() successive con una funzione MRDOP().
  Motivo:
  Riduzione del carico di comunicazione e incremento delle prestazioni.
- Leggere i parametri dell'azionamento con un clock non più veloce di 1 secondo, se possibile anche più lento.
  Motivo: La comunicazione con gli azionamenti potrebbe risultarne molto disturbata o potrebbe addirittura interrompersi.
- Evitare i calcoli in successione con variabili di dialogo collegate a variabili di sistema o variabili utente. Utilizzare a tal fine Register (REG[x]) oppure variabili ausiliarie (invisibili).
  Motivo: Ogni assegnazione di un valore causa anche una registrazione nella variabile di sistema o utente collegata.
- Il codice di uno stesso tipo che viene utilizzato in blocchi diversi andrebbe riunito in un sottoblocco per motivi di leggibilità, facilità di manutenzione e prestazioni (all'apertura della maschera). Questo sottoblocco si potrà poi richiamare dove necessario con la funzione CALL().
- Monitorando le risorse della CPU nella riga di dialogo (impostazione in slguiconfig.xml) si può controllare in che misura determinate modifiche della maschera incidono sulle prestazioni.

B.2 Suggerimenti per il debugging

- Uso della funzione DEBUG() per scopi diagnostici. Quando è richiamata la funzione DEBUG(), la stringa trasmessa viene scritta nel file "easyscreen_log.txt". Anche l'output delle informazioni nella riga di dialogo tramite la funzione DLGL() può essere utile.
  Sempre per ragioni di prestazioni, una volta ultimato lo sviluppo delle maschere questa funzione andrebbe però rimossa o esclusa come commento.
- Il file di registro "easyscreen_log.txt" non dovrebbe mai contenere elementi al termine dello sviluppo di una maschera.

B.3 Suggerimenti per il metodo CHANGE

- I metodi CHANGE vanno mantenuti molto brevi e ridotti, soprattutto per quelli le cui variabili sono collegate a una variabile di sistema o utente e che variano con elevata frequenza.
  Motivo:
  Incremento delle prestazioni della maschera.
- Se possibile, evitare di progettare le funzioni RNP() nel metodo CHANGE. Conviene invece creare parallelamente una variabile invisibile con la variabile di sistema o utente da leggere e quindi utilizzarla.
  Motivo:
  Ad ogni richiamo verrebbe necessariamente emessa una funzione RNP(). Nell'altro caso si accederebbe solo al valore attuale comunque già presente.

Esempio:

Ad ogni modifica del movimento dell'asse tramite RNP() si attiva una procedura di risoluzione dei nomi per leggere un dato macchina specifico del canale:

```
DEF AXIS_POSITION_X =  (R///,""///"$AA_IM[X]")

CHANGE(AXIS_POSITION_X)
      DLGL("Axis """ << RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]") << """
has moved: "
<< AXIS_POSITION_X)
END_CHANGE
```

Con l'aiuto di una variabile invisibile si può tenere aggiornato il dato macchina specifico del canale, ricopiando ogni variazione dei valori in una variabile temporanea. come ad es. Register.

Questa variabile temporanea si può poi usare nel metodo CHANGE di modifica del valore della posizione dell'asse senza dover ogni volta risolvere il nome del dato macchina ed effettuare un accesso in lettura:

```
DEF AXIS_POSITION_X =  (R///,""///"$AA_IM[X]")
DEF AXIS_NAME_X = (S///,""/WR0//"$MC_AXCONF_GEOAX_NAME_TAB[0]")

CHANGE(AXIS_NAME_X)
      REG[0] = AXIS_NAME_X
END_CHANGE

CHANGE(AXIS_POSITION_X1)
   DLGL("Axis """ << REG[0] << """ has moved " << AXIS_POSITION_X)
END_CHANGE
```

- La frequenza di aggiornamento, e quindi l'esecuzione del relativo metodo CHANGE delle variabili collegate alle variabili di sistema o utente con alta frequenza di modifica del valore, si può ridurre sfruttando la proprietà UR della variabile, ad es. una variabile accoppiata ai valori dell'asse.
  Motivo:
  In questo modo, in caso di modifica del valore, il relativo metodo CHANGE viene eseguito in un arco di tempo prefissato.

B.4 Suggerimenti per i loop DO-LOOP

Poiché, a seconda della progettazione, i loop possono pregiudicare le prestazioni di SINUMERIK Operate, gli stessi vanno adottati con cautela e possibilmente rinunciando ad azioni che richiedano molto tempo.

Come variabile di esecuzione, ad esempio, si raccomanda l'impiego di un registro (REG[]) poiché anche le variabili di visualizzazione normali (in particolare quelle con collegamento BTSS) possono influire sulle prestazioni per effetto degli aggiornamenti o delle operazioni di scrittura estremamente frequenti.

Tenere presente che il numero di righe di script attualmente elaborabili "al pezzo" è limitato a 10.000. Una volta raggiunto questo numero, l'elaborazione dello script viene interrotta con una registrazione corrispondente nel file "easyscreen_log.txt".

Motivi:

- Impedire i loop infiniti
- "Run MyScreens" deve restare utilizzabile

Elementi animati

C

C.1 Premessa

Introduzione

Il manuale descrive come si lavora con X3D Viewer, che permette di integrare scene grafiche fisse e in movimento (elementi animati) – di seguito denominati figure di help – nell'interfaccia grafica di SINUMERIK Operate a partire dalla versione V4.7 SP1.

È possibile rappresentare in modo mirato procedure e parametri in figure di help contestuali. Si può così migliorare l'operabilità delle applicazioni e realizzare l'interfaccia utente in modo più attraente.

La realizzazione della figura di help finita prevede le operazioni seguenti:

- Creazione di elementi grafici e modelli 3D per le figure di help che verranno visualizzate in X3D Viewer (vedere il capitolo Modellazione (Pagina 318)).
- Creazione del file di descrizione scene in cui i dati del modello del file grafico vengono assegnati in modo mirato alle scene e animazioni da rappresentare (vedere il capitolo Struttura del file di descrizione scene (Pagina 325)).
- Conversione dei file X3D e XML in file HMI (vedere il capitolo Conversione in file hmi (Pagina 329)).
- Integrazione C++ di X3D Viewer nella propria applicazione (vedere il capitoloEsempio di implementazione (Pagina 331)).
- Applicazione in Run MyScreens (vedere il capitolo Visualizzazione in Run MyScreens (Pagina 332)).

