SIEMENS

SINUMERIK

SINUMERIK 828D
Easy XML

編程手冊

適用範圍：

CNC 軟體　　版本 4.95

07/2021
6FC5398-3EP40-0MA0

法律聲明

警告事項意涵

為了您的人身安全以及避免財產損失，必須注意本手冊中的提示。有關人身安全的提示通過一個警告三角表示，僅與財產損失有關的提示不帶警告三角。

⚠ 危險
表示如果不采取相應的小心措施，將會導致死亡或者嚴重的人身傷害。

⚠ 警告
表示如果不采取相應的小心措施，可能導致死亡或者嚴重的人身傷害。

⚠ 小心
表示如果不采取相應的小心措施，可能導致輕微的人身傷害。

注意
表示如果不采取相應的小心措施，可能導致財產損失。

當出現多個危險等級的情況下，每次總是使用最高等級（較低數字）的警告提示。如果在某個警告提示中帶有警告可能導致人身傷害的警告三角，則可能在該警告提示中另外還附帶有可能導致財產損失的警告。

合格的崙業人員

唯有與各項工作要求**資格符合的人員**才能操作本文件所屬產品/系統，遵照各附帶文件說明，特別是其中的安全及警告提示。資格符合的人員由於具備相關訓練及經驗，可以察覺本產品/系統的風險，並避免可能的危險。

Siemens 產品

請注意下列說明：

⚠ 警告
Siemens 產品只允許用於目錄和相關技術文件中規定的使用情況。如果要使用其他公司的產品和組件，必須得到 Siemens 推薦和允許。正確的運輸、儲存、組裝、裝配、安裝、調試、操作和維護是產品安全、正常運行的前提。必須保證允許的環境條件。必須注意相關檔中的提示。

商標

所有帶有標記符號 ® 的都是 Siemens AG 的注冊商標。標簽中的其他符號可能是一些其他商標，任何第三方將其用于其他目的都會損坏所有者的利益。

責任免除

我們已對印刷品中所述內容與硬件和軟件的一致性作過檢查。然而不排除存在偏差的可能性，因此我們不保証印刷品中所述內容與硬件和軟件完全一致。印刷品中的數據都按規定經過檢測，必要的修正值包含在下一版本中。同時歡迎您提出改進建議。

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
德國

文檔訂購號: 6FC5398-3EP40-0MA0
Ⓟ 07/2021 本公司保留更改的權利

目錄

簡介 1

1.1 關於 SINUMERIK

從簡單的標準化 CNC 機台到優質的模組化機台設計，SINUMERIK CNC 都可為所有機台概念提供正確的解決方案。無論是個別零件或是大量生產、簡單或複雜的工件，SINUMERIK 都是高度動態的自動化解決方案，適合整合在所有生產領域上。從原型結構、刀具設計到模具製造，再到大規模的序列生產。

若需詳細資訊，請造訪官方網站 SINUMERIK (https://www.siemens.com/sinumerik)。

1.2 關於本文件

目標群組

本出版品適用於下列對象：

- 程式設計師
- 專案工程師

用途

編程手冊讓目標群體能夠使用以 XML 為基礎的命令集元素，來設計客戶及應用程序專屬的使用者介面。

標準範圍

本文件僅說明標準版本的功能。此處可能與實際提供的系統功能範圍有所不同。所提供的驅動系統功能請務必參閱訂購文件。

可以執行本文件未說明的系統中其他功能。然而，並不代表應於新控制系統內或維修時提供此類功能。

為簡明起見，本文件並未包括所有產品類型的詳細資訊。此外，本文件也無法考慮各種可能的安裝、操作及維修/保養等類型。

機台製造商必須記錄他們對產品本身所做的任何新增或修改。

第三方公司的網站

本文件可能包含指向第三方網站的超連結。西門子對這些網站及其內容概不負責，也不承擔任何責任。西門子無法控制這些網站上所顯示的資訊，也無法對其中所提供的內容及資訊負責。使用者必須承擔在使用時的風險。

1.3 網際網路上的文件

1.3.1 SINUMERIK 828D 文件概觀

4.8 SP4 及更高版本 SINUMERIK 828D 配備功能的相關綜合文件，請見 828D 文件概觀 (https://support.industry.siemens.com/cs/ww/en/view/109766724)。

文件的顯示或下載格式為 PDF 或 HTML5。

文件種類可區分如下：

- 使用者：操作
- 使用者：程式設計
- 製造商 / 服務：設定
- 製造商 / 服務：調試
- 製造商 / 服務：功能
- 製造商/服務：Safety Integrated
- SINUMERIK Integrate/MindApp
- 資訊及訓練

1.3.2 SINUMERIK 操作元件文件概觀

SINUMERIK 操作元件的相關綜合文件，請見 SINUMERIK 操作元件文件概觀 (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW)。

文件的顯示或下載格式為 PDF 或 HTML5。

文件種類可區分如下：

- 操作面板
- 機台控制面板
- 機台按鈕面板
- 手持裝置/迷你手持裝置
- 更多操作元件

SINUMERIK 概觀 - 主題頁面 (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW)提供 SINUMERIK 最重要文件、項目及連結的概觀。

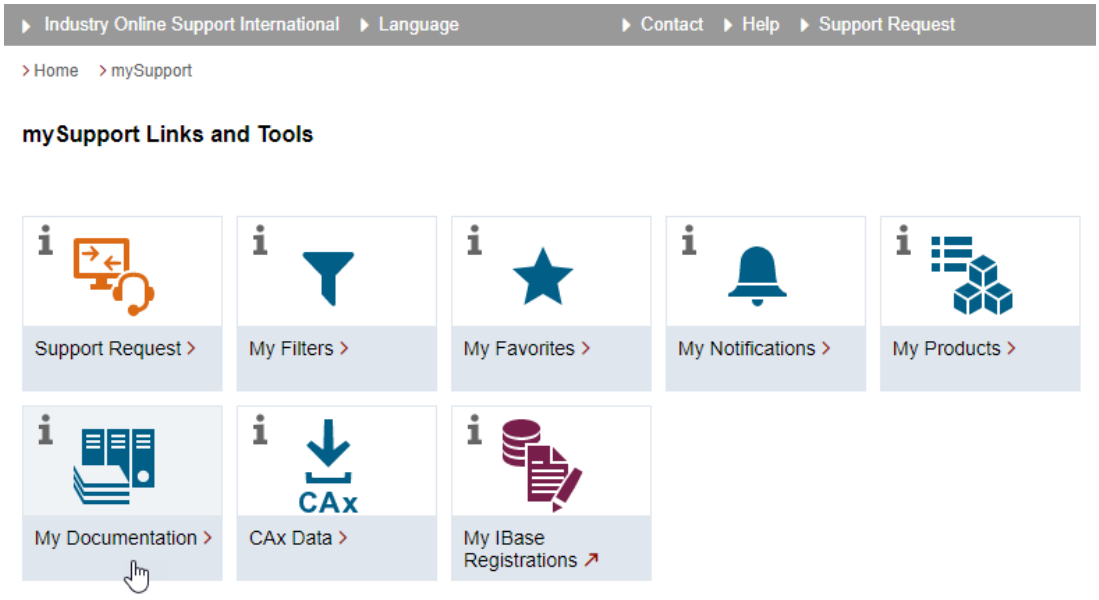
1.4 技術文件的回饋

如果您對「西門子工業線上支援」中發布的技術文件有任何疑問、建議或修正，請使用項目末尾的「傳送回饋」連結。

1.5 mySupport 文件

使用「mySupport 文件」網頁式系統，您可以根據西門子內容編譯自己的個別文件，並修改成自己的機台文件。

如要啟動應用程式，請按一下 mySupport 首頁 (https://support.industry.siemens.com/cs/ww/en/my)上的「我的文件」圖標：



設定的手冊可以用 RTF、PDF 或 XML 格式匯出。

說明

可以透過「設定」連結來標識支援 mySupport 文件應用程式的西門子內容。

1.6 服務和支援

產品支援

有關產品的詳細資訊請見下列網址：

產品支援 (https://support.industry.siemens.com/cs/ww/en/)

下列項目在此位址中提供：

- 最新產品資訊（產品公告）
- FAQ（常見問題）
- 手冊
- 下載項目
- 包含有關您產品最新資訊的電子報
- 使用者與專家之間共享資訊及最佳實務的全球論壇
- 透過我們在西門子資料庫中的聯絡資訊聯繫當地聯絡人（→「聯絡資訊」）
- 有關現場服務、維修、備品等資訊（→「現場服務」）

技術支援

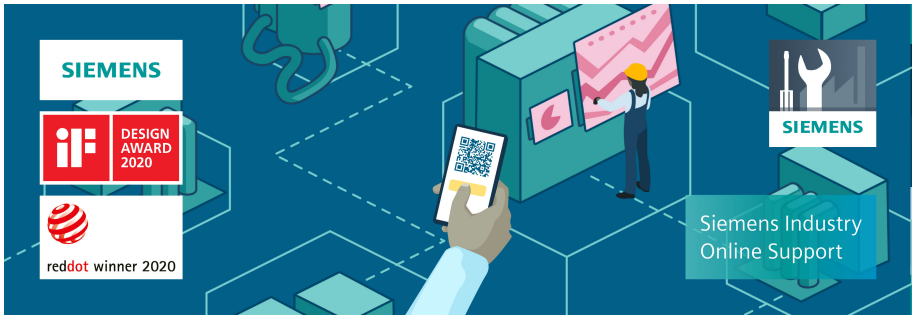
在網際網路上「聯絡資訊」區的位址 (https://support.industry.siemens.com/cs/ww/en/sc/4868)中，有各國專屬的技術支援電話號碼。

如有任何技術問題，請使用「支援請求」區中的線上表單。

訓練

您可以在下列位址 (https://www.siemens.com/sitrain)找到 SITRAIN 的資訊。
SITRAIN 提供西門子的自動化及驅動產品、系統與解決方案等訓練課程。

西門子支援隨身帶著走



透過屢獲殊榮的「西門子工業線上支援」應用程式，您可以隨時隨地存取超過 30 萬份西門子工業產品的文件。此應用程式可支援的領域包括：

- 解決專案在實施中遇到的問題
- 出現故障時的故障排除
- 擴充系統或規劃新系統

此外，您還可以存取技術論壇，以及本公司專家的其他文章：

- FAQ
- 應用範例
- 手冊
- 憑證
- 產品公告等

「西門子工業線上支援」應用程式提供 Apple iOS 及 Android 系統。

銘牌上的二維條碼

銘牌上的二維條碼包含一些專屬的裝置資料。可使用智慧型手機讀取此代碼，並透過「工業線上支援」行動應用程式來顯示有關裝置的技術資訊。

1.7 重要產品資訊

使用 OpenSSL

此產品可以包含下列軟體：

- 由 OpenSSL 專案所開發，用於 OpenSSL 工具包的軟體
- 由 Eric Young 所建立的加密軟體。
- 由 Eric Young 所開發的軟體

詳細資訊請見下列網址：

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

符合一般資料保護規範

西門子遵循標準資料保護原則，特別是資料最小化規則（設計保密）。

針對此產品，表示：

此產品不會處理或儲存任何個人資料，只限於技術功能資料（例如時間戳記）。若使用者將此資料與其他資料連結（例如調度計畫）或是使用者將個人相關資料儲存在相同的資料媒體（例如硬碟）中，從而將此資料個人化，此時使用者必須確保符合適用的資料保護規定。

基本安全指示

2

2.1 一般安全指示

⚠ 警告

因其他能源而造成電擊與人員生命危險

觸碰帶電元件可能會導致死亡或重傷。

- 僅合格人員可以在電氣裝置上進行作業。
- 務必隨時遵守特定國家的安全規範。

一般而言，安全規範共有以下六項實施要點：

1. 準備斷接。通知所有受程序影響的人員。
2. 隔離電源供應器的驅動系統，並採取防止再被切換回去的必要措施。
3. 等待警告標籤上所載之放電時間經過。
4. 檢查任何電源連接之間，以及任何電源連接與保護導體連接之間是否無電壓。
5. 檢查輔助電源電路是否為已經斷電。
6. 確定馬達無法移動。
7. 檢查所有其他危險能源，例如壓縮空氣、液壓系統或水。將能源切換至安全狀態。
8. 檢查正確的驅動系統是否已完全鎖定。

當您完成所有作業後，以相反順序將機器恢復到準備就緒的作業狀態。

⚠ 警告

連接不合適的電源供應器會導致電擊

設備連接至不穩定的電源供應器時，外露的元件可能有危險的電壓。接觸到危險的電壓，可能造成人員重傷或死亡。

- 所有電子模組連接與端子僅可使用 SELV（安全超低電壓）或 PELV-（防護性超低電壓）輸出電壓。

⚠ 警告

設備損壞會導致電擊

不正確的處理可能會造成設備損壞。對於損壞裝置而言，其機殼或外露的元件可能帶有致命的電壓；如果碰觸，可能會導致死亡或重傷。

- 在運輸、儲存與操作時，請確保相關數據符合技術資料上所載限制值。
- 請勿使用任何損壞裝置。

⚠ 警告

纜線屏蔽層未連接會導致電擊

電容交叉耦合時，若纜線屏蔽層未連接，可能會產生危險接觸電壓。

- 在最低限度的情況下，將纜線屏蔽層與未使用的纜線線芯連接至一端的接地外殼電位。

⚠ 警告

若無接地連接會導致電擊

若具有 I 級保護的裝置缺少或不正確連接防護導體，裝置的開放或外露零件可能會存在高電壓，若不慎觸碰可能會造成人員死亡或重傷。

- 請依照適用規範對裝置進行接地。

注意

因不合適的鎖緊工具而造成設備損壞。

不合適的鎖緊工具或固定方法可能會損壞設備的螺絲。

- 務必只使用與螺絲頭完全符合的螺絲起子。
- 使用技術文件中指定的扭矩來鎖緊螺絲。
- 使用具有動態扭矩感測器和限速系統的扭力扳手或機械精密螺帽鎖緊機。

⚠ 警告

從內建裝置蔓延火勢

發生火災事件時，內建裝置的機殼無法防止火勢及煙霧擴散。這會導致人員重傷或資產損失。

- 將內建單元安裝在適當的金屬機箱內，以此方式保護人員免於受到火災及煙霧的危害，或是採取其他的適當措施保護人員。
- 確保煙霧只能經由受控制與監視的通道排放。

⚠ 警告

因無線電裝置或行動電話造成的機器意外動作

在靠近元件附近使用無線電裝置或行動電話可能會導致設備故障。這將會損害機器的功能安全，使人員暴露於危險之中，或造成資產損失。

- 因此，如果離元件的距離小於 20 公分，請務必關閉無線電裝置或行動電話。
- 只有在電源已經關閉的設備上，才使用「SIEMENS 工業線上支援 App」。

⚠ 警告

因不適當通風間隙造成的火災

不適當的通風間隙能造成元件過熱，隨後發生火災及煙霧。這可能會造成重傷或甚至死亡。同時也可能會導致停機時間增加，縮減裝置/系統的使用壽命。

- 確保通風間隙符合各元件指定的最小通風間隙。

注意

由於安裝位置不允許而導致過熱

若安裝在不允許的位置，裝置可能會過熱而因此損壞。

- 只能在允許的安裝位置操作裝置。

⚠ 警告

由於沒作用的安全功能導致機器意外動作

沒作用或未調整的安全功能可能觸發意外的機器動作，導致重傷或死亡。

- 在開始調試前請先參閱相關的產品文件。
- 對整個系統的相關安全功能進行安全檢查，包括所有安全相關元件。
- 確保驅動器與自動化作業的安全功能已經以正確的參數調整並啟動。
- 執行功能測試。
- 只有在能夠確保所有安全相關功能皆能正確運作時，才能將工廠投入實際作業。

說明

安全性整合功能的重要安全注意事項

若要使用安全性整合功能，您必須遵守安全性整合手冊中的安全注意事項。

⚠ 警告

錯誤或經變更的參數設定會造成機器故障

錯誤或經變更的參數設定會導致機台故障，並導致傷害或死亡。

- 保護參數設定不受未經授權的存取。
- 處理可能的故障應採取適當的措施，例如緊急停止或緊急關閉。

2.2 由於電場或靜電放電造成設備損壞

靜電感應裝置 (ESD) 是可能會因為電場或靜電放電而損壞的個別元件、積體電路、模組或裝置。

注意

由於電場或靜電放電造成設備損壞

電場或靜電放電可能會損壞個別元件、積體電路、模組或裝置，而導致故障發生。

- 在包裝、存放、運輸與寄送電子元件、模組或裝置時，僅可使用其原始包裝或合適的包裝材料，例如鋁箔導電泡棉橡膠。
- 只有在使用以下方法接地後，才可以碰觸元件、模組與裝置：
  - 配戴 ESD 腕帶
  - 在配有導電地板的 ESD 區域中配戴 ESD 鞋或 ESD 接地片
- 僅可將電子元件、模組或裝置放置於導電表面（有 ESD 表面的桌子、ESD 導電泡綿、ESD 包裝、ESD 運輸容器）。

2.3 應用範例的保證與責任

應用範例不具約束力，未聲稱設定、設備為完整內容，亦不得用於任何可能衍生的不可測性。應用範例並不代表客戶專屬解決方案，但僅能用於針對一般工作提供支援。

作為使用者必須負責確保產品依說明正確地操作。應用範例不得解除您在使用、安裝、操作及維護設備時，對於安全處理應負之責任。

2.4 安全性資訊

Siemens 西門子提供具有工業安全功能的產品和解決方案，以支援設備、系統、機器和網路的安全操作。

為了保護設備、系統、機器和網路免於網路威脅，我們必需要實施及持續保有完整、先進的工業安全概念。西門子的產品和解決方案只是此概念的其中一個環節。

客戶必須負責防止他人未經授權存取其設備、系統、機器和網路。只有在必要的情況下，也唯有在備妥適當的安全性措施 (例如，使用防火牆和網路區隔時) 時，方能將系統、機器和元件連線至企業網路或網際網路。

如需更多可能被執行的工業安全性措施之相關訊息，請造訪以下網址: https://www.siemens.com/industrialsecurity (https://www.siemens.com/industrialsecurity)

西門子持續致力於開發更安全的產品和解決方案。西門子強烈建議您立即套用可用的產品更新，且一律使用最新的產品版本。若使用不再支援的產品版本且無法套用最新的更新，可能會增加客戶遭受網路威脅之風險。

若要持續收到產品更新通知，請訂閱西門子工業安全性 RSS 摘要，訂閱網址為: https://www.siemens.com/industrialsecurity (https://new.siemens.com/global/en/products/services/cert.html#Subscriptions)

若需進一步資訊，請至網站：

工業安全設定手冊 (https://support.industry.siemens.com/cs/cn/zh/view/108862708/en)

⚠ 警告

因軟體操作而導致的不安全操作狀態

軟體操縱，例如病毒、特洛伊木馬程式或蠕蟲，可能導致系統發生不安全的操作狀態，進而引發生命危險、嚴重傷害及財產損失。

- 將軟體保持在最新的狀態。
- 將安裝環境或機台的自動化與驅動元件整合至完整且最先進的工業安全概念。
- 確保您已將所有已安裝的產品納入完整工業安全概念。
- 請利用合適的保護措施（如病毒掃描程式）來保護儲存在可交換式儲存媒體上的檔案遠離惡意軟體。
- 完成調試後，請檢查所有與安全相關的設定。

2.5 動力驅動系統的殘餘風險

在以當地相關法規評估機器或系統相關風險時（例如歐規機械指令），機台製造商或系統安裝公司必須考慮下列由驅動系統的控制與驅動元件所產生的殘餘風險：

1. 在調試、操作、維護與修復期間由以下因素導致驅動機器或系統元件意外移動，例如：
   - 感測器、控制系統、致動器與纜線和連接的硬體及/或軟體錯誤
   - 控制系統及驅動器的回應時間
   - 超出規格的作業及/或環境情況
   - 冷凝/導電污染
   - 參數化、編程、佈線及安裝錯誤
   - 在緊臨電子元件處使用無線裝置/行動電話
   - 外部影響/損壞
   - X 光、離子射線及宇宙射線
2. 發生故障情況時，元件內部及外部可能會發生異常高溫，包括明火以及產生光線、噪音、粒子、氣體等，例如：
   - 元件故障
   - 軟體錯誤
   - 超出規格的作業及/或環境情況
   - 外部影響/損壞
3. 由以下因素導致的危險觸電電壓，例如：
   - 元件故障
   - 靜電電荷期間的影響
   - 移動馬達的電壓感應
   - 超出規格的作業及/或環境情況
   - 冷凝/導電污染
   - 外部影響/損壞
4. 如果裝有心臟起搏器、植入器或金屬人工關節等裝置的人員過於接近，則操作中產生的電子、磁力與電磁場可能對這些人員造成危險
5. 若不當操作系統及/或未能安全且正確地處置元件，可能會產生環境污染物或排放
6. 漣波控制發射器等連網通訊系統或經由網路之通訊資料的影響

關於驅動系統元件殘餘風險的詳細資訊，請參閱使用者技術文件的相關章節。

建立使用者對話方塊 3

3.1 功能範圍

總覽

「產生使用者對話方塊」的功能提供開放式結構，供使用者開發客戶專屬與應用程式專屬的 SINUMERIK Operate 使用者介面。

控制系統提供以 XML 為基礎的指令碼語言，用來產生使用者對話方塊。

這個指令碼語言容許在 SINUMERIK Operate 的 <CUSTOM> 操作區中，顯示機台特有的功能表與對話方塊格式。

所有對話方塊格式都能夠以語言中立的基礎理念來設計。在這種情況下，系統從隨機的語言資料庫中讀取要顯示的文字。

使用

定義的 XML 指令提供下列特性：

1. 顯示對話方塊包含下列元素：
   - 軟鍵
   - 變數
   - 文字與說明文字
   - 圖形與說明顯示
2. 呼叫對話方塊的方式：
   - 按下（開始）軟鍵
3. 以動態方式重新組織對話方塊：
   - 編輯與刪除軟鍵
   - 定義與設計變數欄位
   - 插入、交換及刪除顯示文字（語言相關或語言中立）
   - 插入、交換及刪除圖形
   - 動態建立可執行命令集工件，並將其包含在命令集處理中

4. 起始回應下列動作的操作：
   – 顯示對話方塊
   – 輸入數值（變數）
   – 選擇軟鍵
   – 退出對話方塊
5. 對話方塊之間的資料交換
6. 變數
   – 讀取（NC、PLC 及使用者變數）
   – 寫入（NC、PLC 及使用者變數）
   – 結合數學、比較或邏輯運算子
7. 執行功能：
   – 子程式
   – 檔案功能
   – PI 服務
8. 根據使用者群組應用的保護等級

指令碼語言的有效元件（標籤）說明於「XML 標籤」(頁 45)一節。

說明

下列「設定的基本原則」一節並不作為 XML（可擴展標示語言）的綜合說明。請參考相關的專業著作以取得進一步資訊。

3.2 設定的基本原則

設定檔

新使用者介面的說明儲存於設定檔案。這些設定檔會自動轉譯，然後顯示在畫面上。設定檔並非儲存在隨附軟體內，必須先由使用者設定並載入。

XML 編輯器或其他形式的文字編輯器可以用來產生設定檔。

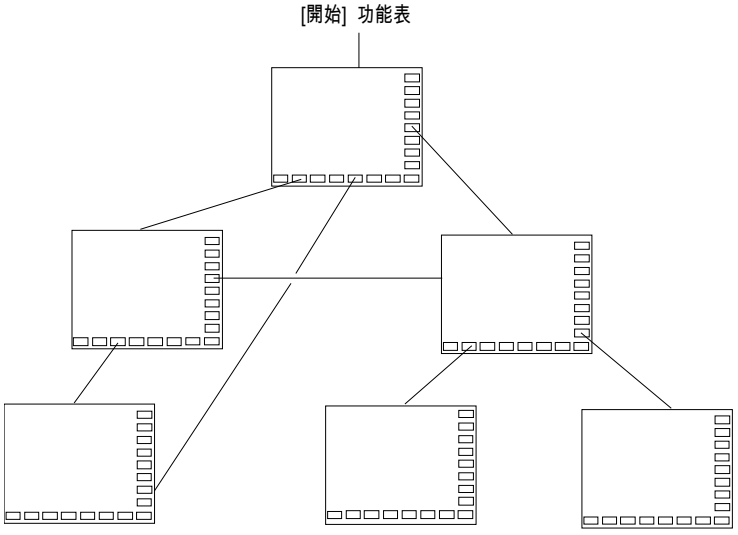
說明

檔案名稱只能包含小寫字母。

樹狀功能表原則

一些互相連結的對話方塊形成樹狀功能表。若可以從一個對話方塊切換到另一個，則存在一個連結。可以利用此對話方塊中新定義的橫向/縱向軟鍵，呼叫前一個或是任何其他對話方塊。

已設定的開始軟鍵可以用來在開始功能表後面建立進一步的樹狀功能表：



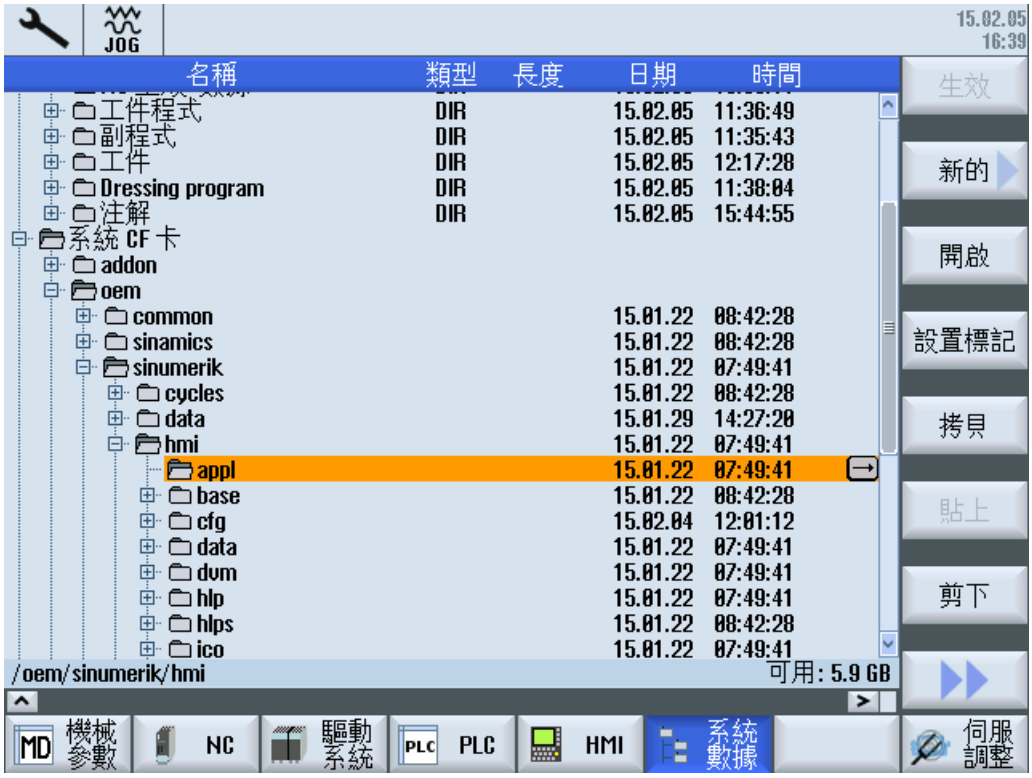
圖像 3-1　使用者對話方塊的樹狀功能表

開始功能表

開始功能表在「xmldial.xml」檔案中的定義名稱為「main」。開始功能表是用來初始化您自己的操作程序。

您可以利用「main」功能表來連結自己的對話方塊或額外的軟鍵列，將其載入並用來執行額外的動作。

下圖顯示控制器上的製造商資料夾「System CF-Card/oem/sinumerik/hmi」。



圖像 3-2　OEM 目錄

需要以下控制器上的 OEM 目錄「System CF-Card/oem/sinumerik/hmi」中的檔案，來設定使用者對話方塊：

表格 3-1　用來設定的檔案

檔案類型	**檔案名稱**	**意義**	**操作區的儲存位置：設定 > 系統資料**
指令碼檔案	「xmldial.xml」	這個指令碼檔案使用 XML 標籤，以控制 SINUMERIK Operate 中設定的軟鍵功能表與對話方塊格式如何顯示於 <CUSTOM> 操作區。	OEM 目錄 > 應用程式的「appl」子目錄
文字檔案	「oem_xml_screens_xxx.ts」	客戶操作區的標準文字檔案 這個文字檔案包含個別語言的功能表與對話方塊格式使用的文字。	OEM 目錄 > 所要語言的「lng」子目錄

檔案類型	檔案名稱	意義	操作區的儲存位置：設定 > 系統資料
點陣圖	---	控制系統支援 BMP 及 PNG 格式。	OEM 目錄 > 子目錄「ico」
XML 檔案與「INCLUDE」XML 標籤插入「xmldial.xml」控制檔案。	例如「machine_settings.xml」	這些檔案同時包含程式化的指令，以顯示 SINUMERIK Operate 的對話方塊格式及參數。	OEM 目錄 > 應用程式的「appl」子目錄

Easy XML 連接點

可以透過下列連接點來使用 Easy XML 指令碼：

Hardkey/Softkey	連接點
客戶	標準指令碼檔案名稱「xmldial.xml」

軟鍵操作功能表	標準指令碼檔案名稱「xmldial.xml」
	在檔案「slamconfig.ini」中啟動
	
	範例：
	`[CustomXML]`
	`TextId=SL_AM_CUSTOM`
	`SidescreenTextId=SL_SIDESCREEN_CUSTOM`
	`TextFile=slam`
	`TextContext=SlAmAreaMenu`
	`SoftkeyPosition=12`
	`Visible=true`

功能表使用者	標準指令碼檔案名稱「menu_user.xml」
功能表功能	標準指令碼檔案名稱「menu_function.xml」

Hardkey/Softkey	**連接點**

PLC 硬鍵	關於按鍵說明的指令碼檔案名稱。
	
	範例：
	`KEY50.0 = area:=CustomXML, dialog:=SlEECustomDialog, cmdline:="-conf slagmdialog.hmi -mainModule restore.xml -entry cmc2"`
	`KEY51.0 = area:=CustomXML, dialog:=SlEECustomDialog, cmdline:="-conf slagmdialog.hmi -mainModule activate.xml -entry cmc1"`

MMC 命令	關於 NC 命令說明的指令碼檔案名稱。 **範例：** `MMC("POPUPDLG, PICTURE_ON, TEST.XML, cmd1","A)`

工作區的保留軟鍵	關於功能表說明的指令碼檔案名稱。
	
	範例：
	`<MENU name="DgGlobalHu">`
	`<ETCLEVEL id="0">`
	`<SOFTKEYGROUP name="GroupEtc">`
	`<SOFTKEY position="7">`
	`<PROPERTY name="textID" type="QString">DG_SK</PROPERTY>`
	`<PROPERTY name="translationContext" type="QString">EASY_XML</PROPERTY>`
	`<FUNCTION args="-area CustomXML -dialog SlEECustomDialog -mainModule dg.xml -entry main" name="switchToDialog"/>`
	`</SOFTKEY>`
	`</SOFTKEYGROUP>`
	`</ETCLEVEL>`
	`</MENU>`

側邊螢幕	關於檔案「slsidescreen.ini」側邊螢幕說明的指令碼檔案名稱。
簡易延伸軟鍵	標準指令碼檔案名稱「agm.xml」

快速選擇鍵

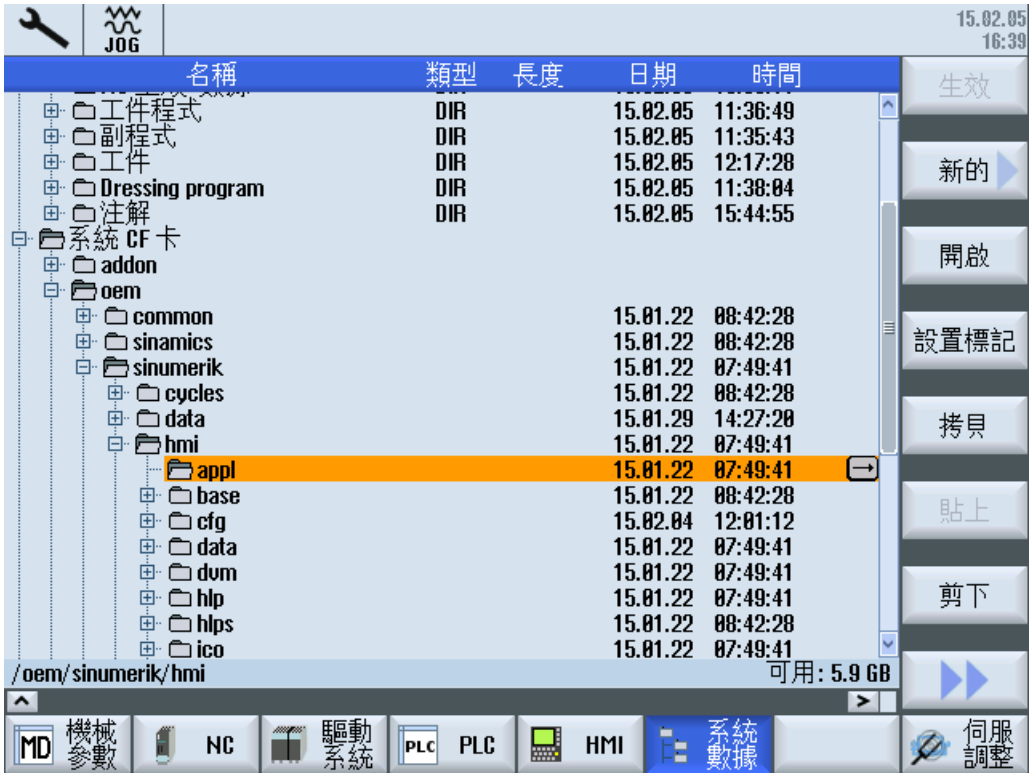
PPU 操作面板正面的快速選擇鍵 <MENU USER> 和 <MENU FUNCTION> 也可指派至任何使用者對話方塊。命令集檔名稱「menu_user.xml」和「menu_function.xml」皆可供使用。

使用者可從工具箱複製命令集檔到製造商目錄，或可自行採用這些名稱來建立檔案。「main」名稱定義為輸入選單。

3.3 設定檔

簡介

下圖顯示控制器上的製造商資料夾「System CF-Card/oem/sinumerik/hmi」。



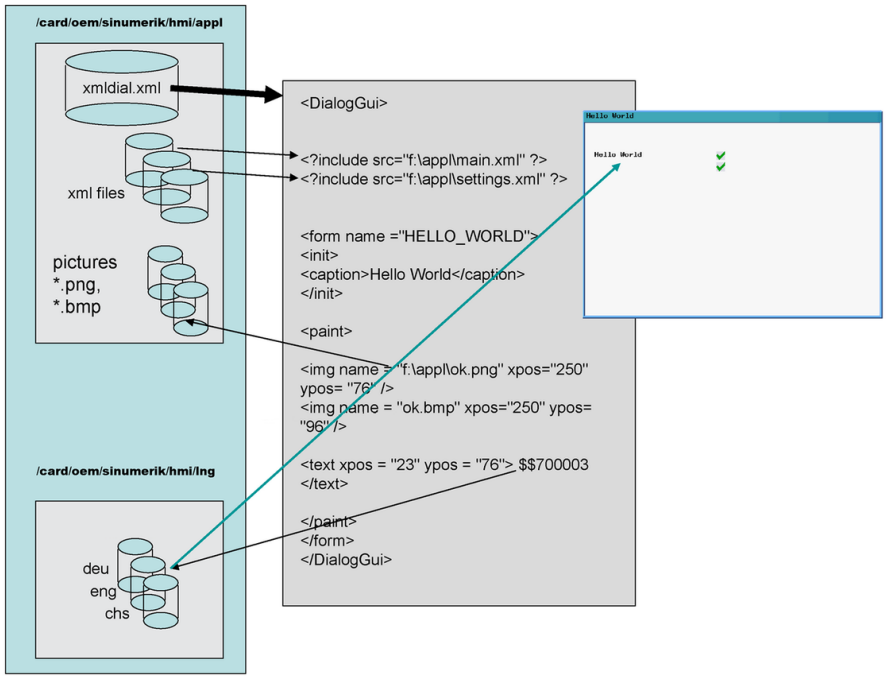
圖像 3-3　製造商資料夾

需要以下控制器上的製造商資料夾「System CF-Card/oem/sinumerik/hmi」中的檔案，來設定使用者對話方塊：

表格 3-2　用來設定的檔案

檔案類型	**檔案名稱**	**含義**	**操作區的儲存位置：設定 > 系統資料**
指令碼檔案	「xmldial.xml」	這個指令碼檔案使用 XML 標籤，以控制 SINUMERIK Operate 中設定的軟鍵功能表與對話方塊形式如何顯示於 <CUSTOM> 操作區。	製造商資料夾 > 應用程式的「appl」子目錄
文字檔案	「oem_xml_screens_xxx.ts」	這個文字檔案包含個別語言的功能表與對話方塊形式使用的文字。	製造商資料夾 > 語言的「lng」子目錄
點陣圖		控制系統支援 BMP 及 PNG 格式。	製造商資料夾 >「ico」子目錄 點陣圖儲存於子目錄中以控制畫面解析度。 注意事項：若點陣圖的路徑已指定，檔案可以直接儲存在這個目錄。
XML 檔案與「INCLUDE」XML 標籤插入「xmldial.xml」控制檔案。	例如： 「machine_settings.xml」	這些檔案同時包含程式化的指令，以顯示 SINUMERIK Operate 的對話方塊形式及參數。	製造商資料夾 > 應用程式的「appl」子目錄