Riepilogo dei formati dati

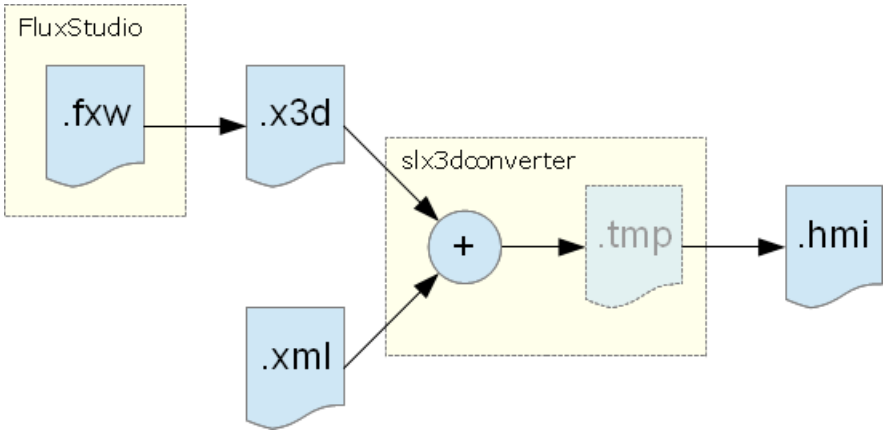


Figura C-1 Formati dati

Formato file	Descrizione
.fxw	File di grafica di Vivaty Studio
.x3d	File di modello in formato di scambio
.xml	File di definizione per animazioni e scene
.hmi	File di output per X3D Viewer

C.2 Modellazione

C.2.1 Presupposti

Lo scopo della modellazione è quello di creare un file con i modelli 3D creati nel formato dati .x3d, uno standard ufficiale per contenuti 3D.

La modellazione avviene in **Vivaty Studio**. Vivaty Studio è un tool di modellazione 3D disponibile gratuitamente con svariate possibilità di esportazione e importazione.

Presupposti

- È installato Vivaty Studio, versione 1.0.
- Si ha familiarità con la modellazione e con l'uso di Vivaty Studio.
- Il formato dati .x3d non viene descritto ulteriormente in questa sede, si presuppone che l'utente disponga delle conoscenze necessarie.

Nota

Vivaty Studio si può scaricare in Internet da questo link (https://www.web3d.org/projects/vivaty-studio).

C.2.2 Regole per la modellazione

Nella modellazione vanno rispettate le regole descritte di seguito. Il rispetto di queste regole è necessario per la conseguente preparazione del file delle figure di help.

Regole per la modellazione

1. TimeSensor deve avere il nome "Animation".
2. Tutti gli elementi correlati vanno assegnati a un gruppo.
3. Tutti i gruppi devono essere impostati alla posizione seguente:
   x = 0, y = 0, z = 0

4. Per non visualizzare i gruppi, i valori di traslazione vengono impostati nel seguente modo: x = 1000000, y = 1000000 und z = 1000000
5. Gli elementi seguenti devono avere lo stesso nome nei modelli grafici Vivaty Studio e nelle definizioni scene XML (cfr. il capitolo Panoramica (Pagina 324)):
   - Utensile
   - Fotocamere
6. Per creare un file .x3d, occorre effettuare l'impostazione seguente nella finestra di dialogo "Export Options":

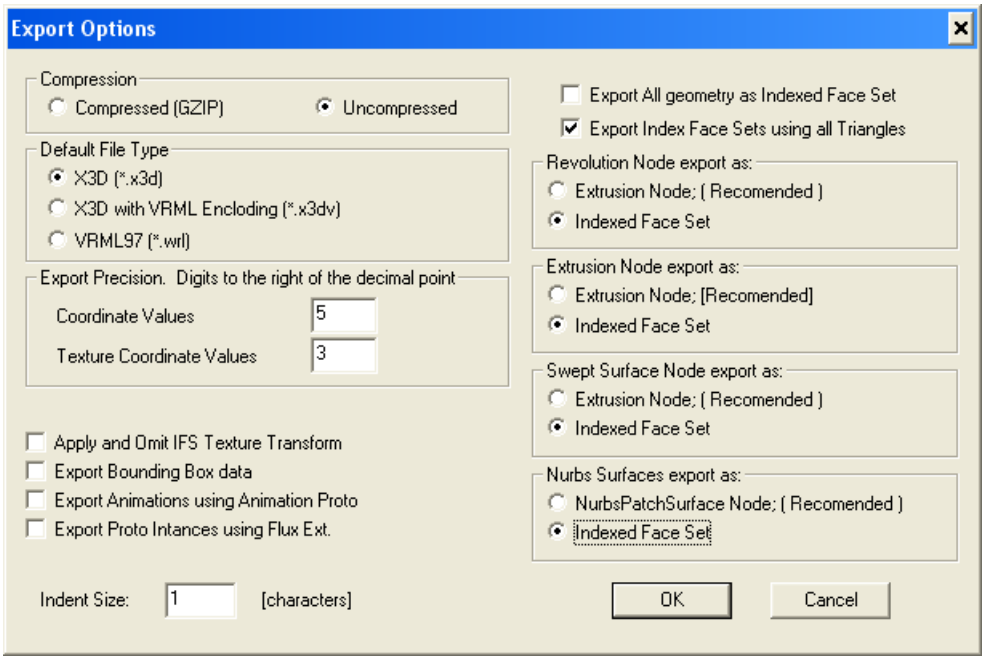


Figura C-2 Impostazioni nella finestra di dialogo Export Options

7. Per i testi si consigliano le impostazioni seguenti (vedere anche la finestra di dialogo seguente):
   - I testi devono essere sempre centrati.
   - La dimensione del testo deve essere 0,2. In questo modo è possibile posizionare correttamente il testo. L'output del testo in X3D Viewer avviene indipendentemente da questo valore nelle dimensioni carattere dell'interfaccia utente.