使用者對話方塊設定檔案之間的相依性



圖像 3-4　相依性

載入設定

如先前的「用來設定的檔案」表格內「操作區的儲存位置」欄所述，產生的檔案必須複製到製造商資料夾內適當的子目錄。

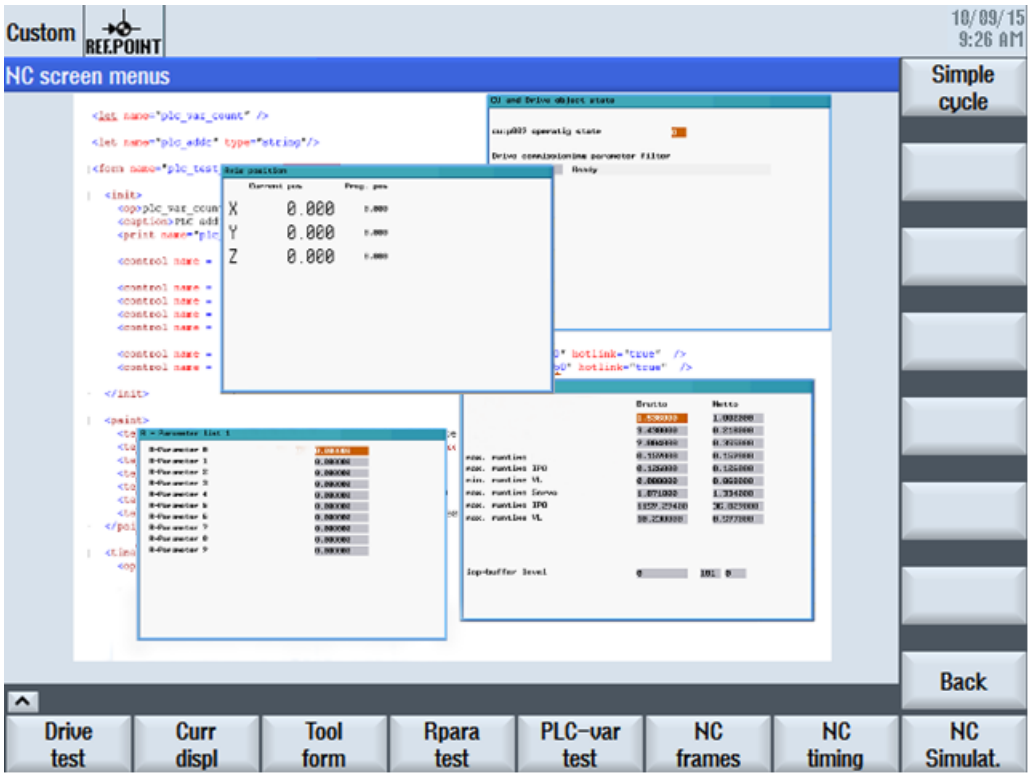
說明

一旦應用程式的子目錄中有「xmldial.xml」指令碼檔案，使用者即可在 <CUSTOM> 操作區中開始這個使用者對話方塊。

最初的複製程序完成後，控制系統必須經由「Normal power-up」重置。

SINUMERIK Operate 的使用者對話方塊範例

呼叫 <CUSTOM> 操作區域時，會顯示已設定的軟鍵功能表。讓使用者可以操作已設定的對話方塊形式。



圖像 3-5 <CUSTOM> 操作區的使用者對話方塊範例

工具箱提供更多的範例。

3.4 設定檔的結構

概觀

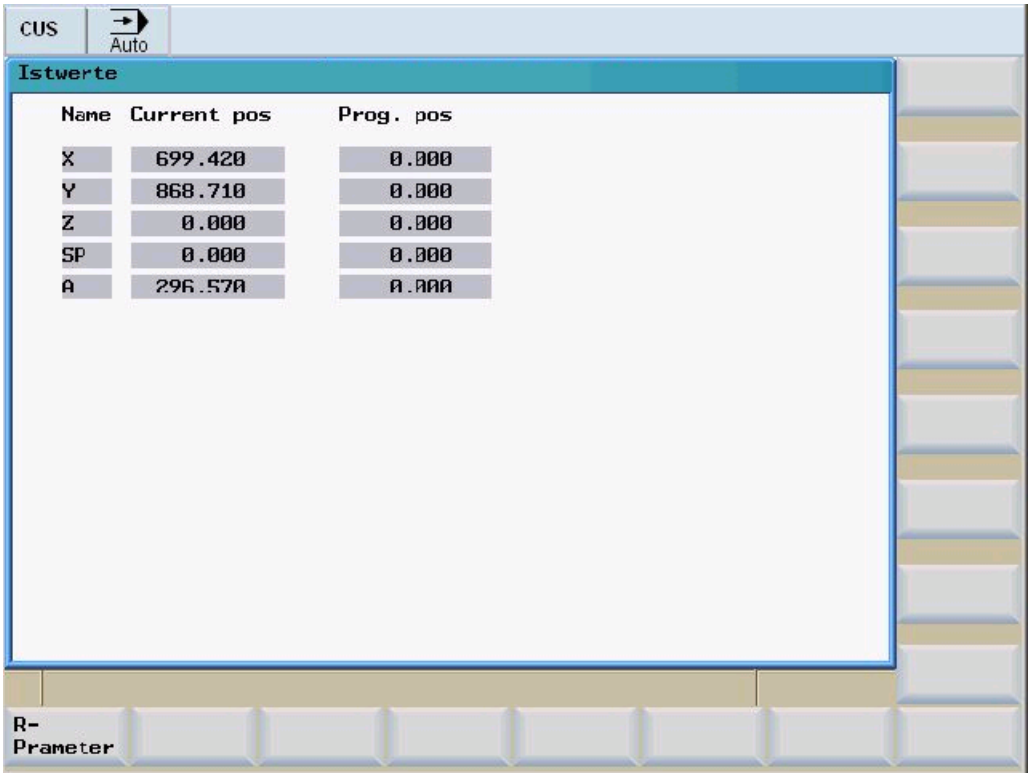
設定檔由下列元素組成：

- 「main」開始功能表與開始軟鍵的說明
- 對話方塊的定義
- 變數的定義
- 單節的說明
- 軟鍵列的定義

下列的範例顯示「xmldial.xml」檔案的 XML 命令集與相關截圖。

此指令集包含對話方塊，用以顯示實際值、剩餘距離、及 R 參數表。

對話方塊格式章節「實際值」



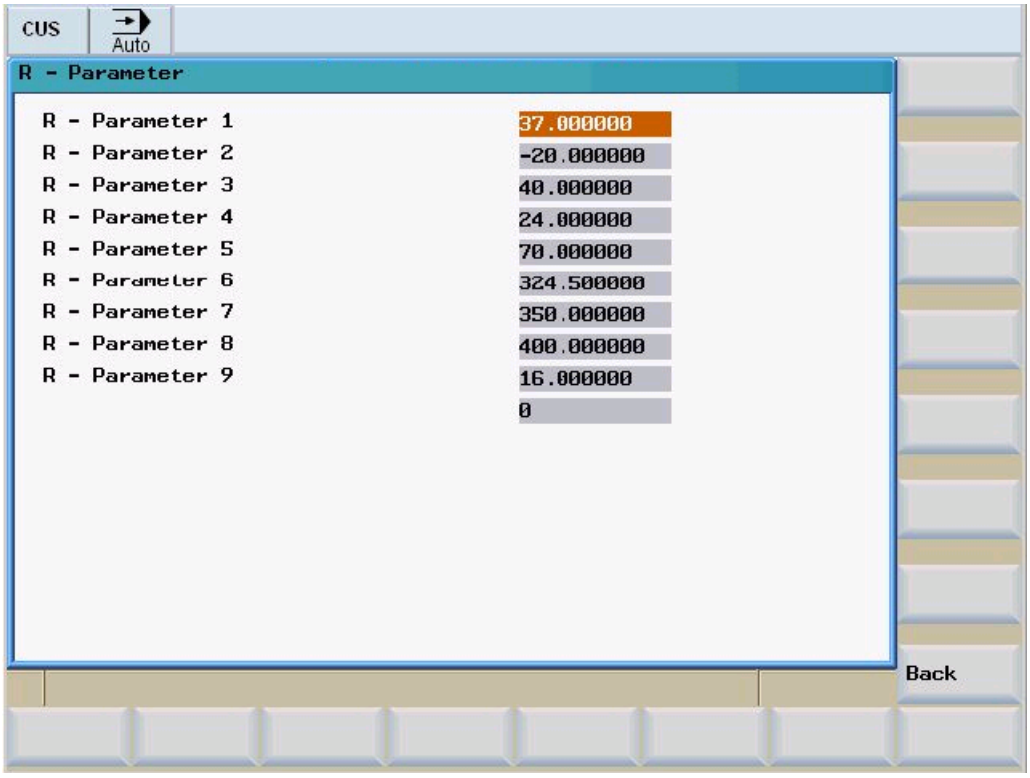
xmldial.xml

```
<DialogGui>

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

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


</menu>


<form name = "CURRENT_DISPLAY">

<init>
<caption>Istwerte</caption>

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

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

xmldial.xml

```
<control name = "edit5" xpos = "80" ypos = "136" refvar="nck/Channel/
GeometricAxis/actProgPos[4]" hotlink="true" fieldtype="readonly"
format="%9.3f" time="super fast"/>

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

</init>

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

</paint>
</form>

.....
```

對話方塊格式章節「R 參數」



xmldial.xml

```
.....

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

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

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

</menu>

<form name = "R-Parameter">

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

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

<paint>
<text xpos = "23" ypos = "34">R - Parameter 1</text>
<text xpos = "23" ypos = "54">R - Parameter 2</text>
<text xpos = "23" ypos = "74">R - Parameter 3</text>
```

xmldial.xml

```
<text xpos = "23" ypos = "94">R - Parameter 4</text>
<text xpos = "23" ypos = "114">R - Parameter 5</text>
<text xpos = "23" ypos = "134">R - Parameter 6</text>
<text xpos = "23" ypos = "154">R - Parameter 7</text>
<text xpos = "23" ypos = "174">R - Parameter 8</text>
<text xpos = "23" ypos = "194">R - Parameter 9</text>
</paint>
</form>

</DialogGui>
```

3.5 語言相依性

語言相關文字用於：

- 軟鍵符號
- 標題
- 說明文字
- 任何其他文字

語言相關文字儲存在文字檔案。

說明

使用這些文字檔案時，您必須執行下列步驟：

- 使文字檔案可用於所需的語言中。
- 將文字檔案傳輸至控制系統的相關語言目錄。

3.6 XML 診斷

系統提供 Easy XML 診斷功能，可用以診斷及查找指令碼錯誤。

使用診斷功能檢查 XML 語法，並且執行完畢所有屬於專案的指令碼。若要執行這項操作，則需同時檢查函數、功能表、格式及變數是否存在及是否有效。列出找到的錯誤。

Easy XML 診斷功能可以透過 **DialogGui** 標籤屬性或顯示機械資料予以啟動。

啟動 Easy XML 診斷

範例：

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

或

顯示機械資料：

MD9113 $MM_EASY_XML_DIAGNOSE	Easy XML 指令碼的診斷及改正說明
= 0	無診斷啟用中
= 1	語法檢查啟用中
= 0x100	訊息也會另外儲存在 error_log.txt 檔案中。

使用追蹤輸出進行診斷

追蹤輸出可以使用標籤**除錯**整合至命令集中。只有在標籤 **DialogGui** 中數值為 **on** 的屬性 **debug_msg** 時，才儲存資料。

軟鍵功能總覽

對話方塊	軟鍵	功能
主功能表「EasyXML 診斷」	開始指令碼	直接開始已下載的程式。
	開始檢查	正開始語法檢查。可以看到所有累計訊息。更新檢查可供使用。

對話方塊	軟鍵	功能
「錯誤」及「警告」對話方塊	錯誤	結果會根據錯誤及警告來加以顯示、排序。
	警告	
	前往錯誤	若找到錯誤或警告，軟鍵即顯示。 軟鍵會開啟錯誤清單中已標示的 XML 檔案以供編輯。游標會自動置放在錯誤列上。
	檢查結果	可以看到所有累計訊息。
「文件」對話方塊	儲存	正儲存 XML 檔案中的變更。
	取消	XML 檔案關閉且無變更。再次顯示檢查結果。

3.7 XML 識別碼

3.7.1 一般結構

對話方塊設定的指令碼檔案的結構與指令

所有對話方塊設定必須儲存在 **DialogGui** 標籤。

```
<DialogGui>
…
</DialogGui>
```

範例：

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

</DialogGui>
```

指令

此語言提供下列指令，用以執行條件指令與迴圈控制：

- For 迴圈
- WHILE 迴圈
- Do while 迴圈
- 條件處理
- 切換與事件指令
- 對話方塊格式的操作者控制
- 軟鍵說明
- 定義變數

指令的詳細說明可以在「指令/識別碼說明 (頁 46)」一節中取得。

3.7.2 指令/識別碼說明

下列 **XML 標籤**已定義，用來建立對話方塊、功能表、及執行程式序列：

說明

引號 "<...>" 中的屬性值應以目前使用的表示式取代。

範例：

<DATA_LIST action="read/write/append" id="<list name>">
的程式設計如下：

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

在複製多行 XML 標籤及範例時，必須注意標記不得以不需要的換行符分隔之。

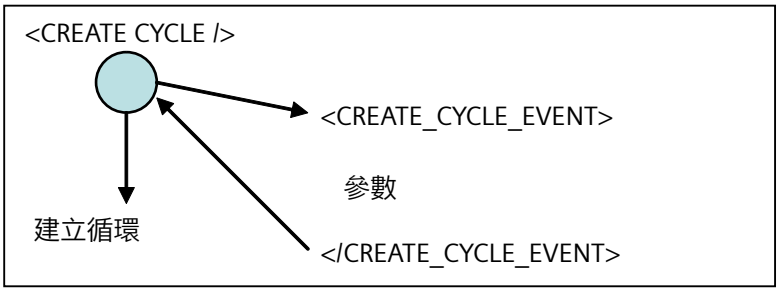
標籤識別碼	意義
BREAK	迴圈的條件取消。

CONDITION	必須在一行中編程標籤，或者在使用多行標記的情況下，指定的條件必須與標籤名稱分開。
	
	範例：
	`<if>`
	`<condition>test <= 2</condition>`
	`...`
	
	或
	
	`<if>`
	`<condition>`
	`test <= 2`
	`</condition>`
	`...`

CONTROL	此標籤用來產生控制元素。說明請參閱「建立軟鍵功能表與對話方塊形式 (頁 85)」一章。

標籤識別碼	意義
CONTROL_RESET	此標籤使一個或多個控制元件重啟。 **語法：** `<CONTROL_RESET resetnc="TRUE" />` **屬性：** • **RESETNC = "TRUE"** 重新啟動 NC 元件 • **RESETDRIVE = "TRUE"** 重新啟動驅動元件。

標籤識別碼	意義

CREATE_CYCLE_EVENT	如果語法分析器開始處理標籤 CREATE_CYCLE，最初會將訊息 <CREATE_CYCLE_EVENT> 傳送至啟用的格式。在語法分析器從參數清單及產生規定產生 NC 操作前，此訊息可以用來準備循環參數。
	 
	
	語法：
	`<CREATE_CYCLE_EVENT>`
	…
	…
	`</CREATE_CYCLE_EVENT>`
	範例：
	`<SOFTKEY_OK>`
	…
	`<CREATE_CYCLE />`
	…
	…
	`<SOFTKEY_OK>`
	…
	…
	`<FORM>`
	`<NC_INSTRUCTION>MY_CYCLE($P1, $P2)</ NC_INSTRUCTION >`
	…
	…
	`<CREATE_CYCLE_EVENT>`
	`<type_cast name="P1" type="int"/>`
	`<OP> P1 = P1 * 150 </OP>`
	`</CREATE_CYCLE_EVENT>`
	…
	…
	`</FORM>`

標籤識別碼	意義
DATA	此標籤允許寫入到 NC、PLC、GUD 及驅動資料。 「元件定址 (頁 157)」一章描述位址如何形成。 **屬性：** • **name** 變數位址 **標籤值：** 標籤數值通常是一個常數、**變數，或是術語**。 **語法：** <DATA name="<variable name>"> value </DATA> <DATA name="<variable name>"> <term> </DATA> **範例：** <DATA name = "plc/mb170"> 1 </DATA> … <LET name = "tempVar"> 7 </LET> <!– 區域變數的內容 "tempVar" 將寫入位元記憶體位元組 170 → <DATA name = "plc/mb170">$tempVar</DATA>
DATA 續	**範例：** 術語的計算。將結果指派給指定的變數。 <let name="a" >#f</let> <let name="b" >7</let> <data name = "b"> a & b</data> <let name="c" type="string"></let> <data name = "c"> _T" test" + _T" and test"</data>

標籤識別碼	意義

DATA_LIST	此標籤使表列的驅動器與機械資料進行儲存或還原。
	位址資訊表列於行中。「元件定址 (頁 157)」一章描述位址如何形成。
	可以建立最多 20 個暫存資料清單。
	
	屬性：
	• **action**
	read – 表列變數的值已儲存於暫存記憶體
	append – 表列變數的值已加到現存的清單中
	write – 備份的值已複製到相關的機械資料中
	• **id**
	識別碼用來識別暫存記憶體
	語法：
	`<DATA_LIST action="<read/write/append>"  id="<list name>">`
	`NC/PLC 位址編譯`
	`</DATA_LIST>`
	
	範例：
	`<DATA_LIST action ="read" id="<name>">`
	`nck/channel/parameter/r[2]`
	`nck/channel/parameter/r[3]`
	`nck/channel/parameter/r[4]`
	`$MN_USER_DATA_INT[0]`
	`…`
	`</ DATA_LIST>`
	`<DATA_LIST action ="write" id="<name>" />`

DIALOGGUI	該標籤代表 Easy XML 功能的根標籤。包含的所有指令均使用 Easy XML 解析器進行編輯。 **屬性：** 可以選擇指定列出的屬性，以啟用中央功能： • **diagnose** - 數值 **TRUE** 啟動 XML 診斷 • **debug_msg** - 數值「on」- 儲存追蹤訊息 **更多資訊**請參閱章節「XML 診斷 (頁 43)」 • **textfile** - 使用文字檔案的名稱 • **textcontext** - 僅與屬性文字檔案結合使用的文字內容為有效 • **textfilelist** - 允許載入帶有不同文字檔案的清單 **更多資訊**請參閱章節「專案專屬的文字檔 (頁 285)」

標籤識別碼	意義
DIALOGGUI 續	• **restoresession** - 數值 **true** 如果再次選擇 Easy XML 專案，則解析器會開啟上次選擇的功能表及最後一次選擇的格式。重新啟動 HMI 之前，將持續保留此行為。 **更多資訊**請參閱章節「還原工作階段 (頁 293)」
DEBUG_MSG	這個屬性控制追蹤輸出的儲存。相關說明請參閱標籤 DEBUG 底下的章節「建立軟鍵功能表與對話方塊形式 (頁 85)」。 若是指定了數值 **on**，則解析器會將所有輸出儲存在一個檔案中。這會導致指令集執行時變慢。預設值是設定為 **off**。

DO_WHILE	Do while 迴圈
	`DO`
	`  指令`
	`WHILE (測試)`
	
	語法：
	`<DO_WHILE>`
	`指令`
	`…`
	`<CONDITION>…</CONDITION>`
	`</DO_WHILE>`
	Do While 迴圈包含指令單節與一個條件。先執行指令單節內的程式碼，再套用條件。若該條件為 true，本功能會再次執行該程式碼區段。一直重複執行到該條件為 false。
	
	範例：
	`<DO_WHILE>`
	`<DATA name = "PLC/qb11"> 15 </DATA>`
	`<CONDITIION> "plc/ib9" == 0 </CONDITION>`
	`</DO_WHILE>`

標籤識別碼	意義

DYNAMIC_INCLUDE	標籤包含 XML 指令碼檔案。
	與 INCLUDE 標籤不同的是，執行對應操作時只會先讀入。
	在大型專案中使用標籤可以減少客戶區及/或循環支援的載入時間。此外，資源的平均層級也能降低，因為在工作階段時不會呼叫所有的對話方塊。
	
	語法：
	<DYNAMIC_INCLUDE src="path name"/>
	
	範例：
	`<SOFTKEY POSITION="3">`
	`<CAPTION>MY_MENU</CAPTION>`
	`<DYNAMIC_INCLUDE src="f:\appl\sk3_script.xml"/>`
	`<NAVIGATION>MY_MENU</NAVIGATION>`
	</SOFTKEY>

ELSE	未符合條件時的指令（**IF**、**THEN**、**ELSE**）

標籤識別碼	意義
FOR	For 迴圈 for （初始化；測試；繼續）指令 **語法：** <FOR> <INIT>…</INIT> <CONDITION>…</CONDITION> <INCREMENT>…</INCREMENT> 指令 … </FOR> For 迴圈依下列方式執行： 1. 表示式**初始化** (INIT) 的評估。 2. 表示式**測試** (CONDITION) 的評估作為布林表示式。 若數值為 false，結束 For 迴圈。 3. 執行以下指令。 4. 表示式**延續** (INCREMENT) 的評估。 5. 繼續進行 2。 所有在 INIT、CONDITION 及 INCREMENT 分支內的變數，必須在 For 迴圈之外宣告及初始化。 **範例：** <LET name = "count">0</LET> <FOR> <INIT> <OP> count = 0</OP> </INIT> <CONDITION> count <= 7 </CONDITION> <INCREMENT> <OP> count = count + 1 </OP> </INCREMENT> <OP> "plc/qb10" = 1+ count </OP> </FOR>
FORM	此標籤包含使用者對話方塊的說明。說明請參閱「建立軟鍵功能表與對話方塊形式 (頁 85)」一章。
HMI_RESET	標籤會觸發 HMI 重新啟動。 解譯會在此指令之後停止。

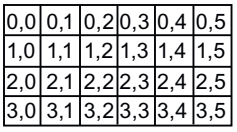
標籤識別碼	意義

IF	條件指令 (**IF, THEN, ELSE**)
	THEN 及 ELSE 標籤包含在 IF 標籤內。
	在 CONDITION 標籤內執行的條件接在 IF 標籤後。針對指令的進一步處理，視操作的結果而定。若函數結果為 true，則執行 THEN 分支並略過 ELSE 分支。若函數結果為 false，語法分析器執行 ELSE 分支。
	
	語法：
	`<IF>`
	`<CONDITION> 條件 != 7 </CONDITION>`
	`<THEN>`
	個案說明：符合條件
	`</THEN>`
	`<ELSE>`
	個案說明：不符合條件
	`</ELSE>`
	`</IF>`
	
	範例：
	`<IF>`
	`<CONDITION> "plc/mb170" != 7 </CONDITION>`
	`<THEN>`
	`<OP> "plc/mb170" = 7 </OP>`
	…
	`</THEN>`
	`<ELSE>`
	`<OP> "plc/mb170" = 2 </OP>`
	…
	`</ELSE>`
	`</IF>`

標籤識別碼	意義

IF 續	如果條件中僅使用本地變數，則可以在 IF 標籤中將條件指定為屬性。
	
	範例：
	`<let name="var1">1</let>`
	`<if condition="var == 1">`
	`<then>`
	`...`
	`...`
	`</then>`
	`</if>`
	
	附註：
	未與參考變數耦合的控制將指派為整數資料類型。
	例外：
	控制、類型 **imagebox** 及 **multiline** 則指派為資料類型的**字串**。

INCLUDE	指令包含 XML 說明。 （另請參閱本表格的 DYNAMIC_INCLUDE ） **屬性：** • **src** 包含路徑名稱。 **語法：** `<?INCLUDE src="<Path name>" ?>`

標籤識別碼	意義
LET	本指令在指定的名稱下建立區域變數。 **欄位：** 使用屬性 **dim**（維度）可以建立一維或二維欄位。欄位索引定出個別欄位元件的位址。 對於二維欄位而言，會先指定行索引，接著是列索引。 • 一維欄位： 索引 0 至 4  • 二維欄位： 索引行 0 至 3，索引列 0 至 5  **屬性：** • **name** 變數名稱 • **type** 變數類型可以是整數 (INT)、無符號整數 (UINT)、雙精度浮點數 (DOUBLE)、浮點數 (FLOAT)、字串 (STRING) 或 STRUCT。也可以使用 **typedef** 所建立的結構，作為變數類型（參閱標籤識別碼 TYPEDEF）。若無指定類型指令，系統會建立一個整數變數。 `<LET name = "VAR1" type = "INT" />` • **permanent** 若屬性設定為 **true**，變數值會永久儲存。本屬性僅於全域變數有效。 • **poweroff** 若是屬性的數值為 **true**，在關閉控制之前，仍會保留這些值。 **更多資訊**請參閱章節「還原工作階段 (頁 293)」 • **dim** 下列的欄位元件編號必須指定。二維欄位的第一維度與第二維度是以逗號分隔。 欄位元件的存取係透過欄位索引，在變數名稱後以方括號指定。 name[index] 或 name[row,column] – 一維欄位：dim="<元件數>"

標籤識別碼	意義
	– 二維欄位：dim="<行數>,<列數>" 未初始化的欄位元件預先指派為「0」。

標籤識別碼	意義

LET 續	**範例：**
	一維欄位：
	`<let name="array" dim="10"></let>`
	二維欄位：
	`<let name="list_string" dim="10,3" type="string"></let>`
	
	預先指派：
	變數可以用數值初始化。
	`<LET name = "VAR1" type = "INT"> 10 </LET>`
	
	若包含 NC 或 PLC 變數的值儲存在區域變數，指派操作會自動將格式轉換成已載入變數的格式。
	• 針對字串變數的預先指派：
	若格式化的文字轉換成值，超過一行的文字可以指派為字串變數。若一行的結尾為 line feed <LF>（換行）則必須在行末加上字元 "\\n"。
	`<LET name = "text" type = "string"> F4000 G94\\n`
	`G1 X20\\n`
	`Z50\\n`
	`M2\\n`
	`</LET>>`
	
	欄位（Arrays）：
	`<let name="list" dim="10,3">`
	`  {1,2,3},`
	`  {1,20}`
	`</let>`
	
	`<let name="list_string" dim="10,3" type="string">`
	`  {"text 10","text 11"},`
	`  {"text 20","text 21"}`
	`</let>`
	
	指派：
	由機械資料或子程式組成的值，可以利用指派操作「＝」指派為變數。
	變數在較高層級的 XML 區塊結束之前維持有效。
	若要使變數變成全域有效，應在其前面加上 **DialogGUi** 標籤。
	對話方塊必須注意下列事項：
	• 訊息處理會開啟對應的標籤。
	• 訊息執行以後關閉標籤。
	• 關閉時刪除標籤內的所有變數。

標籤識別碼	意義

LET 續	**變數類型結構**：
	變數類型包含可使用結構名稱定址的變數組成，而一個結構可包含所有變數類型及多個結構。
	結構內的變數以「element」標籤做宣告，標籤屬性與初始化對應於 let 指令的屬性與初始化。
	
	`<let name="name" type="struct">`
	`<element name="<Name>" type="<Variable type>" />`
	…
	…
	…
	`</let>`
	
	可以為結構元素指派一個預設值，該預設值在建立變數時將當作初始值。
	更多資訊請參閱章節「結構的初始化」。
	若是在建立變數時指定了初始值，則該值將被覆蓋。

標籤識別碼	意義

LET 續	結構變數的存取是經由結構名稱與變數名稱，而這兩個名稱以一個點算子隔開。
	
	語法：
	
	`<op>`
	`Structure_name.variable_name = value;`
	`</op>`
	
	範例：
	`<let name="info" type="struct">`
	`  <element name="id" type="int" />`
	`  <element name="name" type="string" />`
	`  <element name="phone" type="string" />`
	`</let>`
	
	`<op>`
	`  info.id = 1;`
	`  info.name = _T"my name";`
	` info.phone = _T"0034 45634";`
	`</op>`

標籤識別碼	意義

LET 續	**結構的初始化：**
	藉由為每個結構元素指定初始值以建立變數時，結構即可進行初始化。在結構陣列中，每個結構都必須以大括號與其他結構隔開。
	
	範例：
	`<typedef name="StructRect" type="struct" >`
	`  <element name="left" type="int">22</element>`
	`  <element name="top" type="int">122</element>`
	`  <element name="right" type="int">220</element>`
	`  <element name="bottom" type="int">56</element>`
	`</typedef>`
	`<let name="rect" type="StructRect" >`
	透過建立 **rect** 變數，結構元素將使用以下數值初始化：
	`rect.left= 22`
	`rect.top = 122`
	`rect.right = 220`
	`rect.bottom = 56`
	
	如果將初始值指派給結構，則它們將覆蓋預設值。
	`<let name="rect" type="StructRect" >`
	`  {11,12}`
	`</let>`
	透過建立 rect 變數，結構元素將使用以下數值初始化：
	`rect.left= 11`
	`rect.top = 12`
	`rect.right = 220`
	`rect.bottom = 56`

標籤識別碼	意義
LET 續	**巢狀結構：** 結構是使用者定義的變數類型之變數，因此可以用作結構中的結構元素。 元素根據所述的規則初始化。 **範例：** `<typedef name="StructRectRect" type="struct" >   <element name="r1" type="StructRect">77, 99, 232, 23</element>   <element name="r2" type="StructRect">77, 88, 45, 12</element> </typedef>` `<let name="rect_rect_array" type="StructRectRect" >` 透過建立 **rect_rect_array** 變數，結構元素將使用以下數值初始化： `r1.left= 77` `r1.top = 99` `r1.right = 232` `r1.bottom = 23` `r2.left= 77` `r2.top = 88` `r2.right = 45` `r2.bottom = 12` `<let name="rect_rect_array1" type="StructRectRect" >` `  {{11,12,13,14},{111,188,199,114}}` `</let>` 透過建立 **rect_rect_array** 變數，結構元素將使用以下數值初始化： `r1.left= 11` `r1.top = 12` `r1.right = 13` `r1.bottom = 14` `r2.left= 111` `r2.top = 188` `r2.right = 199` `r2.bottom = 114`
LET 續	**全域字串變數的初始化：** 如果使用文字識別碼來初始化全域字串變數，則在標準檔案 **oem_xml_screens_xxx.ts** 或是 **DialogGUI** 標籤所指定的文字檔案中，應使用所要替換的識別碼及文字。載入應用程式時，啟用的語言文字會替換文字識別碼。如果在應用程式處於啟用狀態時執行語言更改，則不會對全域字串變數進行新的初始化。文字可以在 **LANGUAGE_CHANGED** 訊息中交換。

標籤識別碼	意義
LOCK_OPERATING_AREA	操作區切換已鎖定。 操作區切換鎖定係使用 **UNLOCK_OPERATING_AREA** 標籤予以解除。 **語法：** `<LOCK_OPERATING_AREA />`
MSG	操作員元件展現出顯示在標籤內的訊息。 若警報編號已使用，對話方塊會顯示以該編號儲存的文字。 **範例：** `<MSG text ="my message" />`

MSGBOX	本指令打開訊息方塊，該訊息方塊的傳回值可以用在分支上。
	
	語法：
	`<MSGBOX text="<Message>" caption="<caption>" retvalue="<variable name>" type="<button type>" />`
	
	屬性：
	• **text**
	文字
	• **caption**
	頁首
	• **retvalue**
	複製傳回值的變數名稱：
	1 – OK
	0 – CANCEL
	• **type**
	確認選項：
	"BTN_OK"
	"BTN_CANCEL"
	"BTN_OKCANCEL"
	若警報編號用在 **"text"** 或 **"caption"** 屬性中，訊息方塊會以該編號所儲存的文字顯示。
	
	範例：
	`<MSGBOX text="測試訊息" caption="資訊" retvalue="result" type="BTN_OK" />`

標籤識別碼	意義

OP	標籤會執行指定的操作。
	可以執行「運算子 (頁 83)」一章中所列的操作。
	為達到存取 NC、PLC 及驅動資料的目的，完整的變數名稱必須放置在**引號**之中。「元件定址」(頁 157)一章描述位址如何形成。
	`PLC: "PLC/MB170"`
	`NC: "NC/Channel/…"`
	
	範例：
	`<LET name = "tmpVar" type="INT"> </LET>`
	`<OP> tmpVar = "plc/mb170" </OP>`
	`<OP> tmpVar = tmpVar *2 </OP>`
	`<OP> "plc/mb170" = tmpVar </OP>`
	
	操作標籤內可使用超過一個方程式，分號為架構的末端。
	
	範例：
	`<op>`
	`x = x+1;`
	`y = y+1;`
	`</op>`
	
	字元字串處理：
	本操作說明可以處理字元字串，並將結果指派到方程式指定的字串變數。
	識別碼 _T 應置於起始的位置，作為辨識文字表示式的工具。同時允許變數值的格式化操作。識別碼 _F 應置於格式化起始的位置，後面接著格式指令。然後針對該變數指定位址。
	
	範例：
	`<LET name="buffer" type="string"></LET>`
	`…`
	`…`
	`…`
	`<OP> buffer = _T"unformatted value R0= " + "nck/Channel/Parameter/R[0]" + _T" and " + _T"$$85051" + _T" formatted value R1 " + _F %9.3f"nck/Channel/Parameter/R[1]" </OP>`

標籤識別碼	意義

OPERATION	**操作**
	可在方程式內移動指令。
	移動向左「<<」運算子
	<< 函數向左移動位元，您可以直接或透過變數來指定一個值及移動增量的數量，若達到資料格式的限制，位元將移動超過限制，並且不會發出錯誤訊息。
	執行規則
	從左到右執行；「+」和「-」優先順序高於「<<」和「>>」
	
	範例：
	`<op> idx = idx << 2 </op>`
	`<op> idx = 3 + idx << 2 </op>`
	
	移動向右「>>」運算子
	>> 函數向右移動位元，您可以直接或透過變數來指定一個值及移動增量的數量，若達到資料格式的限制，位元將移動超過限制，並且不會發出錯誤訊息。
	執行規則
	從左到右執行；「+」和「-」優先順序高於「<<」和「>>」
	
	範例：
	`<op> idx = idx >> 2 </op>`
	`<op> idx = 3+idx >> 2</op>`

標籤識別碼	意義

PASSWORD	使用此標籤將「EasyExtend」功能的其他單元解除鎖定。
	將指定的字元字串寫入指定的參考變數，而且必須在 PLC 使用者程式中處理。
	
	語法：
	`<PASSWORD refVar ="<variable name>" />`
	
	屬性：
	• **refVar**
	參考變數的名稱
	• **text**
	指定屬性取代預設文字（選擇性）
	
	範例：
	`<PASSWORD refvar="plc/mw107" />`
	
	範例選擇性：
	`<password refVar = "plc/MD108" text="Password" />`

POWER_OFF	提示操作者關閉機台的訊息。本訊息文字永久儲存在系統中。

標籤識別碼	意義

PRINT	本標籤輸出對話行中的文字，或複製該文字到指定的變數。
	若該文字包含格式識別碼，則將變數值插入適當的位置。
	
	語法：
	`<PRINT name="變數名稱" text="text %格式化"> Variable, … </PRINT>`
	`<PRINT text="text %格式化"> Variable, … </PRINT>`
	屬性：
	• **name**
	用來儲存文字的變數名稱（選擇性）
	• **text**
	文字
	
	格式化：
	字元「%」使變數以待格式化的值指定。
	%[旗標] [寬度] [.小數點位數] 類型
	• 旗標：
	定義輸出格式的選配字元：
	– 靠右對齊或靠左對齊（「-」作為靠左對齊）
	– 加入前置零（「0」）
	– 填入空格
	• 寬度：
	本引數定義正數的最小輸出寬度。若欲輸出的值位數小於定義的引數，多餘的空間會填入空格。
	• 小數點位數：
	使用浮點數，此選配的參數定義小數點位數。
	• 類型：
	類型字元定義傳送何種資料格式作為列印指令。這些字元必須指定。
	– d：整數值
	– f：浮點數
	– s：字串

標籤識別碼	意義

PRINT 續	**值：**
	該值將插入變數編號至文字。
	變數類型必須匹配對應的格式化指令類型識別碼，必須使用逗號加以分隔。
	
	範例：
	資訊行中的文字輸出
	`<PRINT text="資訊文字" />`
	含變數格式化的文字輸出
	`<LET name="trun_dir"></LET>`
	`<PRINT text="M%d">trun_dir</PRINT>`
	字串變數中含變數格式化的文字輸出
	`<LET name="trun_dir"></LET>`
	`<LET name="str" type="string" ></LET>`
	`<print name="str" text="M%d ">trun_dir</print>`

PROGRESS_BAR	標籤開啟或關閉進度列。進度列會顯示在應用程式的視窗底下。
	
	語法：
	`<PROGRESS_BAR type="<true/false>"> value </ PROGRESS_BAR>`
	
	屬性：
	• **type =** "TRUE" - 開啟進度列
	• **type** = "FALSE" - 關閉進度列
	• **min**
	（選配）– 最小值
	• **max**
	（選配）– 最大值
	
	值：
	• **Value**
	進度列的百分比位置
	範例：
	`<PROGRESS_BAR type="true" min="0" max="101" >20< /`
	`PROGRESS_BAR>......<PROGRESS_BAR >50< /`
	`PROGRESS_BAR>......<PROGRESS_BAR type="false" >100< /PROGRESS_BAR>`

標籤識別碼	意義

SEND_MESSAGE	標籤傳送兩個參數的訊息給啟用的格式，在標籤訊息中進行處理。
	
	語法：
	`<SEND_MESSAGE>p1, p2</SEND_MESSAGE>`
	
	範例：
	`<SOFTKEY POSITION="3">`
	`  <caption>Set%nParameter</caption>`
	`  <send_message>1, 0</send_message>`
	`</SOFTKEY>`
	
	`<FORM>`
	…
	…
	`<MESSAGE>`
	`<SWITCH>`
	`<CONDITION>$message_par1</CONDITION>`
	`<CASE value="1">`
	…
	…
	`</CASE>`
	`</SWITCH>`
	`</MESSAGE>`
	…
	…
	
	`</FORM>`

標籤識別碼	意義

SHOW_CONTROL	控制器的能見度可以使用標籤加以控制。
	
	語法：
	`<SHOW_CONTROL name="<name>" type="<type>" />`
	
	屬性：
	• **name**
	控制項的名稱
	• **type =** "TRUE" - 控制器變成可見
	• **type** = "FALSE" - 控制器變成不可見（隱藏）
	
	範例：
	`<SHOW_CONTROL name="myEditfield" type="false" />`
	…
	…
	`<SHOW_CONTROL name="myEditfield" type="true" />`

SLEEP	標籤在指定的時間間隔中斷指令碼執行。中斷時間係從傳輸值乘上 50 毫秒的時間基準計算而得。
	
	語法：
	`<SLEEP value="中斷時間" />`
	
	範例：
	等候時間，1.5 秒
	`<SLEEP value="30" />`

STOP	在此點取消解譯。

標籤識別碼	意義

SWITCH	**SWITCH** 指令說明多重選擇。條件評估一次並與一些常數比較。若該表示式與常數匹配，則在 **CASE** 指令中執行指令。
	若無常數與該表示式匹配，則執行 **DEFAULT** 指令。
	
	語法：
	`<SWITCH>`
	`<CONDITION> 值 </CONDITION>`
	`<CASE value="<常數 1>">`
	指令
	…
	`</CASE>`
	`<CASE value="<常數 2>">`
	指令
	…
	`</CASE>`
	`<DEFAULT>`
	指令
	…
	`</DEFAULT>`
	`</SWITCH>`