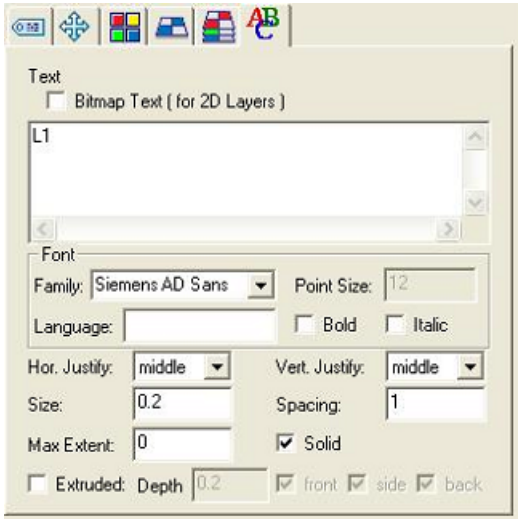


Figura C-3 Impostazioni per il testo

8. Per creare un tratteggio utilizzare il file schnitt.png come trama. Le linee vanno sempre da sinistra verso il basso e da destra verso l'alto in un angolo di 45°. La scalatura va impostata a 3 in entrambe le dimensioni. La rotazione dipende dalla forma costruttiva e deve essere adattata manualmente.

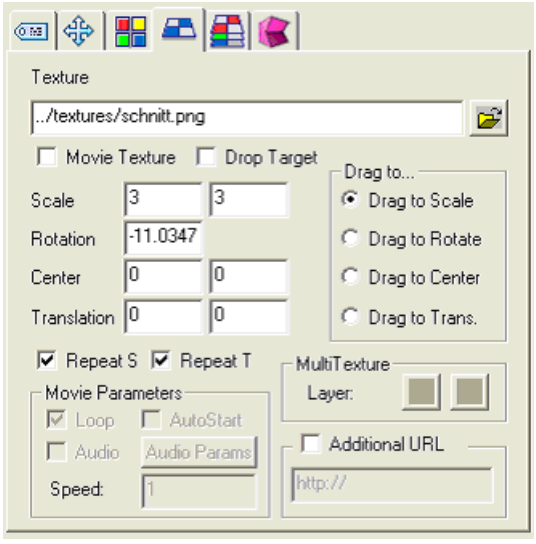


Figura C-4 Impostazioni per tratteggio

9. Per i pezzi grezzi consigliamo le seguenti dimensioni/misure:
   - Cilindri per la tornitura:
     Lunghezza = 5,5 ; raggio = 1,4 (cfr. modello turning_blank.fxw)
   - Parallelepipedo per fresatura:
     x = 4,75 ; y = 3 ; z = 3 (cfr. modello milling_blank.fxw)

Vedere anche

Comandi XML (Pagina 324)

C.2.3 Importazione di grafici (modelli)

In Flux Studio si possono importare grafici esterni (modelli). I modelli utilizzati di frequente, come ad esempio gli utensili, possono essere memorizzati a livello centrale in formato .x3d o .hmi e riutilizzati come elementi finiti in altre figure di help. Un elenco dei modelli di modellazione è riportato nel capitolo Modelli per la modellazione (Pagina 323).

Gli oggetti complessi possono essere creati e quindi importati anche con altri tool di modellazione, ad esempio Cinema4D.

Qui di seguito viene spiegato come si possono riutilizzare questi modelli.

Importazione come Inline

Per l'importazione di file .x3d, Flux Studio propone l'oggetto "Inline". Gli elementi esterni possono essere così integrati senza moltiplicare i dati 3D di questi modelli.

L'oggetto viene inserito selezionando la voce di menu **Create** -> **Create Inline**. Nelle proprietà dell'oggetto si immette quindi il nome file.

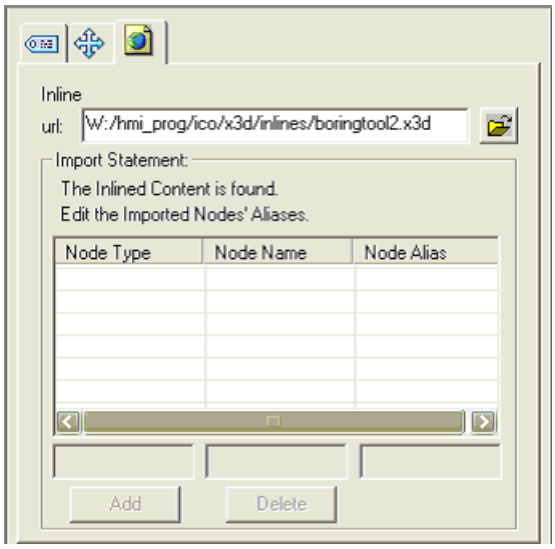


Figura C-5 Importazione tramite l'oggetto "Inline"

Importazione di dati modello 3D

Per l'importazione di dati modello da un file procedere come segue.

1. Selezionando la voce di menu **File** -> **Import Other Format** si apre la finestra di dialogo "Flux Studio Accutrans3D Translation Utility".
2. Selezionare il formato corrispondente dal campo di selezione "File Types". Ad esempio, per importare un file Cinema4D, occorre selezionare il tipo di file "3D Studio".
3. Selezionare quindi il file desiderato tramite **Select Files** e avviare l'importazione.