標籤識別碼	意義

SWITCHTOAREA	SWITCHTOAREA 標籤從客戶區變更為指定的操作區。
	參數以絕對值指定。
	
	語法：
	`<switchToArea name="area" args="argument "/>`
	
	屬性：
	name
	以下名稱針對操作區進行宣告：
	• **AreaMachine** - 機台
	• **AreaParameter** - 參數
	• **AreaProgramEdit** - 編輯器
	• **AreaProgramManager** - 程式管理員
	• **AreaDiagnosis** - 診斷
	• **AreaStartup** - 設定
	args（已保留）
	執行時間引數可傳遞至操作區以啟用對話方塊。
	
	範例：
	`<switchToArea name="AreaMachine" />`

SWITCHTODYNAMICTARGET	如果前一個對話方塊已定義為動態跳躍的目的地，則 SWITCHTODYNAMICTARGET 標籤將啟用此對話方塊並結束指令碼執行。 **語法：** `<switchToDynamicTarget />`
THEN	指令，若已符合條件 (**IF, THEN, ELSE**)

標籤識別碼	意義
TYPE_CAST	標籤轉換區域變數的資料類型。 **語法：** <type_cast name="variable name" type=" new type" /> **屬性：** • **name** 變數名稱 • **type** 新的資料類型已指派至變數。 • **convert** 新的資料類型已指派至變數。變數值也已轉換為新的資料類型。

標籤識別碼	意義

TYPEDEF	資料類型的新識別碼可用這項標籤來定義。這對於結構定義是有幫助的，因為資料類型只須定義一次，然後在 LET 指令中也可使用此資料類型。
	識別碼及類型被預期為屬性。
	語法分析器只支援結構定義的規格。
	
	在類型定義中，以「element」標籤來宣告變數，標籤屬性對應於 let 指令的屬性。
	
	`<typedef name="<identifier>" type="struct">`
	`<element name="<name>" type="<variable type>" />`
	…
	…
	…
	`</typedef>`
	
	定義結束後，識別碼可用做 LET 指令的資料類型。
	
	`<let name="<variable name>" type="<identifier>"></let>`
	
	範例：
	`<typedef name="my_struct" type="struct">`
	`<element name="id" type="int" />`
	`<element name="name" type="string" />`
	`<element name="phone" type="string" />`
	`</typedef>`
	
	`<let name="info" type="my_struct"></let>`
	…
	…
	`<op>`
	`info.id = 1;`
	`info.name = _T"my name";`
	`info.phone= _T"0034 45634";`
	`</op>`

標籤識別碼	意義
TYPEDEF 續	有些預先定義的功能需要結構類型的變數 RECT、POINT 或 SIZE 作為呼叫參數。這些結構已定義於檔案 struct_def.xml。 **更多資訊**請參閱章節「建立檔案 struct_def.xml (頁 155)」 **RECT：** `<typedef name="StructRect" type="struct" > <element name="left" type="int">0</element> <element name="top" type="int">0</element> <element name="right" type="int">0</element> <element name="bottom" type="int">0</element> </typedef>` **POINT：** `<typedef name="StructPoint" type="struct" > <element name="x" type="int">0</element> <element name="y" type="int">0</element> </typedef>` **SIZE：** `<typedef name="StructSize" type="struct" > <element name="width" type="int">0</element> <element name="height" type="int">0</element> </typedef>`
UNLOCK_OPERATING_AREA	操作區切換鎖定解除 **語法：** `<UNLOCK_OPERATING_AREA />`

標籤識別碼	意義

WAITING	本標籤等待元件在 NC 或驅動器重置後進行熱重啟。
	
	屬性：
	• **WAITINGFORNC** = "TRUE" - 系統等候 NC 重新啟動
	• **WAITINGFORDRIVE** = "TRUE" - 系統等候驅動器重新啟動
	
	語法：
	`<WAITING WAITINGFORNC = "TRUE"/>`
	
	範例：
	...
	`<CONTROL_RESET resetnc = "true" resetdrive = "true"/>`
	`<WAITING waitingfornc = "true" waitingfordrive = "true" />`
	...

WHILE	WHILE 迴圈
	`WHILE (Test)`
	`  指令`
	
	語法：
	`<WHILE>`
	`<CONDITION>…</CONDITION>`
	指令
	…
	`</WHILE>`
	條件符合時，While 迴圈會重複執行一連串指令，該連串指令執行前會測試該條件。
	
	範例：
	`<WHILE>`
	`<CONDITION> "plc/ib9" == 0 </CONDITION>`
	`<DATA name = "PLC/qb11"> 15 </DATA>`
	`</WHILE>`

標籤識別碼	意義
XML_PARSER	「XML_PARSER」標籤可用於解析 XML 檔案。 語法分析器解譯 XML 檔案及已定義的回撥函數呼叫。每個回撥函數分別屬於一項預先定義的事件。編程人員可在此函數中處理 XML 資料。 **預先定義的事件：** • 開始文件 語法分析器開啟文件並開始語法分析。 • 結束文件 語法分析器關閉文件。 • 開始元件 語法分析器找到一個元件並建立所有屬性及屬性值的清單。 這些清單會轉送到回撥函數。 • 結束元件 已找到元件的結尾。 • 字元 語法分析器轉發元件的所有字元。 • 錯誤 語法分析器已偵測到語法錯誤。 當一個事件發生，語法分析器會呼叫回撥函數並檢查函數回傳值。如果函數回傳數值「true」，則語法分析器會繼續處理。

標籤識別碼	意義

XML_PARSER 續	介面
	name 屬性值包含 XML 檔案的路徑。
	為了指派事件給回撥函數，必須指定下列特性：
	標準
	startElementHandler
	endElementHandler
	charactersHandler
	選擇性
	errorHandler
	documentHandler
	屬性值定義回撥函數的名稱。
	
	範例：
	`<XML_PARSER name="f:\appl\xml_test.xml">`
	`<!-- standard handler -->`
	`<property startElementHandler="startElementHandler" />`
	`<property endElementHandler="endElementHandler" />`
	`<property charactersHandler="charactersHandler" />`
	`<!-- optional handler -->`
	`<property errorHandler="errorHandler" />`
	`<property documentHandler="documentHandler" />`
	`</XML_PARSER>`

標籤識別碼	意義

XML_PARSER 續	語法分析器也提供變數使回撥函數可以存取事件資料。
	
	startElementHandler：
	函數參數
	tag_name - 標籤名稱
	num - 找到的屬性數字
	系統變數
	$xmlAttribute 字串陣列含 num 指出的元件編號。
	$xmlValue 字串陣列含數字 0 屬性值範圍。
	
	範例：
	`<function_body name="startElementHandler" return="true" parameter="tag_name, num">`
	`<print text="attribute name %s"> $xmlAttribute[0]</print>`
	`<print text="attribute value %s"> $xmlValue[0]</print>`
	`</function_body>`
	
	endElementHandler：
	函數參數
	tag_name - 標籤名稱
	
	範例：
	`<function_body name="endElementHandler" parameter="tag_name">`
	`<print text="name %s"> tag_name </print>`
	`</function_body>`

標籤識別碼	意義

XML_PARSER 續	**charactersHandler：**
	系統變數
	$xmlCharacters 資料字串
	$xmlCharactersStart 永遠為 0
	$xmlCharactersLength 位元組數目
	
	範例：
	`<function_body name="charactersHandler" return="true" >`
	`  <print text="chars %s"> $xmlCharacters </print>`
	`</function_body>`
	
	documentHandler：
	函數參數
	state 1 開始文件、2 結束文件
	
	errorHandler：
	系統變數
	$xmlErrorString 包含無效行（系統變數）
	
	範例：
	`<function_body name="errorHandler" >`
	`  <print text="error %s">$xmlErrorString</print>`
	`</function_body>`
	
	回撥結果：
	$return 若回傳 1 (true)，則語法分析器繼續分析檔案

3.7.3 色彩編碼

顏色屬性使用 HTML 語言的色彩編碼方案。

依據語法，顏色的規格包含「 # 」（hash）字元與六個十六進位數字，每個顏色由兩個數字表示。

R －紅色

G －綠色

B －藍色

#RRGGBB

範例：

color= "#ff0011"

範例顏色：

紅色	綠色	藍色	黃色	白色	黑色
#FF0000	#00FF00	#0000FF	#ffff00	#FFFFFF	#000000

3.7.4 特殊的 XML 語法

若要在一般的 XML 編輯器正確顯示，XML 語法裡有特殊含義的字元必須重寫。

下列字元受到影響：

字元	XML 內的標記
<	&lt；
>	&gt；
&	&amp；
"	&quot；
'	&apos；

3.7.5 HTML 特殊字元

為了表示字元或使用控制字元，因此提供了 HTML 特殊字元評估。

可以使用代碼頁 Latin 1 的十進制及十六進制編碼。

範例

字元	說明	標記法十進制	標記法十六進制
"	引號	"	"
LF	換行

3.7.6 運算子

操作指令處理下列運算子：

操作者	含義
=	配置
==	等於
<, <	小於
>, >	大於
<=, <=	小於或等於
>=, >=	大於或等於
|	OR 位元操作
||	邏輯 OR 操作
&, &	AND 位元操作
&&, &&	邏輯 AND 操作
+	加法
-	減法
*	乘法
/	除法
!	不
!=	不等於

操作指令由左至右處理。 在某些情況下，將條件放在括弧內，用以定義執行次條件的優先順序是較合理的做法。

3.7.7 系統變數

系統變數是每個指令碼提供的變數，用來進行語法分析器與指令碼執行之間的資料交換。

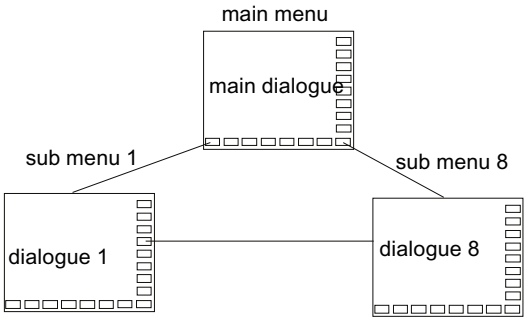
下表提供自動產生變數的標籤概觀。

變數名稱	意義	有效標籤
$actionresult	傳至語法分析器的訊號，指示是否以語法分析器處理事件。	KEY_EVENT
$focus_name	包含具有輸入焦點的欄位名稱	FOCUS_IN INDEX_CHANGED EDIT_CHANGED
$focus_item_data	包含指派給欄位的數值 item_data	
$gestureinfo	有關手指手勢，語法分析器提供變數及執行 gesture_event 標籤的手勢資訊。	GESTURE_EVENT
$return	變數會傳輸子函數的傳回值	FUNCTION_BODY
$message_par1 $message_par2	包含 SEND_MESSAGE 功能的呼叫參數	MESSAGE
$xmlAttribute	包含所找到的標籤屬性清單	XML_PARSER startElementHandler
$xmlValue	包含所找到的標籤值清單	
$xmlCharacters	包含資料流	XML_PARSER charactersHandler
$xmlCharactersStart	包含開始索引：	
$xmlCharactersLength	包含儲存在資料流內的字元數量	
$mouse_event.type $mouse_event.x $mouse_event.y $mouse_event.id $mouse_event.buttons $mouse_event.button	用來傳輸滑鼠事件參數的結構	MOUSE_EVENT

3.7.8 建立軟鍵功能表與對話方塊形式

只有在 XML 說明中有名稱為「main」的功能表標籤時，才能整合對話方塊格式。系統在 <CUSTOM> 操作區啟用後，會呼叫此標籤。可以分支其他對話方塊格式，並且啟用標籤內定義的對話方塊。

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

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

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



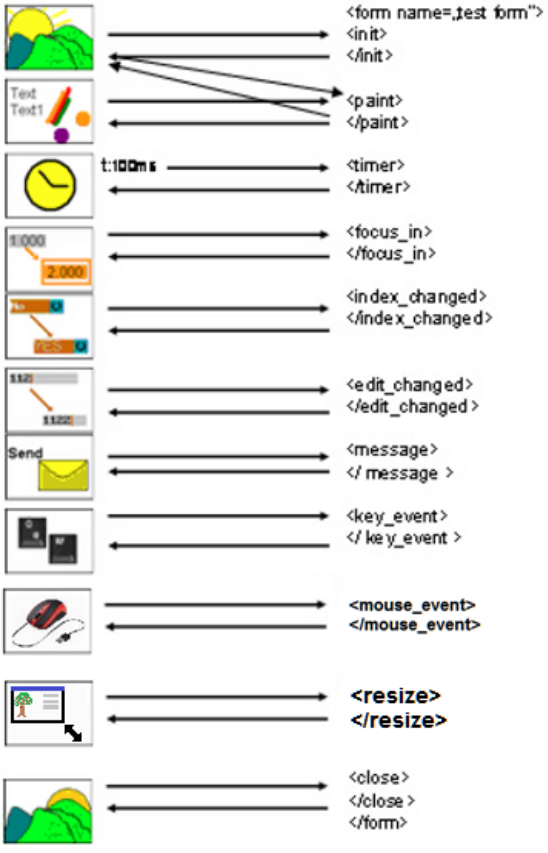
圖像 3-6　功能表結構

標籤識別碼	意義
FORM	此標籤包含使用者對話方塊的說明。相關的標籤說明，請參閱有關產生功能表與對話方塊遮罩的章節。 **語法：** `<FORM name="<dialog name>" color="#ff0000">` **屬性：** • **color** 對話方塊遮罩的背景色彩 **更多資訊**請參閱章節「色彩編碼 (頁 81)」 – 預設為白色 • **name** 格式的識別碼 • **type** 允許值為 *cycle* ，用以識別使用者循環畫面表單 • **xpos** 對話方塊左上角的 X 位置（選配） • **ypos** 左上角的 Y 位置（選配） • **width** X 方向延伸（以像素表示）（選配） • **height** Y 方向延伸（以像素表示）（選配）

標籤識別碼	意義
FORM 續	使用 AUTOSCALE_CONTENT 屬性，您可定義表單的控制項在不同螢幕解析度中的行為。依據預設，控制項會自動調整以符合已設定的螢幕解析度。如果您希望自行控制位置與尺寸，請在 Form 標籤中將此屬性值設為 OFF。 **屬性：** **autoscale_content** **值：** • **on** 控制項的座標會自動調整以符合螢幕解析度（預設值） • **off** 使用控制項的座標而不做變更 **範例：** <form name="main_form" autoscale_content="off"> … … … </form>
FORM 續	屬性： • **textfile** 該屬性允許指定與格式相關的文字檔案。針對標準文字內容，系統使用識別碼 EASY_XML。 • **textcontext** 屬性允許所要使用的文字內容指定識別碼。該屬性僅與屬性**文字檔案**結合使用為有效。

標籤識別碼	意義

FORM 續	為了能夠使用標準版面配置，或儲存在 easyscreen.ini 檔案中的版面佈置，應使用屬性 LAYOUT。輸入的數值應使用版面配置的名稱。
	附註：
	僅建議變更彈出視窗的預設設定，因為這些設定不會隱藏呼叫的對話方塊。
	
	屬性：
	layout
	
	語法：
	`<form name="格式的名稱" layout="版面配置的名稱" >`
	…
	…
	`</form>`
	
	範例：
	`<form name="form0" layout="slstandardscreenlayout.`
	`SlStandardScreenLayout.LowerForm" >`
	…
	…
	`</form>`
	
	除了預先定義的螢幕格式位置及尺寸，還可以將自己的定義儲存在 easyscreen.ini 檔案中。
	easyscreen.ini 之中的項目：
	`[640x480]`
	`MyPanel = x:=0, y:=220, width:=340, height:=174`
	`[800x480]`
	`MyPanel = x:=0, y:=220, width:=420, height:=174`
	
	範例：
	`<form name="form0" layout="MyPanel" >`
	…
	…
	`</form>`

標籤識別碼	意義
FORM 續	對話方塊訊息： • INIT • PAINT • TIMER • CLOSE • FOCUS_IN • INDEX_CHANGED • EDIT_CHANGED • GESTURE_EVENT • KEY_EVENT • MESSAGE • MOUSE_EVENT • RESIZE • CHANNEL_CHANGED • LANGUAGE_CHANGED
FORM 續	

標籤識別碼	意義

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

INIT	**對話方塊訊息** 此標籤在產生對話方塊後立即執行。對話方塊格式的所有輸入元素與「熱連結」應該在此建立。

標籤識別碼	意義

KEY_EVENT	對話方塊訊息
	標籤 KEY_EVENT 可以整合在格式中，以評估鍵盤事件。如果形式提供了標籤，系統會將 MF2 鍵盤代碼傳送至啟用的格式。如果變數 **$actionresult** 並未設定為零，系統接下來會處理鍵盤事件。
	變數 **$keycode** 會以整數值提供鍵盤代碼。
	
	範例：
	輸入至變數 **exclude_key** 的字元應從輸入串流中篩選出來。
	
	`<LET name="stream" type="string"/>`
	`<LET name="exclude_key" type="string"/>`
	
	`<FORM name = "keytest_form">`
	`  <INIT>`
	`  <CONTROL name = "p1" xpos = "120" ypos = "84" width ="200" refvar="stream" hotlink="true" />`
	`  <CONTROL name = "p2" xpos = "160" ypos = "104" width ="8" refvar="exclude_key" hotlink="true" />`
	`  </INIT>`
	
	`  <PAINT>`
	`  <text xpos = "8" ypos = "84">data stream</text>`
	`  <text xpos = "8" ypos = "104">exclude key</text>`
	`  </PAINT>`
	
	`  <KEY_EVENT>`
	`  <LET name="excl_keycode" type="string"/>`
	`  <OP>excl_keycode = exclude_key</OP>`
	`  <type_cast name="excl_keycode" type="int" />`
	`  <PRINT text="%d %d">$keycode, excl_keycode</PRINT>`
	
	`  <IF>`
	`  <CONDITION>$keycode == excl_keycode</CONDITION>`
	`  <THEN>`
	`  <op> $actionresult = 0</op>`
	`  </THEN>`
	`  </IF>`
	
	`  </KEY_EVENT>`
	
	`</FORM>`

標籤識別碼	意義

MOUSE_EVENT	此標籤可連結至處理滑鼠事件的指令碼。它會在使用滑鼠進行以下活動時間時執行：
	• 已按下按鈕
	• 已放開按鈕
	• 已移動滑鼠
	語法分析器提供結構中的資訊並以下列元件建立結構變數 **$mouse_event**：
	
	結構元件：
	• **type**
	活動的編碼
	– 2 - 已按下按鈕
	– 3 - 已放開按鈕
	– 5 - 已移動滑鼠
	• **x**
	游標的 X 位置（以像素表示）；相對於目前螢幕解析度
	• **y**
	游標的 Y 位置（以像素表示）；相對於目前螢幕解析度
	• **id**
	識別碼
	– -1，如果此位置無法指派至任何控制項
	– != -1，如果滑鼠游標位於控制項內，將傳回 **idemdata** 屬性的內容
	• **button**
	包含事件發生時的按鈕狀態
	– 0 - 無按鈕
	– 1 - 左按鈕
	– 2 - 右按鈕
	– 4 - 中央按鈕
	這些按鈕可與逐個位元的 OR 操作關聯。
	
	範例：
	`<MOUSE_EVENT>`
	`  <print text="button %d type %d x %d y %d ">$mouse_event.button, $mouse_event.type, $mouse_event.x, $mouse_event.y</print>`
	`</MOUSE_EVENT>`

標籤識別碼	意義

MESSAGE	**對話方塊訊息**
	如果 Send_message 操作是在指令碼中執行，則語法分析器將處理標籤訊息。變數 $message_par1 及 $message_par2 分別提供數值 P1 及 P2（請參閱「SEND_MESSAGE」標籤）。
	
	語法：
	`<MESSAGE>`
	`</MESSAGE>`
	
	範例：
	`<LET name="user_selection" />`
	
	`<SOFTKEY POSITION="3">`
	`<CAPTION>Set%nParameter</CAPTION>`
	`<SEND_MESSAGE>1, 10</SEND_MESSAGE>`
	`</SOFTKEY>`
	…
	…
	
	`<FORM>`
	…
	…
	`<MESSAGE>`
	`<SWITCH>`
	`<CONDITION>$message_par1</CONDITION>`
	`<CASE value="1">`
	`<OP> user_selection = $message_par2 </OP>`
	…
	…
	`</CASE>`
	`</SWITCH>`
	`</MESSAGE>`
	…
	…
	`</FORM>`

標籤識別碼	意義

SEND_MESSAGE	標籤傳送兩個參數的訊息給啟用的格式，在標籤訊息中進行處理（另請參閱 MESSAGE）。
	
	語法：
	`<SEND_MESSAGE>p1, p2</SEND_MESSAGE>`
	
	範例：
	`<LET name="user_selection" />`
	
	`<SOFTKEY POSITION="3">`
	`  <CAPTION>Set%nParameter</CAPTION>`
	`  <SEND_MESSAGE>1, 10</SEND_MESSAGE>`
	`</SOFTKEY>`
	`…`
	`…`
	`<FORM>`
	`…`
	`…`
	`<MESSAGE>`
	`<SWITCH>`
	`<CONDITION>$message_par1</CONDITION>`
	`<CASE value="1">`
	`<OP> user_selection = $message_par2 </OP>`
	`…`
	`</CASE>`
	`</SWITCH>`
	`</MESSAGE>`
	`…`
	`…`
	`</FORM>`

標籤識別碼	意義
RESIZE	**對話方塊訊息** 此標籤可連結至處理 RESIZE 事件的指令碼。此事件是由動態解析度切換所建立。 

標籤識別碼	意義

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

標籤識別碼	意義

CHANNEL_CHANGED	**對話方塊訊息**
	標籤能夠處理變更通道的事件。當使用者按下通道前進按鈕時執行。
	
	語法：
	`<channel_changed>`
	`</channel_changed>`
	
	範例：
	變更通道後，格式將關閉並再次開啟，以便顯示所選通道的資料。
	`<channel_changed>`
	`  <open_form name = "form1" reopen="true"/>`
	`</channel_changed>`
	
	或
	
	變更通道後，將啟動另一個功能表。
	`<channel_changed>`
	`  <navigation>menu2</navigation>`
	`</channel_changed>`

LANGUAGE_CHANGED	**對話方塊訊息**
	標籤允許處理語言的切換要求。當使用者開啟螢幕視窗切換語言時執行。
	
	語法：
	`<language_changed>`
	`</language_changed>`
	
	範例：
	切換語言後，視窗將關閉並再度開啟，以便能夠為控制提供已啟動語言的文字元素。
	`<language_changed>`
	`  <open_form name = "form1" reopen="true"/>`
	`</language_changed>`

標籤識別碼	意義

FOCUS_IN	**對話方塊訊息**
	系統將焦距置於控制時，會呼叫這個標籤。為了識別控制項，系統會將控制的名稱複製到變數 **$focus_name** ，並將屬性 **item_data** 的值複製到變數 **$focus_item_data**。
	舉例來說，這個訊息可以根據焦距位置用來輸出影像。
	範例：
	`<focus_in>`
	`<PRINT text="focus on filed:%s, %d">$focus_name, $focus_item_data </PRINT>`
	`</focus_in>`

PAINT	**對話方塊訊息** 此標籤在對話方塊顯示後執行。所有待顯示在對話方塊的文字與影像必須在此指定。 其次，如果系統識別對話方塊有些部分需要重新顯示，即執行標籤。例如，關閉高階層視窗可以初始化此動作。
TIMER	**對話方塊訊息** 此標籤會週期性執行。 每個格式指派一個計時器，將計時器初始化，標籤大約每 100 毫秒執行一次。

標籤識別碼	意義

CAPTION	此標籤包含對話方塊的標題。
	此標籤必須用在 INIT 標籤內。
	可將標題列分割為多個欄位。
	為了分割標題列，「caption」標籤必須在文字輸出前以「define_section」屬性程式化。
	「define_section」屬性指定欄位數量。
	
	「property」屬性會用來定義每個欄位的長度、開始位置及文字對齊，位置與長度的規格是參照格式寬度的百分比值。
	「value」屬性值指示欄號（從零開始）。
	
	`<caption define_sections="3">`
	`<property value="0" length="20" position="2" alignment="left" />`
	`<property value="1" length="20" position="79" alignment="right" />`
	`</caption>`
	
	藉由使用欄位索引來額外指定索引屬性，即可將文字指派到欄位。
	`<caption index="0">column1</caption>`
	`<caption index="1">column2</caption>`
	
	語法：
	`<CAPTION>Titel</CAPTION>`
	
	範例：
	`<CAPTION>my first dialogue</CAPTION>`

CLOSE	**對話方塊訊息** 此標籤在對話方塊關閉前執行。

標籤識別碼	意義

CLOSE_FORM	標籤關閉啟用的對話方塊。
	若以 MMC 命令打開對話方塊且提供使用者軟鍵功能來關閉對話方塊時，此項指令才有必要。一般而言，對話方塊會自動管理，不需要明確關閉。
	
	語法：
	`<CLOSE_FORM/>`
	
	範例：
	`<softkey_ok>`
	` <caption>OK</caption>`
	` <CLOSE_FORM />`
	` <navigation>main_menu</navigation>`
	`</softkey_ok>`

標籤識別碼	意義
CONTROL	此標籤用來產生控制元素。 **語法：** `<CONTROL name = "<control name>" xpos = "<X 位置>" ypos = "<Y 位置>" refvar = "<NC 變數>" hotlink = "true" format = "<格式>" />` **附註：** 使用變數設定屬性值時，必須宣告為全域變數。 **屬性：** • **name** 欄位的識別碼。 識別碼同時代表區域變數，不得在格式中多次使用。 • **xpos** 左上角的 X 位置 • **ypos** 左上角的 Y 位置 • **fieldtype** 欄位類型 若無指定類型，欄位會設定為一個編輯欄位。 – **edit** 可以變更資料 – **readonly** 無法變更資料 **combobox** 此欄位顯示對應的識別碼，而非數值。 若選擇 **combobox** 欄位類型，則待顯示的表示式必須指派到此欄位。 <ITEM> 標籤應該用在此目的。 組合方塊將當前選擇的文字指標，儲存在屬於控制的變數中（請參閱屬性 **refvar**）。 控制器建立後，可以使用函數 **addItem** 或 **insertItem** 插入其他元件。 – **progressbar** 顯示值從 0 到 100 的進度列。 谷值與峰值內容可以用來調整值域到待顯示的資料。

標籤識別碼	意義

CONTROL 續	• **fieldtype edit** 及 **readonly**
	針對欄位類型 **edit** 及 **readonly**，**multiline** 屬性允許文字以多重行顯示。換行符發生在字組邊界或換行符 <LF> 處。
	範例：
	`<control name="mulitline_edit" xpos="280" ypos="60" width="200" height="100" type="string">`
	`  <property multiline="true" />`
	`</control>`
	`<control name="c2" xpos="280" ypos="260" width="200" height="100" type="string" fieldtype="readonly">`
	`  <property multiline="true" />`
	`</control>`

標籤識別碼	意義

CONTROL 續	• **fieldtype**
	– **graphicbox**
	欄位類型產生一個 2D 的虛線圖形控制。使用標籤 <ITEM> 可以將圖形元件插入控制中。參數 **width** 及 **height** 指定方塊的寬度及高度。
	控制器建立後，可以使用函數 **addItem** 或 **insertItem** 插入其他元件。參數 **itemdata** 並未評估這項控制。
	如果參考變數已指派至此控制項，其數值將以 100 ms 為時間基礎進行記錄。最多可記錄 10000 個值。使用 **max** 特性，即可無限制地記錄值。達到最大的記錄時間時，則會刪除最舊的值。必須以毫秒指定記錄時間。
	雖然數值是在分散時間儲存，但是控制項會以連接的線條顯示這些數值。屬性 **channel** 最多允許另外記錄三個參考變數。索引從 1 開始。索引 0 用於設定控制標籤中指派的變數特性。
	範例：
	`<control name= "c_gbox1" xpos = "250" ypos="24" width="240" height="356" fieldtype="graphicbox" refvar="val" hotlink="true" COLOR_BK="#ffffff">`
	`  <property max="60000" /> <!-- steps of ms -->`
	`  <property min="0" />`
	`  <property channel="0" color="#00ffff" penwidth="3"/>`
	`  <property ORDINATE="Y sinus" />`
	`  <property ABSCISSA="time *100 ms" />`
	`  <property channel="1" refvar="val2" color="#000000" />`
	`</control>`
	
	`<request name="nck/Channel/Parameter/R[2]" />`
	`<control name= "c_gbox" xpos = "0" ypos="5"  width="260" height="200" fieldtype="graphicbox" refvar="nck/Channel/Parameter/R[1]" hotlink="true" COLOR_BK="#ffffff">`
	`  <property max="400" />`
	`  <!-- steps of  ms  -->`
	`  <property min="0" />`
	`  <property channel="0" color="#00ff00" penwidth="1"/>`
	`  <property ORDINATE="Power" />`
	`  <property ABSCISSA="Time *100 ms" />`
	`  <property channel="1" refvar="nck/Channel/Parameter/R[2]" color="#FF0000" penwidth="1"/>`
	`  <property FIX_TIME_AREA="on" />`
	`</control>`

標籤識別碼	意義
CONTROL 續	**範例圖形盒：** `<CONTROL name= "graphic" xpos = "8" ypos="23" width="300" height="352" fieldtype="graphicbox" />` • 新增元件： 使用 **additem** 或 **loaditem** 函數新增元件。 可使用下列二維元件： – 直線 - l(inc) – 圓形區段 - c(ircle) – 點 - p(oint) **元件的結構：** <元件類型>；座標 • 直線： l、xs、ys、xe、ye l - 直線標記 Xs - X 起始位置 Ys - Y 起始位置 Xe - X 結束位置 Ye - Y 結束位置 • 圓形： C、xs、ys、xe、ye、cc_x、cc_y、r C - 圓形區段標記 Xs - X 起始位置 Ys - Y 起始位置 Xe - X 結束位置 Ye - Y 結束位置 Cc_x - X 座標，圓心點 Cc_y - Y 座標，圓心點 • 半徑： R • 點： P、x、y P - 點標記 X - X 位置 Y - Y 位置 • 刪除圖形： 使用 **empty** 函數來刪除內容。

標籤識別碼	意義
CONTROL 續	以下的欄位類型將於「設定自己的按鈕 (頁 241)」一節中說明。 • **fieldtype** – **pushbutton** 使用欄位類型作為按鈕或開關。 – **radiobutton** 欄位類型能夠選擇幾個選項的其中一個。 – **checkbox** 欄位類型能夠選擇好幾個選項。 – **groupbox** 欄位類型以框架及標題，將控制群組以視覺化涵蓋其中。 – **switch** 欄位類型為圖形化元件，使用圖示發出兩個狀態的其中一個訊號。 – **scrollarea** 使用欄位類型將控制項顯示在指定區域內。

標籤識別碼	意義
CONTROL 續	• **fieldtype** – **listbox** 欄位類型產生空白的列表方塊控制。 使用標籤 <ITEM> 可以將列表方塊元件插入列表方塊中。 ITEM 屬性 **value** 允許指派唯一值給這個元件。 例如，這可以用來識別元件。 參數 **width** 及 **height** 指定列表方塊的寬度及高度。 控制器建立後，可以使用函數 **addItem**、**insertItem** 或 **loadItem** 插入其他的列表方塊元件。 控制器建立後，可以使用函數 **addItem** 或 **insertItem** 插入其他的列表方塊元件。參數 **itemdata** 並未評估這項控制。 – **itemlist** 此欄位類型產生一個靜態控制，以顯示對應的識別碼，而非數值。 可以使用 <ITEM> 標籤將識別碼指派給欄位。 • **item_data** 使用者專屬的整數值可以指派到此屬性。這個值作為 FOCUS_IN 訊息的一部分，用來辨識焦距區域。 • **refvar** 可以連結至欄位的參考變數識別碼（選配）。 • **hotlink** = "TRUE" 如果參考變數的數值變更，則欄位會自動更新（選配）。 • **format** 屬性定義指定變數的顯示格式。 格式化資料，請參閱 print-Tag （選配）。 **格式**屬性還允許將整數值顯示在其他數字系統中，或顯示為欄位類型**編輯**及**僅能讀取**的 Boolean 值。不支援使用字符數及對齊方式進行格式化。 可使用下列的表示法類型： • **%o** - 八進制 • **%x** - 十六進制 • **%n** - 二進制 • **%b** - bool

CONTROL 續	**建立列、屬性的範例：**
	`<control name="lb1" xpos = "10" ypos = "94" width="200" height="250" fieldtype="listbox" refvar="tmp" hotlink="true">`
	`  <property columns="4" />`
	`  <property column="0" type="width">190</property>`
	`  <property column="1" type="width">100</property>`
	`  <property column="2" type="width">190</property>`
	`  <property column="3" type="width">16</property>`
	`</control>`

標籤識別碼	意義

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

標籤識別碼	意義
CONTROL 續	**屬性：** • **color_bk** 這項屬性用來設定控制的背景顏色。 • **color_fg** 這項屬性用來設定控制的前景顏色。 **更多資訊**請參閱章節「色彩編碼 (頁 81)」 • **display_format** 屬性定義指定變數的處理格式。存取 PLC 浮點變數時必須使用此一屬性，因為存取須要透過讀取雙字元組來完成。 可允許的資料格式如下： – FLOAT – INT – DOUBLE – STRING 將表示式（例如所要顯示的文字或圖形元件）指派給列表方塊、圖形方塊或組合方塊： **語法：** <ITEM>表示式</ITEM> <ITEM value ="<數值>">表示式</ITEM>

標籤識別碼	意義

CONTROL 續	**範例：**
	`<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = "combobox ">`
	`<ITEM>text1</ITEM>`
	`<ITEM>text2</ITEM>`
	`<ITEM>text3</ITEM>`
	`<ITEM>text4</ITEM>`
	`</CONTROL>`
	若任一整數值將指派到一個表示式，屬性 **value = "value"** 應加到此標籤。
	控制變數現在包含該項目的指派值，而非連續數字。
	
	範例：
	`<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = "combobox ">`
	`<ITEM value = "10">text1</ITEM>`
	`<ITEM value = "20">text2</ITEM>`
	`<ITEM value = "12">text3</ITEM>`
	`<ITEM value = "1">text4</ITEM>`
	</CONTROL>
	進度列範例：
	`<CONTROL name = "progress1" xpos = "10" ypos = "10" width = "100" fieldtype = "progressbar" hotlink = "true" refvar = "nck/Channel/GeometricAxis/actProgPos[1]">`
	`<PROPERTY min = "0" />`
	`<PROPERTY max = "1000" />`
	`</CONTROL>`
	例如，列表方塊：
	`<let name="item_string" type="string"></let>`
	`<let name="item_data" ></let>`
	
	`<CONTROL name="listbox1" xpos = "360" ypos="150" width="200" height="200" fieldtype="listbox" />`
	• 新增元件：
	使用函數 **additem** 或 **loaditem** 新增元件。
	• 刪除內容：
	使用函數 **empty** 來刪除內容。
	
	`<op> item_string = _T"text1\\n" </op>`
	`<function name="control.additem">_T"listbox1", item_string, item_data </function>`
	`<op> item_string = _T"text2\\n" </op>`
	`<function name="control.additem">_T"listbox1", item_string, item_data </function>`

標籤識別碼	意義

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

CONTROL 續	影像方塊控制項管理點陣或 GIF 格式的圖片。如果圖片大於可顯示區域，控制項將顯示捲動軸。 為控制可見區域，系統提供 CONTROL.IMAGEBOXSET 函數。 **更多資訊**請參閱章節「預先定義之函數 (頁 172)」 **語法：** `<function name="control.imageboxget"> ... </function>`
CONTROL 續	**屬性：** • **disable** 屬性會在編輯控制器中鎖定/允許輸入。 • **tooltip** 若游標放置在控制項上，即顯示資訊文字。 • **factor** 轉換係數 • **font** 字型的定義 • **fontpixelsize** 字元高度 - 該值是以像素的數量指定，指的是 640x480 像素的解析度。 根據實際的解析度與參考解析度的比率，確定顯示的字元高度。 **範例：** `<control name="c2" xpos="180" ypos="120" fontname="arial" fontpixelsize="26" height="100" type="string" />`

標籤識別碼	意義
CONTROL 續	**屬性：** • **fontname** 該屬性係用來指定所要使用的字型。可以使用 HMI.GET_FONT_FAMILIES 函數來確定已安裝的字型。 **值：** 字型名稱 以下西門子字型安裝於系統中： • Siemens AD CH 簡體 • Siemens AD CH 繁體 • Siemens AD JP • Siemens AD KS • Siemens AD Mono • Siemens AD Mono CH 簡體 • Siemens AD Mono CH 繁體 • Siemens AD Mono JP • Siemens AD Mono KS • Siemens AD Mono TH • Siemens AD Mono VN • Siemens AD Sans • Siemens AD TH • Siemens AD VN • Siemens Logo • Siemens Sans Cond Global • Siemens Sans Cond Mono • Siemens Sans Global 以 MS Windows 為基礎的系統可能包含其他字型。 **範例：** `<control name="c2" xpos="180" ypos="120" fontname="arial" fontpixelsize="26" height="100" type="string" />`

標籤識別碼	意義
CONTROL 續	**在建立後變更控制** 控制標籤於建立後變更現有控制器的特性。必須使用將要變更的控制名稱及新的特性來指定標籤。只可在格式標籤內執行。可變更下列特性，例如： • 名稱 • xpos • ypos • 寬度 • 高度 • color_bk • color_fg • 存取層級 • fieldtype • itemdata • 最小值 • 最大值 • 預設值 • 停用 • 工具提示 • 字型 • 係數 無法修改參考變數。若特性將藉由軟鍵事件引起的觸發來變更，則傳送訊息標籤必須傳輸這項請求到格式語境。訊息標籤用於取得此訊息。

標籤識別碼	意義

CONTROL 續	**在操作說明中的控制變更**
	在執行時間中變更控制特性的另一項可能性，即是在操作說明中進行變更。因此必須定義控制項的名稱，以及所要指派的新值特性。其特性與控制項名稱之間以點分開。
	語法：
	<名稱>.<特性>
	範例：
	`...`
	`...`
	`<let name="value" />`
	`<let name="w" />`
	`<let name="h" />`
	`...`
	`...`
	`<control name="c_move" xpos="$xpos" ypos="124" />`
	`...`
	`...`
	`<op>`
	`c_move.xpos = 300;`
	`value = c_move.xpos;`
	`h = c_move.height;`
	`w = c_move.width;`
	`</op>`

標籤識別碼	意義

HELP_CONTEXT	此標籤定義待呼叫的說明主題，應該在 INIT 單節中由程式設計。
	808/802D sl 系統
	屬性指定的名稱以前置碼 XmlUserDlg_ 加以補充，並傳輸至說明系統。說明檔案的相關結構應從主題擷取以產生線上說明。
	啟動說明系統的順序：
	1. 按下「資訊」軟鍵。
	2. 對話方塊提供表示式 "my_dlg_help"。
	3. 語法分析器將表示式轉換為 "XmlUserDlg_my_dlg_help" 。
	4. 啟動說明系統。
	5. 提交搜尋詞語 "XmlUserDlg_my_dlg_help"。
	840D sl/828D 系統
	有關說明手冊的版面配置及產生的**更多資訊**：
	SINUMERIK Operate 調試手冊
	