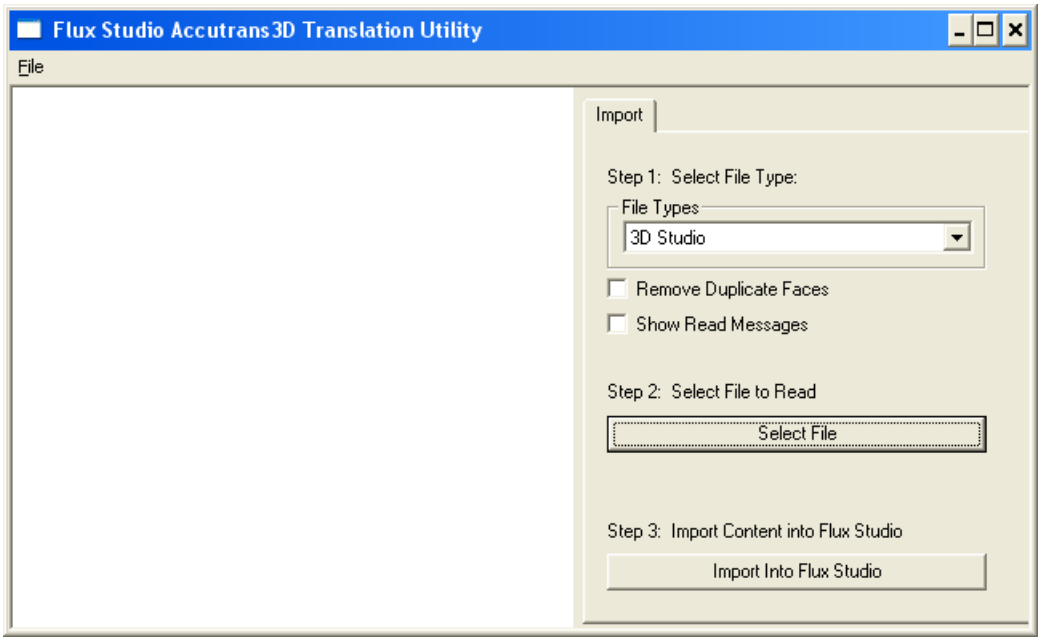


Figura C-6 Importazione di dati modello 3D

I dati modello del file Cinema4D vengono inseriti come gruppo completo. Tutte le fotocamere e le altre informazioni che vengono importate devono essere eliminate. Sono necessari solo gli elementi grafici del file Cinema4D.

C.2.4 Modelli per la modellazione

Questo capitolo contiene un elenco di elementi grafici che servono come base per la raffigurazione, i colori e le dimensioni. Affinché l'interfaccia utente presenti un "look and feel" omogeneo, si consiglia di utilizzare lo stile dei modelli o i modelli stessi per creare i propri grafici.

Informazioni generali

Modello	Descrizione
intersection_texture.png	Trama per una sezione
rapid_traverse_line_hori.fxw	Linea orizzontale per illustrare meglio il percorso dell'utensile in rapido
rapid_traverse_line_vert.fxw	Linea verticale per illustrare meglio il percorso dell'utensile in rapido

Modello	Descrizione
feed_traverse_line_hori.fxw	Linea orizzontale per illustrare meglio il percorso dell'utensile nell'avanzamento
feed_traverse_line_vert.fxw	Linea verticale per illustrare meglio il percorso dell'utensile nell'avanzamento
dimensioning_lines_hori.fxw	Linee di dimensionamento orizzontali
dimensioning_lines_vert.fxw	Linee di dimensionamento verticali
z1_inc.fxw	Linea di quota orizzontale con l'identificatore Z1
x1_inc.fxw	Linea di quota verticale con l'identificatore X1
3d_coordinate_origin.fxw	Incrocio coordinate 3D
3d_zero_point.fxw	Punto zero 3D

Tornitura

Modello	Descrizione
turning_blank.fxw	Il pezzo grezzo per la tecnologia di tornitura: un cilindro semplice
turning_centerline.fxw	Linea centrale
turning_centerpoint.fxw	Sistema di coordinate per la rappresentazione del punto zero nel sistema di coordinate pezzo
turning_refpoint.fxw	Punto di riferimento, ad es. per una lavorazione
turning_machining_area.fxw	Linee di limitazione per la superficie di lavorazione

Fresatura

Modello	Descrizione
milling_blank.fxw	Il pezzo grezzo per la tecnologia di fresatura: un parallelepipedo semplice
milling_centerline.fxw	Linea centrale
milling_refpoint.fxw	Punto di riferimento, ad es. per una lavorazione

C.3 Comandi XML

C.3.1 Panoramica

Affinché X3D Viewer possa accedere ai grafici è necessario il file di descrizione scene (*.xml). Nel file di descrizione scene si assegnano i nomi delle scene che vengono richiamati dalla progettazione ai tempi del file *.x3d (TimeSensor) in cui si trovano i grafici.

C.3.2 Struttura del file di descrizione scene

Di seguito viene illustrata la struttura del file di descrizione scene con i relativi comandi XML:

Struttura e descrizione

<!-- Trattamento di testi dipendenti dal piano nella tornitura -->

```
<TextPlane plane="G18_ZXY" />
```

Descrizione

<TextPlane plane="G18_ZXY" />	Trattamento di testi dipendenti dal piano (nomi assi) specifico per lavorazioni di tornitura. Se si utilizza ad es. un testo con l'identificativo "Z" (per G18 il primo asse geometrico), nella figura di help viene rappresentato il corrispondente identificativo dell'asse geometrico del controllore (ad es. "Z").

<!--Allineamento del testo-->

```
<TextPosition center='true' />
```

Descrizione

<TextPosition center='true' />	Indica che i testi sono centrati. Questo è rilevante per la rappresentazione dei testi in immagini ruotate. Si consiglia di utilizzare solo questa impostazione.

<!--Adattamento della velocità utensile-->

```
<ToolSpeedFactors planeSpeedFactor="0.4" rapidSpeedFactor="0.7"
reducedSpeedFac¬tor="1.0"/>
```

Descrizione

<ToolSpeedFactors	Questa voce può essere aggiunta per modificare le velocità dell'utensile.
planeSpeedFactor="0.4"	Fattore per la velocità di avanzamento (0.4 = 40%)
rapidSpeedFactor="0.7"	Fattore per la velocità in rapido (0.7 = 70%)
reducedSpeedFactor="0.1" />	Fattore per una terza velocità (0.1 = 10%)

<!--Definizione di un'animazione-->

```
<SceneKey name='BoringAnimation' masterRotationSpeed="-64.0"
maxRotAngle="45.0" begin-Time='0.110 endTime='0.118 view='camiso'
speedMaster='boringtool' Type="VIEW_3D_TURN_CYL">
```

Descrizione

<SceneKey name='BoringAnimation'	Nome dell'animazione
masterRotationSpeed="-64.0"	Senso di rotazione e velocità di rotazione dell'utensile. L'impostazione è opzionale.
maxRotAngle="45.0"	Viene evitato l'effetto alias (l'utensile sembra ruotare nella direzione errata). Il valore è generalmente un quarto della simmetria dell'utensile. L'impostazione è opzionale.
beginTime='0.110	Ora di inizio dell'animazione sul timesensor
endTime='0.118'	Ora di fine dell'animazione sul timesensor
view='camiso'	Fotocamera utilizzata per l'animazione
speedMaster='boringtool'	Nome dell'utensile per l'animazione
type="VIEW_3D_TURN_CYL">	Definisce il tipo di vista (vedere il capitolo Tipo di vista (Pagina 328))

<!-- Elementi -->

```
<Element name='1' time='0.110' feed='rapid' dwell=’2’/>
<Element name='2' time='0.112' feed='rapid'/>
<Element name='3' time='0.114' feed='rapid'/>
<Element name='4' time='0.116' feed='plane'/>
<Element name='5' time='0.118' feed='plane'/>
```

Descrizione

<Element name='1'	Indica che si tratta del primo elemento dell'animazione.
time='0.110'	Ora del primo elemento sul timesensor
feed='rapid'	Velocità di traslazione dell'elemento plane = avanzamento di lavorazione rapid = rapido reduced = velocità ridotta, più lenta di "plane"
dwell=’2’/>	Pausa nel movimento di traslazione dell'animazione. Un utensile rotante resta in rotazione.

```
<!-- Fine della definizione dell'animazione -->
</SceneKey>
```

Descrizione

</SceneKey>	Fine della definizione dell'animazione

```
<!-- animazione esistente come sorgente di una nuova animazione -->
<SceneKey source=''BoringAnimation' name=''BoringAnimationRear' mirror='
MirrorScreenX ' />
```

Descrizione

<SceneKey source="BoringAnimation'	Nome di un'animazione che deve fungere da sorgente di una nuova animazione.
name="BoringAnimationRear'	Nome della nuova animazione in base alla sorgente
mirror=' MirrorScreenX ' />	Specularità in direzione dell'asse X, l'output avviene nella nuova animazione.

```
<!-- Definizione di una scena statica (figura), (4 esempi) -->
<SceneKey name='Default' time='0.026' view='camiso'
type="VIEW_3D_DRILL_CUT"/>
<SceneKey name='Cut' time='0.046' view='camside' type="VIEW_SIDE"/>
<SceneKey name='Z1ink' time='0.056' view='camside' type="VIEW_SIDE"
 highLightedGroup='dad_Z1ink'/>
<SceneKey name='Z1abs' time='0.066' view='camside' type="VIEW_SIDE"
 highLightedGroup='dad_Z1abs'/>
```

Descrizione

<SceneKeyname='Z1abs'	Nome della figura
time='0.066'	Ora della figura sul timesensor
view='camside'	Fotocamera utilizzata per la figura
type="VIEW_SIDE"	Definisce il tipo di vista (vedere il capitolo Tipo di vista (Pagina 328))
highLightedGroup='dad_Z1abs'/>	Nome del gruppo che deve essere evidenziato. Il gruppo è stato chiamato "Z1abs" in FluxStudio. Il prefisso "dad_" viene aggiunto automaticamente da Flux.

```
<!-- end of xml file –!>
```

C.3.3 Specularità e rotazioni

Il comando **mirror** permette di definire la specularità o la rotazione della figura/modello.

Descrizione

mirror="RotateScreenX"	La figura viene ruotata intorno all'asse X
mirror="RotateScreenY"	La figura viene ruotata intorno all'asse Y
mirror="RotateScreenZ"	La figura viene ruotata intorno all'asse Z
mirror="MirrorScreenX"	La figura viene rispecchiata in direzione dell'asse X
mirror="MirrorScreenY"	La figura viene rispecchiata in direzione dell'asse Y
mirror="MirrorScreenZ"	La figura viene rispecchiata in direzione dell'asse Z
mirror="RotatePieceX"	Il modello viene ruotato intorno all'asse X
mirror="RotatePieceY"	Il modello viene ruotato intorno all'asse Y
mirror="RotatePieceZ"	Il modello viene ruotato intorno all'asse Z
mirror="MirrorPieceX"	Il modello viene rispecchiato in direzione dell'asse X
mirror="MirrorPieceY"	Il modello viene rispecchiato in direzione dell'asse Y
mirror="MirrorPieceZ"	Il modello viene rispecchiato in direzione dell'asse Z

Tutte le possibilità di rotazione e specularità possono essere combinate. La rotazione richiede la definizione di un angolo di rotazione.

Esempi

mirror="RotatePieceZ=180"

mirror="RotatePieceZ=-90 MirrorScreenX"

mirror="RotateScreenY=90"

mirror="MirrorPieceX MirrorPieceY"

mirror="MirrorPieceZ RotatePieceX=-90"

C.3.4 Tipo di vista

Il tipo di vista permette di definire le viste. Le viste si basano su valori sperimentali e regole che portano a una rappresentazione plausibile degli oggetti.
Viene così garantito che la rappresentazione degli oggetti avvenga conformemente all'impostazione del sistema di coordinate dell'interfaccia utente (MD 52000 $MCS_DISP_COORDINATE_SYSTEM o MD 52001 $MCS_DISP_COORDINATE_SYSTEM_2).

Descrizione

"VIEW_STATIC"	Vista diretta senza conversione
"VIEW_3D_TURN_CYL"	Vista 3D per lavorazioni di tornitura (cilindro)

"VIEW_3D_MILL_CUBE"	Vista 3D per lavorazioni di fresatura (parallelepipedo)
"VIEW_3D_DRILL_CUT"	Vista 3D per lavorazioni di foratura (rappresentazione di sezione)
"VIEW_SIDE"	Vista 2D per lavorazioni di foratura (rappresentazione di sezione)
"VIEW_TOP_GEO_AX_1"	Vista 2D dalla direzione del 1° asse geometrico
"VIEW_TOP_GEO_AX_2"	Vista 2D dalla direzione del 2° asse geometrico
"VIEW_TOP_GEO_AX_3"	Vista 2D dalla direzione del 3° asse geometrico

C.4 Conversione in file hmi

Nota

I file X3D inclusi i relativi file XML vengono convertiti in file HMI durante l'avviamento dell'HMI.