	屬性：
	• **name**
	在屬性中指定的名稱將傳輸至說明系統。在標籤 **entry** 中的說明手冊所使用的檔案名稱必須指定。
	• **anchor**
	使用屬性可以輸入關鍵字。
	
	語法：
	`<HELP_CONTEXT name="<file name>" anchor="key word"/>`
	
	範例：
	`<form name = "form1">`
	` <init>`
	`  <help_context name="chapter_1.html" anchor="Keyword_1" />`
	`...`
	`...`
	`  <control name = "cpwd" xpos = "322" ypos = "64" >`
	`   <property password="true" />`
	`   <property help_context="chapter_1.html" anchor="Keyword_2" />`
	`  </control>`
	`</form>`

標籤識別碼	意義
DATA_ACCESS	使用者輸入儲存後，此標籤控制對話方塊格式的行為。 此行為必須定義在 INIT 標籤中。 若不使用此標籤，在每種情況下都會緩衝輸入。 例外情形：將 **hotlink** 屬性設定為 **true** 的控制器總是直接遭到讀寫。 **屬性：** • **type** = "TRUE" – 輸入值未經過緩衝。對話方塊格式將輸入值直接複製到參考變數。 • **type =** "FALSE" – 數值只複製到含有 **UPDATE_CONTROLS** type = "FALSE" 標籤的參考變數。 **範例：** `<DATA_ACCESS type = "true" />`

標籤識別碼	意義

DEBUG	該標籤用於儲存追蹤輸出，可以在命令集中對其進行編程。屬性 **text** 係用來寫入所要儲存的文字。此屬性及其數值對應於 **text** 屬性，及列印指令的值。
	在預設情況下，不執行除錯標籤，因為會降低系統執行速度。欲啟動除錯輸出，編程屬性 **debug_msg**，將數值 **on** 編程在 **DialogGui** 或 **AGM** 標籤中。
	除錯輸出與解析器錯誤訊息一起儲存在以下檔案中：
	card/oem/sinumerik/hmi/dvm/log/dvm.log
	
	語法：
	`<debug text="<文字請參閱列印指令>">變數請參閱列印指令</debug>`
	
	Easy XML 範例 ：
	`<DialogGui debug_msg="on">`
	`...`
	`...`
	`<control name="test_bool" xpos="250" ypos="90"`
	`refvar="val_test" hotlink = "true" format="%b"/>`
	`<debug text="main_form control test_bool created`
	`value:%d">test_bool</debug>`
	
	在檔案 dvm.log 中的輸入項：
	`date: 10/22/ 2018 time: 09:38:44:351`
	`Debug xmldial.xml 17: main_form control test_bool`
	`created value:255`
	
	簡易擴充範例：
	`<AGM debug_msg="on">`
	`...`
	`...`
	`<control name="test_bool" xpos="250" ypos="90"`
	`refvar="val_test" hotlink = "true" format="%b"/>`
	`<debug text="main_form control test_bool created`
	`value:%d">test_bool</debug>`
	
	在檔案 dvm.log 中的輸入項：
	`date: 10/22/ 2018 time: 09:38:44:351`
	`Debug xmldial.xml 17: main_form control test_bool`
	`created value:255`

標籤識別碼	意義

EDIT_CHANGED	**對話方塊訊息**
	如果編輯的控制已變更，呼叫此標籤。
	為了識別控制，系統會將控制的名稱複製到變數 **$focus_name** ，並將屬性 **item_data** 的值複製到變數 **$focus_item_data**。
	
	範例：
	`<EDIT_CHANGED>`
	`<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>`
	…
	…
	`</EDIT_CHANGED>`

GESTURE_EVENT	標籤係用來執行多點觸控操作的手指手勢。 在「多點觸控操作 (頁 233)」一節中有說明標籤。

INDEX_CHANGED	**對話方塊訊息**
	運算子變更組合方塊的選擇時，會呼叫這個標籤。
	為了識別控制，系統會將控制的名稱複製到變數 **$focus_name** ，並將屬性 **item_data** 的值複製到變數 **$focus_item_data**。
	附註：
	指派給控制的參考變數未能及時對正此點的控制變數，並包含先前組合方塊所選擇的索引。
	
	範例：
	`<INDEX_CHANGED>`
	`<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>`
	…
	…
	`</INDEX_CHANGED>`

標籤識別碼	意義

MENU	此標籤定義包含軟鍵說明與待打開的對話方塊的功能表。
	
	屬性：
	• **name**
	功能表名稱
	
	語法：
	`<MENU name = "<menu name>">`
	…
	`<open_form …>`
	…
	`<SOFTKEY …>`
	`</SOFTKEY>`
	`</MENU>`

NAVIGATION	此標籤定義待呼叫的功能表。它可用於軟鍵單節、功能表單節以及表單。如果變數名稱已指派至標籤做為其值，語法分析器將啟用儲存於變數中的功能表。
	在功能表單節中，導覽是在指令中的位置。後續的指令將不再執行。
	
	附註：
	若導覽目標是由變數的內容判定，此變數必須宣告為全域變數。
	
	語法：
	`<NAVIGATION>menu name</NAVIGATION>`
	
	範例：
	`<menu name = "main">`
	`<softkey POSITION="1">`
	`<caption>sec. form</caption>`
	`<navigation>sec_menu</navigation>`
	`</softkey>`
	`</menu>`
	
	`<menu name = "sec_menu">`
	`<open_form name = "sec_form" />`
	`<softkey_back>`
	`<navigation>main</navigation>`
	`</softkey_back>`
	</menu>

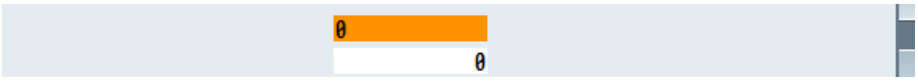
標籤識別碼	意義

OPEN_FORM	此標籤打開指定名稱的對話方塊格式。
	如果指定的螢幕格式已啟用，則不會執行該標籤。
	透過輸入屬性 **reopen**，可以強制螢幕格式再度開啟。如果變更系統狀態，需要重新排列螢幕格式內容，則必需執行此動作。
	
	屬性：
	• **name**
	對話方塊格式的名稱
	• **reopen**
	如果該屬性的值設置為 **true**，則再次呼叫標籤 **open_form** 時，會檢查螢幕格式是否為啟用格式。若是，則分析器會關閉啟用的格式，並再次開啟。
	
	語法：
	`<OPEN_FORM name = "<form name>" />`
	
	範例：
	`<menu name = "main">`
	`  <open_form name = "main_form" />`
	`  <softkey POSITION="1">`
	`  <caption>main form</caption>`
	`  <navigation>main</navigation>`
	`  </softkey>`
	`</menu>`
	
	`<form name="main_form">`
	`  <init>`
	`  </init>`
	
	`  <paint>`
	`  </paint>`
	
	`</form>`

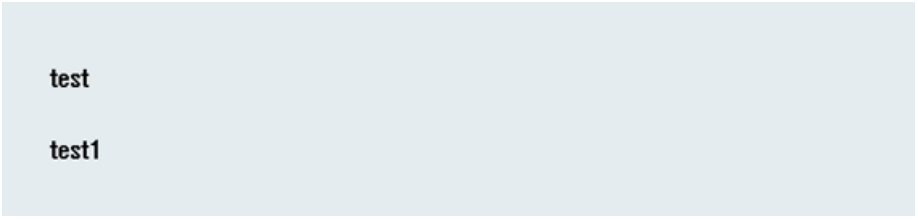
標籤識別碼	意義

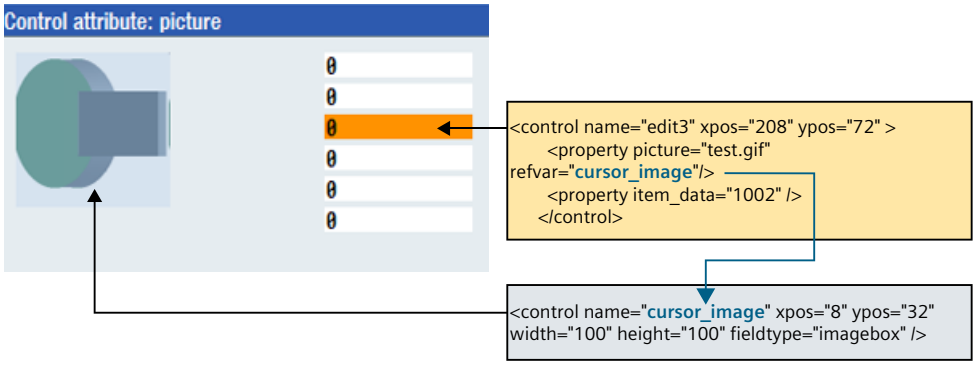
PROPERTY	此標籤可用來定義操作者控制的額外內容。標籤內嵌在控制標籤中。
	
	屬性：
	• **max** = "<最大值>"
	• **min** = "<最小值>"
	• **default** = "<預設值>"
	• **factor** = "轉換係數"
	• **color_bk** = "<背景色彩編碼>"
	• **color_fg** = "<字型色彩編碼>"
	• **password** = "<true>" - 輸入字元顯示為「*****」
	• **multiline** = "<true>" - 在編輯控制器中允許多行輸入
	• **disable** = "<true/false>" - 在編輯控制器中阻斷/允許輸入
	• **transparent** = "點陣圖的透明色"
	更多資訊請參閱章節「色彩編碼 (頁 81)」
	• **tooltip** = 資訊文字只會在游標設為控制時顯示。
	語法：<property tooltip="note text" />
	• **abscissa** = "第一個座標軸的名稱"（僅允許用於圖形方塊）
	• **ordinate** = "第二個座標軸的名稱"（僅允許用於圖形方塊）
	• **fix_time_area** = "on" - 屬性 **min** 及 **max** 定義了時間區間，顯示出訊號。顯示的訊號會從右半部移到左半部。
	範例：
	`<CONTROL name = "progress1" xpos = "10" ypos = "10" width = "100" fieldtype = "progressbar" hotlink = "true" refvar = "nck/Channel/GeometricAxis/actProgPos[1]">`
	`<PROPERTY min = "0" />`
	`<PROPERTY max = "1000" />`
	`</CONTROL>`
	`<CONTROL name = "edit1" xpos = "10" ypos = "10">`
	`<PROPERTY min = "20" />`
	`<PROPERTY max = "40" />`
	`<PROPERTY default = "25" />`
	`</CONTROL>`

標籤識別碼	意義

PROPERTY 續	**屬性：**
	alignment - 該屬性允許左邊或右邊文字對齊。預設值是左邊文字對齊。
	
	值：
	• **left**
	• **right**
	
	語法：
	`alignment="<right/left>"`
	
	範例：
	`<control name="edit2" xpos="208" ypos="52" >`
	`<property alignment="right"/>`
	`</control>`
	
	 

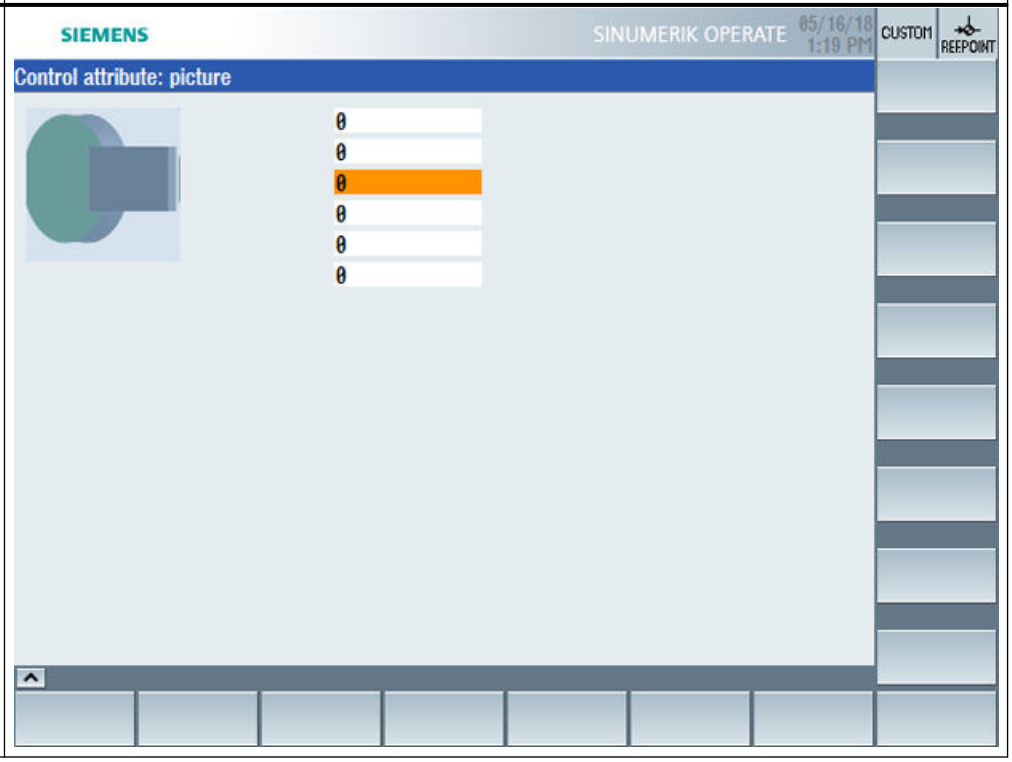
標籤識別碼	意義

PROPERTY 續	**屬性：**
	parent - 該屬性確定控制的成員資格。預設值是將控制指派給格式。若欲將控制指派給捲動檢視，則必須將其指定為母視窗。
	
	值：
	母視窗的名稱
	
	範例：
	`...`
	`<control name="area" xpos="8" ypos="72" width="554" height="320" fieldtype="scrollarea" >`
	`<control name="test" xpos="20" ypos="40" width="20" fieldtype="readonly" />`
	`</control >`
	`<control name="test1" xpos="20" ypos="80" width="20" fieldtype="readonly" parent="area"/>`
	 
	`<init>`
	`<control name="area" xpos="8" ypos="72" width="554" height="120" fieldtype="scrollarea">`
	`<control name="test" xpos="20" ypos="40" width="80" fieldtype="readonly" type="string"/>`
	`</control >`
	
	`<control name="test1" xpos="20" ypos="80" width="80" fieldtype="readonly" type="string" parent="area"/>`
	`<op>test = _T"test"</op>`
	`<op>test1 = _T"test1"</op>`
	`<update_controls type="true" />`
	`...`

標籤識別碼	意義
PROPERTY 續	**屬性：** **picture** - 屬性指定了所要顯示的圖形 若是將輸入焦點設定在這個欄位，所要顯示的指定圖形或圖形名稱會儲存在變數中。將本屬性搭配使用 **refvar** 屬性，可以定義資料接收器。 欲顯示圖形或動畫，必須指定 **imagebox** 類型控制的名稱。此控制器會假定圖形的管理。 若指定區域變數的名稱，則此變數必須為 **String** 資料類型。 圖形會持續顯示，直到新名稱指派給參考變數為止。 **語法：** `<property picture="<Name der Grafik>" refvar="<Datensenke>"/>`
PROPERTY 續	**欄位連結範例：** 

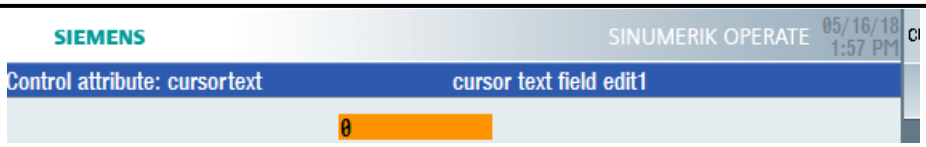
標籤識別碼	意義

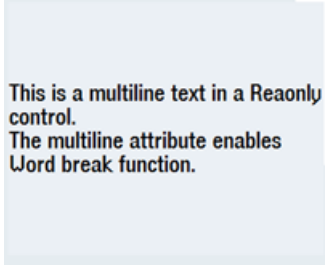
PROPERTY 續	**範例：**
	`<let name="cursor_icon" type="string" />`
	` <form name="main_form1">`
	`  <init>`
	`  <caption>Control attribute: picture</caption>`
	`  <control name="cursor_image" xpos="8" ypos="32" width="100" height="100" fieldtype="imagebox" />`
	`  <control name="edit1" xpos="208" ypos="32"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

標籤識別碼	意義
	 SIEMENS SINUMERIK OPERATE 05/16/18 1:19 PM CUSTOM REFPOINT Control attribute: picture 0 0 0 0 0 0

標籤識別碼	意義

PROPERTY 續	**屬性：**
	cursortext - 此屬性定義所要顯示的游標文字
	
	若是將輸入焦點設定在此欄位上，會將文字發出至標題列中。
	除了這個定義文字對齊及游標文字區的屬性之外，還可以指定兩個其他的屬性。依標準，文字是靠左對齊。游標文字區域預設值將佔掉標題列寬度最多 50%。
	
	`alignment="<right/left>"` - 文字對齊靠右對齊/靠左對齊
	`length="<寬度>"` - 游標文字所需的標題列百分比
	
	語法：
	`<property cursortext="<text>" />`
	或
	`<property cursortext="<text>" alignment="<right/left>" />`
	或
	`<property cursortext="text2" alignment="r="<text>" alignment="<right/left>" length="<比率>"/>`
	
	範例：
	`<form name="main_form2">`
	` <init>`
	`  <caption>Control attribute: cursortext</caption>`
	`  <control name="edit1" xpos="208" ypos="32" >`
	`   <property cursortext="cursor text field edit1" />`
	`  </control>`
	`  <control name="edit2" xpos="208" ypos="52" >`
	`   <property cursortext="cursor text field edit2" alignment="right"/>`
	`  </control>`
	`  <control name="edit3" xpos="208" ypos="72" >`
	`   <property cursortext="cursor text field edit3" alignment="right" length="40"/>`
	`  </control>`
	`  <control name="edit4" xpos="208" ypos="92" >`
	`   <property cursortext="cursor text field edit4" />`
	`  </control>`
	` </init>`
	`</form>`

標籤識別碼	意義
	

PROPERTY 續	**屬性：**
	multiline - 此將允許在編輯或唯讀控制中的多行文字輸出
	
	語法：
	`<property multiline="true" />`
	
	範例：
	`...`
	`...`
	`<op>`
	`textlist[2] = _T"This is a multiline text in a Reaonly control.\`
	`\nThe multiline attribute enables Word break function.";`
	`</op>`
	`<control name="lstring" xpos="8" ypos="120" width="200" height="160"`
	`fieldtype="readonly" refvar="textlist[2]" >`
	`<property multiline="true" />`
	`</control>`
	
	 

標籤識別碼	意義

PROPERTY 續	**屬性：**
	• **help_context**
	透過該屬性可以為控制指派說明主題。在說明手冊中的標籤 **entry** 所使用的檔案名稱必須指定。
	• **anchor**
	使用屬性可以輸入關鍵字。該屬性僅與屬性 **help_context** 結合使用為有效。
	
	語法：
	`<property help_context="<file name>" anchor="<key word>"/>`
	
	範例：
	`<control name = "cpwd" xpos = "322" ypos = "64" >`
	`<property password="true" />`
	`<property help_context="chapter_1.html" anchor="Keyword_2" />`
	`</control>`