Per ogni file X3D deve essere creato un corrispondente file XML con lo stesso nome. A questo scopo occorre salvare i file X3D e i file XML nella directory `HMI\ico\x3d\turning` o `milling`. La procedura è analoga a quella per i file .ts.

C.5 Visualizzazione in Create MyHMI /3GL

C.5.1 X3D Viewer

Per visualizzare in modo mirato le figure di help in una propria applicazione OA, occorre integrare in quest'ultima il widget X3D Viewer.

Il widget X3D Viewer mette a disposizione interfacce che consentono la presentazione di contenuti X3D nell'HMI.

Per rappresentare scene grafiche, è disponibile la classe SlX3dViewerWidget. La definizione della classe si trova nel corrispondente file header slx3dviewerwidget.h nella directory GUI Include globale `\hmi_prog\gui\include`.

C.5.2 Classe SlX3dViewerWidget

La classe propone un widget utilizzabile in modo flessibile che rappresenta autonomamente i contenuti di un file di modello specificato durante il tempo di esecuzione e che eventualmente fa scorrere l'animazione.

L'interfaccia della classe è costituita da costruttore, distruttore e due metodi per il controllo dell'uscita grafica.

Come derivazione diretta del QWidget classe Qt è disponibile un'interfaccia molto più ampia e non descritta ulteriormente in questa sede (ad es. show(), hide() e resize(...). Per maggiori informazioni in proposito consultare la documentazione Qt).

C.5.3 Metodi pubblici

SlX3dViewerWidget (QWidget* pParent = 0)

Costruttore del widget X3D Viewer.

Parametri	Significato
pParent	Il parametro viene inoltrato al costruttore del Qwidget.

~SlX3dViewerWidget ( )

Distruttore del widget X3D Viewer.

Parametri	Significato
-	-

C.5.4 Slot pubblici

void viewSceneSlot (const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene, int nChannel, int nPlane, SlStepTechnology nTechnology)

Il metodo viewSceneSlot fornisce l'istruzione a X3D Viewer di caricare la scena statica rsScene e la scena animata rsAnimationSzene dal file rsFileName e di rappresentarle in successione temporale.

Concretamente viene prima ripetuta la scena statica per un intervallo di tempo fisso e quindi viene visualizzata la scena animata.

Se non è specificata una scena statica, viene presentata subito l'animazione; analogamente può non essere specificata una scena animata.

Il numero del canale, il piano e la tecnologia consentono di ruotare le scene nella posizione corretta (in funzione del sistema di coordinate macchina impostato).

Parametri	Significato
rsFileName	Nome del file in cui sono contenute le scene da rappresentare.
rsScene	Nome della scena statica.
rsAnimationScene	Nome della scena animata.
nChannel	Numero di canale

Parametri	Significato
nPlane	Piano
nTechnology	Tecnologia Per il tipo di enumerazione SlStepTechnology sono definite le costanti seguenti: • SL_STEP_NO_TECHNOLOGY • SL_STEP_MILLING • SL_STEP_TURNING • SL_STEP_SURFACE_GRINDING • SL_STEP_CIRCULAR_GRINDING

void viewSceneSlot (const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene)

Una forma semplificata del metodo viewSceneSlot con cui si fornisce l'istruzione a X3D Viewer di caricare la scena statica rsScene e/o la scena animata rsAnimationScene dal file rsFileName e di rappresentarle.

Parametri	Significato
rsFileName	Nome del file in cui sono contenute le scene da rappresentare.
rsScene	Nome della scena statica.
rsAnimationScene	Nome della scena animata.

C.5.5 Librerie

Per poter impiegare X3D-Viewer in progetti dell'utente, occorre aggiungere la voce 'slx3dviewer.lib' alla lista delle dipendenze delle librerie.

C.5.6 Esempio di implementazione

Un esempio di implementazione è contenuto nel pacchetto Create MyHMI/3GL in **\examples \GUIFrameWork\SlExGuiX3D**.

C.5.7 Dati macchina

Dato macchina di visualizzazione MD 9104 $MM_ANIMATION_TIME_DELAY

Ritardo prima dell'impiego dell'animazione nelle figure di help in secondi. Nelle figure di help costituite esclusivamente da animazione questa impostazione non ha effetto.

L'impostazione agisce a livello globale per tutte le animazioni di SINUMERIK Operate.

C.5.8 Note relative all'applicazione

- L'animazione si interrompe quando il widget X3D Viewer viene disattivato. In questo caso non è necessario effettuare la selezione di scene prive di animazione.
- Occorre infatti evitare di istanziare spesso o ripetutamente il widget X3D Viewer a causa delle possibili ripercussioni sulle prestazioni e sull'occupazione di memoria. Per questi casi applicativi si consiglia l'uso (l'implementazione) di un singleton X3D Viewer.
- X3D Viewer consente anche l'impiego in concetti segnale-slot.
- Se si verifica un errore (ad es. impossibile trovare il file o nome di scena sconosciuto), il widget X3D Viewer visualizza una finestra di messaggio. Questa finestra scompare automaticamente appena viene selezionato un altro file di figure di help.

C.6 Visualizzazione in Run MyScreens

Questo capitolo è destinato a sviluppatori esperti di "Run MyScreens". Per informazioni sulle conoscenze di base necessarie consultare la relativa documentazione.

Oltre all'uso di figure in formato .bmp o .png è anche possibile visualizzare figure di help animate con X3D Viewer.

Per inserirle utilizzare come al solito l'interfaccia Run MyScreens per figure di help.

Per l'output di figure in formato .bmp o .png, nella definizione di una finestra di dialogo nella sezione Attributi viene inserito il dato XG0 oppure il parametro viene lasciato vuoto. Per inserire figure di help X3D in una finestra di dialogo, occorre immettere il dato **XG1** negli attributi.

```
//M(MASK_F_DR_O1_MAIN/$85407////52,80/XG1)
```

Per comandare le singole figure di help sono necessari i seguenti parametri, il cui significato è descritto nel capitolo 5.3 Slot pubblici:

1. Nome file
2. StaticScene (opzionale)
3. AnimationScene (opzionale)
4. Tecnologia (opzionale)
5. Piano (opzionale)

I parametri vengono riassunti in quest'ordine in una stringa separati da una virgola.

```
Hlp = "Nome file,StaticScene,AnimationScene,Tecnologia,Piano"
```

Esempi

- La figura di help predefinita può essere impostata con la variabile standard Hlp.
  Nell'esempio seguente, dal file "MyDlgHelp.hmi" viene emessa l'animazione "MyAnimation". Non è specificata una StaticScene, per cui non viene emessa alcuna scena statica. Per la tecnologia e per il piano non vi sono indicazioni, per cui vengono usati i valori predefiniti.
  ```
  Hlp = "MyDlgHelp.hmi,,MyAnimation"
  ```
- Per i singoli parametri della finestra di immissione la proprietà hlp può essere impostata a una figura di help specifica nella variabile relativa.
  Nell'esempio seguente, dal file "MyDlgHelp.hmi" vengono emesse la scena statica "MyParam" e l'animazione "MyAnimation" nel piano G17. Per la tecnologia non vi è alcuna indicazione, per cui viene usato il valore predefinito.
  ```
  VarMyParam.hlp = "MyDlgHelp.hmi,MyParam,MyAnimation,,G17"
  ```

Indice analitico

A

Albero di comando, 37
Allarmi
  Identificativi delle lingue, 310
App "Siemens Industry Online Support", 13
Array
  Definizione, 196
  Elemento, 197
  Indice colonne, 198
  Indice righe, 198
  Modalità di accesso, 198
  Modalità di confronto, 198
  Stato, 201
Attributi, 93

B

Barra di avanzamento senza evidenziazione con colore, 90
Barre di avanzamento con due evidenziazioni con colore, 89

C

Campo di toggle, 92
Campo I/O con selezione di unità integrata, 96
Campo toggle, 87, 100
Casella di riepilogo, 88
Codice Data Matrix, 14
Colore di primo piano, 96
colore di segnale"; "barra di avanzamento, 96
Colore di sfondo, 96
Colori, 96
Colori di sistema, 309
Comandi a sfioramento, 248
Comandi Multitouch, 248
Concatenamento di stringhe, 104
Condizioni, 118
Configurazione standard, 9
Controllo della selezione, 205
Costanti, 117
Custom Widget
  Biblioteca, 206
  Definizione, 206
  Esecuzione di un metodo, 212
  Interfaccia, 207
  Lettura e scrittura di proprietà, 210
  Reazione a un segnale Custom Widget, 215
  Scambio dati automatico, 209
  Scambio dati manuale, 210

D

Dati macchina, 331
Definizione della barra dei softkey, 65
Descrizione comandi, 92, 94
Documentazione mySupport, 11

E

Elemento della finestra di dialogo, 59
ELLISSE (definizione del cerchio), 192
ELLISSE (definizione dell'ellisse), 192

F

File
  Copia, 138
  Eliminazione, 139
  Spostamento, 141
File della Guida, 96
File di progettazione, 35
File DLL, 152
Finestra di dialogo
  a più colonne, 60
  Blocco di definizione, 50
  Definizione, 49
  Proprietà, 51
Finestra di dialogo principale, 160
Finestre di dialogo per password, 62
Formati dati
  fxw, 317
  hmi, 317
  x3d, 317
  xml, 317
Formato numerico, 100
funzione
  Decompilazione del codice NC, 174
  RETURN (Indietro), 173
Funzione
  Accessi ai file, 143
  CALL (Richiamo del sottoprogramma), 135
  CLEAR_BACKGROUND, 194
  CP (Copy Program), 138

CVAR (Check Variable), 137
DEBUG, 146
Decompilazione senza commento, 175
DELETE_GRAPHIC, 195
DLGL (Dialog Line), 145
DO-LOOP), 186
DP (Delete Program), 139
EP (Exist Program), 140
Esecuzione ciclica degli script, 188
EVAL (Evaluate), 151
EXIT, 147
EXITLS (EXIT Loading Softkey), 152
FCT, 152
FORMAT (String), 185
GC (Generate Code), 154
HIDE_GRAPHIC, 195
HMI_LOGIN, 157
HMI_LOGOFF, 157
HMI_SETPASSWD, 157
INSTR (String), 180
LA (Load Array), 158
LB (Load Block), 159
LEFT (String), 181
LEN (String), 180
Lettura e scrittura di parametri di azionamento, 133
LISTADDITEM), 149
LISTCLEAR), 149
LISTCOUNT), 149
LISTDELETEITEM, 149
LISTINSERTITEM, 149
LM (Load Mask), 160
LS (Load Softkey), 162
MIDS (String), 182
MP (Move Program), 141
MRDOP), 133
MRNP (Multiple Read NC PLC), 165
P_LOGICAL_FILEPATH, 167
P_REAL_FILEPATH, 167
Panoramica, 132
PI_START, 168
RDFILE), 143
RDLINEFILE), 143
RDOP), 133
REPLACE (String), 182
RESIZE_VAR_IO, 170
RESIZE_VAR_TXT, 170
RIGHT (String), 182
RNP (Read NC PLC Variable), 168
SB (Search Backward), 178
SF (Search Forward), 178
SHOW_GRAPHIC, 196
SL_HMI_INSTALL_DIR, 179
SL_TEMP_DIR, 179
SP (Select Program), 142
START_TIMER), 188
STOP_TIMER, 188
STRCMP (String), 183
STRINSERT (String), 183
STRREMOVE (String), 184
SWITCH), 164
TRIMLEFT (String), 184
TRIMRIGHT (String), 184
UNTIL, 186
WDOP, 133
WHILE, 186
WNP (Write NC PLC Variable), 169
WRFILE, 143
WRLINEFILE), 143
Funzioni matematiche, 116
Funzioni trigonometriche, 116

G

Generazione del codice NC, 154
grid → tabella, 201

H

H_SEPARATOR (definizione della linea di separazione orizzontale), 193

I

Identificativi delle lingue, 310
Immagine come testo sintetico, 88
Importazione
Grafici (modelli), 321
Inviare feedback, 11

L

LINE (definizione di linea), 191
Livelli di protezione, 308
Livello di accesso, 66

M

Metodo
ACCESSLEVEL, 120
CHANGE, 120
CHANNEL, 122

CONTROL, 122
LANGUAGE, 124
LOAD, 124
LOAD GRID, 163
OUTPUT, 125
Panoramica, 120
PRESS, 126
PRESS(ENTER), 127
PRESS(TOGGLE), 128
RESOLUTION, 128
RESUME, 129
SUSPEND, 130
UNLOAD, 125
Modalità di immissione, 94
Modalità di immissione della password (asterisco), 91
Modalità di passaggio tra finestre di dialogo, 161
Modalità di scrittura, 96
Modalità di visualizzazione, 93

O

OpenSSL, 14
Operatore
Bit, 118
Matematico, 115
Operatori di confronto, 117
Opzione di visualizzazione, 93

P

Pagina di help, 95
Pagine web di terze parti, 10
Position
Ein-/Ausgabefeld, 103
Kurztext, 103
Posizione
Campo di input/output, 95
Testo sintetico, 95
Preassegnazione, 92
Product Support, 12
Progettazione dei softkey del PLC, 283

R

RECT (definizione rettangolo), 192
Registro
Scambio di dati, 171
Stato, 172
Valore, 171
Regolamento generale sulla protezione dei dati, 14

S

Servizi PI, 132
Siemens Industry Online Support
App, 13
SIEsButton
Panoramica delle Properties, 250
SIEsGraphCustomWidgets
addArc, 241
addCircle, 241
addContour, 234
addEllipse, 240
addLine, 239
addPoint, 239
addRect, 240
addRoundedRect, 240
addText, 241
AxisNameX, 221
AxisNameY, 221
AxisY2Factor, 222
AxisY2Offset, 221
AxisY2Visible, 221
BackColor, 227
ChartBackColor, 228
clearContour, 236
CursorColor, 229
CursorStyle, 230
CursorX, 229
CursorY, 230
CursorY2, 230
findX, 237
fitViewToContour, 236
fitViewToContours, 236
ForeColor, 228
ForeColorY2, 228
GridColor, 228
GridColorY2, 229
hideAllContours, 235
hideContour, 235
KeepAspectRatio, 226
moveCursorOnContourBack, 246
moveCursorOnContourBegin, 244
moveCursorOnContourEnd, 245
moveCursorOnContourNext, 245
Panoramica dei segnali, 247
Panoramica delle funzioni, 232
Panoramica delle Properties, 220
removeContour, 235
repaint, update, 239
ScaleTextEmbedded, 223
ScaleTextOrientationYAxis, 224

SelectedContour, 227
serialize, 246, 267
setCursorOnContour, 244
setCursorPosition, 243
setCursorPositionY2Cursor, 243
setFillColor, 243
setIntegralFillMode, 238
setMargins, 267
setMaxContourObjects, 234
setPenColor, 242
setPenStyle, 242
setPenWidth, 242
setPolylineMode, 237
setView, 233
showAllContours, 235
showContour, 234
ShowCursor, 229
ViewChanged, 247
ViewMoveZoomMode, 231
SIEsTouchButton, (Testi dipendenti dalla lingua)
BackColor, 260
BackColorChecked, 261
BackColorDisabled, 262
BackColorPressed, 261
BackgroundPicture, 264
BackgroundPictureAlignment, 264
BackgroundPictureAlignmentString, 265
BackgroundPictureKeepAspectRatio, 266
ButtonStyle, 251
Checkable, 253
checked, 269
Checked, 253
clicked, 269
clickedDisabled, 269
Enabled, 252
Flat, 251
Lettura e scrittura di proprietà, 249
Panoramica dei segnali, 268
Panoramica delle funzioni, 267
Picture, 255
PictureAlignment, 256
PictureAlignmentString, 256
PictureKeepAspectRatio, 257
PicturePressed, 255
ScaleBackgroundPicture, 266
ScalePicture, 257
ShowFocusRect, 254
Testo, 257
TextAlignedToPicture, 260
TextAlignment, 258
TextAlignmentString, 259
TextColor, 262
TextColorChecked, 262
TextColorDisabled, 263
TextColorPressed, 263
TextPressed, 258
Simbolo di toggle, 93
sintassi di progettazione"; "colonne di tabella, 48
sintassi di progettazione"; "maschere, 46
sintassi di progettazione"; "sintassi estesa, 46
sintassi di progettazione"; "softkey, 47
sintassi di progettazione"; "variabili, 47
SINUMERIK, 9
SlEsGraphCustomWidget
Lettura e scrittura di proprietà, 220
slhlp.xml, 79
Slot pubblici, 330
SlX3dViewerWidget, 329
Softkey
Assegnazione di proprietà, 65
Proprietà, 67
Softkey di accesso, 37, 38
Sottofinestra di dialogo, 160
Sottoprogramma, 132
Codice di blocco, 136
Interruzione, 173
richiamo, 135
Variabile, 136
Stato della variabile, 83
Supporto per catene sequenziali, 175

T

Tabella
Definizione, 201
Definizione delle colonne, 204
Programmazione, 203
Technical Support, 13
Testi dipendenti dalla lingua, (SIEsTouchButton)
Testo, 92
Testo completo, 92
Testo grafico, 92
Testo sintetico, 92
Testo unità, 92
Tipo di variabile, 91
INTEGER, 98
VARIANT, 99
Training, 13

V

V_SEPARATOR (definizione della linea di separazione verticale), 193

Valore della variabile, 83
valori di segnale"; "barra di avanzamento, 92
Valori limite, 92
variabile
    CURVER, 106
    ENTRY, 107
    FILE_ERR, 108
Variabile
    Calcolo, 84
    CURPOS, 106
    ERR, 107
    FOC, 109
    Modifica delle proprietà, 96
    Parametri, 91
    S_ALEVEL, 110
    S_CHAN, 111
    S_CONTROL, 112
    S_LANG, 112
    S_NCCODEREADONLY, 113
    S_RESX, 113
    S_RESY, 113
    Trasferimento, 147
    Verifica, 137
Variabile ausiliaria, 84
Variabile di sistema, 85, 95
Variabile NC
    lettura, 168
    Scrittura, 169
Variabile utente, 95
Variabili PLC
    lettura, 168
    Scrittura, 169
Velocità di aggiornamento, 93
Vivaty Studio, 318