SOFTKEY	本標籤定義軟鍵的內容與反應。 **屬性：** • **position** 軟鍵的編號。1-8 為橫向軟鍵，9-16 為縱向軟鍵 下列屬性開始生效處： • **type** 定義軟鍵的屬性。 user_controled - 指令碼定義如何顯示軟鍵 toggle_softkey - 軟鍵在按下與未按下之間交替顯示 • **refvar** 必須搭配 toggle_softkey 一起使用。 複製實際軟鍵屬性的參考變數。 變數，必須指定類型 "String"，包含按下、未按下或鎖定時的屬性（請參閱標籤 state）。 • **picture** 使用屬性，點陣圖可以在軟鍵上向左調整輸出。必須指定完整的路徑名稱。 可顯示的文字字元數減少至點陣圖的寬度。

標籤識別碼	意義
SOFTKEY 續	下列其他動作可以在軟鍵區塊中定義： • **picturealignment** 用這個屬性指定影像方向。 根據預設使用軟鍵左側調整影像。 下列值可於對齊時被指定： – **top** 上方邊緣 – **bottom** 下方邊緣 – **left** 左側邊緣 – **right** 右側邊緣 – **center** 置中 • **caption** 軟鍵文字 • **state** 必須搭配 user_controlled 一起使用。 標籤會指派顯示在系統上所需的軟鍵。 **語法：** <state type="<state>" /> 您可指定下列字串： – **notpressed** 未按下時顯示軟鍵。 – **pressed** 按下時顯示軟鍵。 – **disabled** 軟鍵鎖定並以灰色顯示。 • **navigation** • **update_controls** • **function**

標籤識別碼	意義
SOFTKEY 續	**語法：** 標準軟鍵： <state type="<softkey state>" /> <softkey position = "<1>"> … … </softkey> 或 指令碼控制的軟鍵： <softkey position = "<1>" type="<user_defined>" > <state type="<softkey state>" /> … … </softkey> 或 切換軟鍵： <softkey position = "<1>" type="<toggle_softkey>" refvar="<variable name>" > … … </softkey> **範例：** <let name="define_sk_type" type="string">PRESSED</let> <let name="sk_type">1</let> <softkey POSITION="1" type="user_controled" > <caption>Toggle%nSK</caption> <if> <condition>sk_type == 0 </condition> <then> <op> sk_type = 1 </op> <op> define_sk_type = _T"PRESSED" </op> </then> <else> <op> define_sk_type = _T"NOTPRESSED" </op> <op> sk_type = 0 </op> </else> </if> <state type="$$$define_sk_type" /> … … </softkey>

標籤識別碼	意義
SOFTKEY 續	**範例：** 或 <let name="curr_softkey_state" type="string">PRESSED</let> </softkey> <softkey POSITION="3" type="toggle_softkey" refvar="curr_softkey_state"> <caption>Toggle%nSK</caption> … </softkey>  
SOFTKEY_OK	本標籤定義軟鍵「確定」的反應。  下列其他動作可以在軟鍵區塊中定義： • **navigation** • **update_controls** • **function** **語法：** <SOFTKEY_OK> … … </SOFTKEY_OK>

標籤識別碼	意義

SOFTKEY_CANCEL	本標籤定義軟鍵「取消」的反應。
	 
	
	下列其他動作可以在軟鍵區塊中定義：
	• **navigation**
	• **update_controls**
	• **function**
	語法：
	`<SOFTKEY_CANCEL>`
	…
	…
	`</SOFTKEY_CANCEL>`

SOFTKEY_BACK	本標籤定義軟鍵「返回」的反應。
	 
	
	下列其他動作可以在軟鍵區塊中定義：
	• **navigation**
	• **update_controls**
	• **function**
	語法：
	`<SOFTKEY_BACK>`
	…
	…
	`</SOFTKEY_BACK>`

標籤識別碼	意義

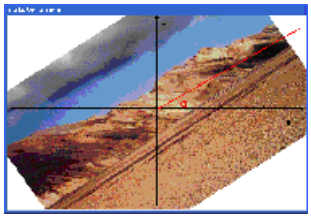
SOFTKEY_ACCEPT	本標籤定義軟鍵「接受」的反應。
	 
	
	下列其他動作可以在軟鍵區塊中定義：
	• **navigation**
	• **update_controls**
	• **function**
	語法：
	`<SOFTKEY_ACCEPT>`
	…
	…
	`</SOFTKEY_ACCEPT>`

TEXT	此標籤用來在指定的位置顯示文字。 若警報編號已使用，對話方塊會顯示以該編號儲存的文字。 **語法：** `<TEXT xpos = "<X 位置>" ypos = "<Y 位置>"> Text </TEXT>` **屬性：** • **xpos** 左上角的 X 位置 • **ypos** 左上角的 Y 位置 • **color** 文字色彩 **更多資訊**請參閱章節「色彩編碼 (頁 81)」 **值：** 待顯示的文字

標籤識別碼	意義

IMG	此標籤用來在指定的位置顯示影像。支援 BMP 與 PNG 影像格式。
	
	語法：
	`<IMG xpos = "<X 位置>" ypos = "<Y 位置>" name = "<名稱>" />`
	`<IMG … xrot="X 軸角度" yrot="Y 軸角度" zrot="Z 軸角度" />`
	
	屬性：
	• **xpos**
	左上角的 X 位置
	• **ypos**
	左上角的 Y 位置
	• **name**
	完整的路徑名稱。路徑資訊以小寫字母表示。
	• **transparent**
	點陣圖的透明色
	更多資訊請參閱章節「色彩編碼 (頁 81)」

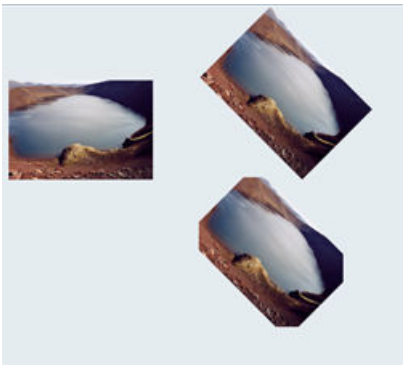
標籤識別碼	意義

IMG 續	**範例**：
	影像繞著 Z 軸旋轉 34 度：
	`<img xpos = "5" ypos = "23" name = "f:/appl/pic3.bmp" height="355" width="550" zrot="34" color_bk="#ffffff" ></img>`
	
	附註：
	驅動器名稱因系統而異。
	 
	
	選項：
	如果顯示的圖像與原始大小不同，尺寸可用屬性 **width** 及 **height** 定義。
	• **width**
	寬度（以像素表示）
	• **height**
	高度（以像素表示）
	
	範例：
	`<IMG xpos = "20" ypos = "40" name = "f:/appl/test.bmp" />`
	`<IMG xpos = "5" ypos = "23" name = "f:/appl/test.bmp" height = "355" width = "550"/>`

標籤識別碼	意義

IMG 續	**屬性：**
	AspectRatioMode
	此屬性允許在執行非比例縮放的任何地方，控制影像的寬高比。
	
	值：
	• **Ignore**
	點陣圖的寬高比適合於規定的高度與寬度。
	• **Keep**（預設值）
	保留寬高比並確保將影像縮放為矩形，該矩形在規定的高度與寬度內，呈現最大的擴展。
	• **KeepByExpanding**
	保持寬高比並確保影像縮放為矩形，該矩形在規定的高度與寬度之外，呈現最大的擴展。
	
	範例：
	`<img xpos = "8" ypos = "30" name = "test.bmp" height="155" width="150" AspectRatioMode="KeepbyExpanding">`
	`  <property transparent="#00ff00" />`
	`</img>`
	`<img xpos = "8" ypos = "200" name = "test.bmp" height="155" width="150" AspectRatioMode="Ignore">`
	`  <property transparent="#00ff00" />`
	`</img>`
	`<img xpos = "200" ypos = "200" name = "test.bmp" height="155" width="150" AspectRatioMode="Keep">`
	`  <property transparent="#00ff00" />`
	`</img>`
	 
	
	• 頂端 - **KeepbyExpanding** 屬性集
	• 左下角 - **Ignore** 屬性集
	• 右下角 - **Keep** 屬性集

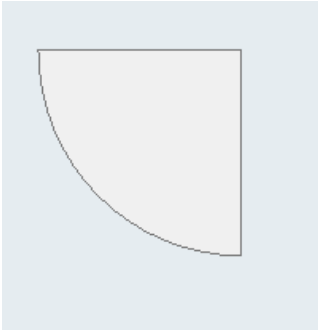
標籤識別碼	意義

IMG 續	**屬性：**
	ExpandingForRotation
	保持寬高比。將影像縮放為矩形，該矩形對應於旋轉及縮放影像的邊界框。
	
	範例：
	`<img xpos = "8" ypos = "100" name = "pic1.bmp" height="155" width="150" >`
	`  <property transparent="#ffffff" />`
	`</img>`
	`<op>`
	`  zrot = 45.0;`
	`</op>`
	`<img xpos = "205" ypos = "23" name = "pic1.bmp" height="155" width="150" zrot="$zrot" AspectRatioMode="ExpandingForRotation">`
	`  <property transparent="#ffffff" />`
	`</img>`
	`<img xpos = "205" ypos = "200" name = "pic1.bmp" height="155" width="150" zrot="$zrot" >`
	`  <property transparent="#ffffff" />`
	`</img>`
	 
	• 左 - 影像縮放到 150x155 像素大小
	• 右上角 - 影像縮放到 150x155 像素大小，並使用 **ExpandingForRotation** 屬性旋轉 45 度左右
	• 右下角 - 影像縮放到 150x155 像素大小，並旋轉 45 度左右

標籤識別碼	意義
ARC	此標籤用來繪製格式中的圓形區段。 **屬性：** 以像素為單位，格式內的位置： • **xpos1**- 圓形區段 X 座標的起點 • **ypos1**- 圓形區段 Y 座標的起點 • **xpos2**- 圓形區段 X 座標的終點 • **ypos2**- 圓形區段 Y 座標的終點 以像素為單位，格式內的位置： • **ccx**- 圓形區段 X 座標的中心 • **ccy**- 圓形區段 Y 座標的中心 • **radius** - 圓形半徑，以像素為單位的值 **附註：** **中心點或半徑：** 如果同時指定了兩個參數，則使用圓心。
ARC 續	• **penwidth** 該屬性指定了以像素為單位的線條寬度。 • **color** 線條色彩 • **color_bk** 圓形區段的填充色彩 **更多資訊**請參閱章節「色彩編碼 (頁 81)」 • **style** 屬性定義線條類型： – **"SolidLine"** - 實線（預設值） – **"DashLine"** - 虛線 – **"DotLine"** - 點線 – **"DashDotLine"** - 點虛線 – **"DashDotDotLine"** - 雙點虛線

標籤識別碼	意義
ARC 續	• **type** – **cw** - 順時針 – **ccw** - 逆時針 • **arrow** 箭頭顯示於區段尾端上： – **"P1"** - 箭頭在線條的起點 – **"P2"** - 箭頭在線條的起點 – **"P1P2"** - 箭頭在線條的起點及終點 – **"P1_FILLED"** - 實心箭頭在線條的起點 – **"P2_FILLED"** - 實心箭頭在線條的起點 – **"P1P2_FILLED"** - 實心箭頭在線條的起點及終點 • **arrow_size** 如果使用 Arrow 屬性，則可以以像素為單位指定箭頭的長度。

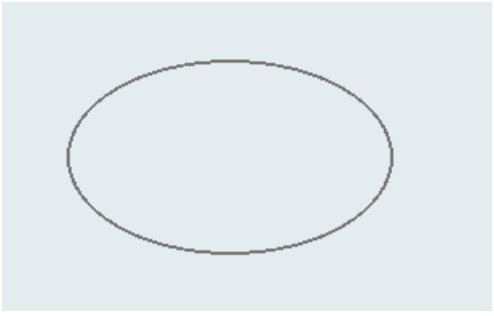
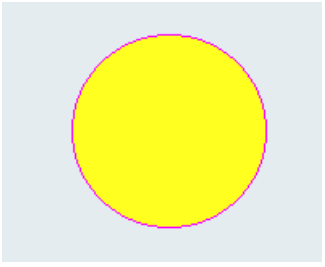
標籤識別碼	意義

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

標籤識別碼	意義
BOX	此標籤於指定的位置畫出矩形，並填上指定顏色。 **語法：** `<BOX xpos = "<X 位置>" ypos = "<Y 位置>" width = "<X 延伸>" height = "<Y 延伸>" color = "<色彩代碼>" />` **屬性：** • **xpos** 左上角的 X 位置 • **ypos** 左上角的 Y 位置 • **width** X 方向延伸（以像素表示） • **height** Y 方向延伸（以像素表示） • **color** 色彩編碼 **更多資訊**請參閱章節「色彩編碼 (頁 81)」

標籤識別碼	意義
ELLIPSE	此標籤用來繪製格式中的橢圓。 **屬性：** 以像素為單位，格式內的位置： • **xpos** - X 座標線條的起點 • **ypos** - Y 座標線條的起點 • **width** - 環繞矩形的寬度 (bounding box) • **height** - 環繞矩形的高度 (bounding box) • **penwidth** 該屬性指定了以像素為單位的線條寬度。 • **color** 線條色彩 • **color_bk** 橢圓的填充色彩 **更多資訊**請參閱章節「色彩編碼 (頁 81)」 • **style** 屬性定義線條類型： – **"SolidLine"** - 實線（預設值） – **"DashLine"** - 虛線 – **"DotLine"** - 點線 – **"DashDotLine"** - 點虛線 – **"DashDotDotLine"** - 雙點虛線

標籤識別碼	意義

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

標籤識別碼	意義

FUNCTION	**函數呼叫**
	此標籤執行該屬性「**name**」指定的函數主體。
	
	屬性：
	• **name** = "函數主體名稱"
	• **return** = "儲存函數結果的變數名稱"
	
	值：
	待轉換至函數主體的變數清單。變數必須以逗號分開。最大可傳輸 10 個參數。
	也可以指定常數或文字表示式作為呼叫參數。識別碼 _T 應置於起始的位置，藉以識別文字表示式。
	
	語法：
	`<FUNCTION name = "<function name>" />`
	呼叫函數預期會有傳回值
	`<FUNCTION name = "<function name>" return = "<Variablenname>" />`
	參數傳輸
	`<FUNCTION name = "<function name>"> var1, var2, var3 </FUNCTION>`
	`<FUNCTION name = "<function name>"> _T"Text", 1.0, 1 </FUNCTION>`
	
	範例：
	請參閱「FUNCTION_BODY」

標籤識別碼	意義

FUNCTION_BODY	**函數主體**
	此標籤包含次函數的函數主體。此函數主體必須在 DialogGui 標籤中程式設計。
	
	屬性：
	• **name** = "函數主體名稱"
	• **parameter** = "參數清單"（選配）
	此屬性列出所需的傳輸參數。參數必須以逗號分開。
	呼叫函數主體時，函數呼叫中指定的參數值會複製到表列的傳輸參數。
	• **return** = "true"（選配）
	若屬性設定為 true 則建立區域變數 $return。發送到呼叫函數，用以退出該函數的函數傳回值必須複製到這個變數。
	
	語法：
	
	無參數的函數主體
	`<FUNCTION_BODY name = "<function name>">`
	…
	…
	…
	`</ FUNCTION_BODY>`
	
	有參數的函數主體
	`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>">`
	…
	`<LET name = "tmp></LET>`
	`<OP> tmp = p1 </OP>`
	…
	`</FUNCTION_BODY>`
	
	有傳回值的函數主體
	`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>" return = "true">`
	…
	`<LET name = "tmp></LET>`
	`<OP> tmp = p1 </OP>`
	…
	`<OP> $return = tmp </OP>`
	`</FUNCTION_BODY>`

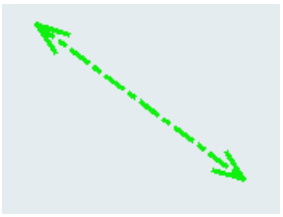
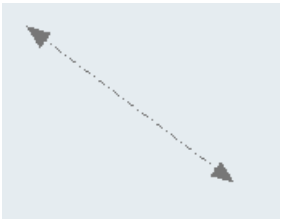
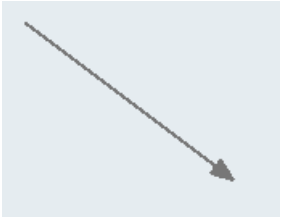
標籤識別碼	意義

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

LINE	此標籤用來繪製格式中的線條。 **屬性：** 以像素為單位，格式內的位置： • **xpos1 -** X 座標線條的起點 • **ypos1** - Y 座標線條的起點 • **xpos2** - X 座標線條的終點 • **ypos2** - Y 座標線條的終點 • **penwidth** 該屬性指定了以像素為單位的線條寬度。 • **color** 線條色彩 **更多資訊**請參閱章節「色彩編碼 (頁 81)」 • **style** 屬性定義線條類型： – **"SolidLine"** - 實線（預設值） – **"DashLine"** - 虛線 – **"DotLine"** - 點線 – **"DashDotLine"** - 點虛線 – **"DashDotDotLine"** - 雙點虛線

標籤識別碼	意義
LINE 續	• **arrow** 箭頭顯示於線條尾端上： – **"P1"** - 箭頭在線條的起點 – **"P2"** - 箭頭在線條的起點 – **"P1P2"** - 箭頭在線條的起點及終點 – **"P1_FILLED"** - 實心箭頭在線條的起點 – **"P2_FILLED"** - 實心箭頭在線條的起點 – **"P1P2_FILLED"** - 實心箭頭在線條的起點及終點 • **arrow_size** 如果使用 Arrow 屬性，則可以以像素為單位指定箭頭的長度。 **語法：** `<line xpos1="<X 座標起點>" ypos1="<Y 座標起點>" xpos2="<X 座標終點>" ypos2="<Y 座標起點>" PENWIDTH="<線條寬度>" color="<線條色彩>" style="<線條類型>" arrow="<箭頭類型>" arrow_size="<箭頭大小>"/>`

標籤識別碼	意義

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

標籤識別碼	意義

REQUEST	此標籤用來將變數加到循環讀取服務（熱連結）。其結果是未連結至控制器的變數存取時間縮短。
	若要在數值變更時自動呼叫函數，則應將函數名稱指定為其他屬性。
	此標籤僅在 INIT 操作中處理。
	
	屬性：
	• **name**
	位址識別碼
	• **function**
	函數名稱
	
	語法：
	`<REQUEST name = "<NC-Variable>" />`
	或
	`<REQUEST name = "<NC-Variable>" function="<function name>"/>`
	
	範例：
	`<request name ="plc/mb10" />`
	
	或
	
	`<function_body name="my_function" >`
	`<print text="value changed" />`
	`</function_body>`
	…
	…
	…
	`<request name ="plc/mb10" function="my_function"/>`

標籤識別碼	意義

RECALL	若將透過 Recall 按鈕啟用導覽，則可在功能表中使用此標籤。若此標籤經過程式設計，則顯示 Recall 圖示，語法分析器可以處理 Recall 按鈕。
	
	語法：
	`<recall>`
	…
	…
	`</recall>`

UPDATE_CONTROLS	此標籤執行操作者控制與參考變數之間的比較作業。
	
	屬性：
	• **type**
	此屬性定義資料比較的方向。
	= TRUE – 資料由參考變數讀取並複製到操作者控制。
	= FALSE – 資料由操作者控制複製到參考變數。
	
	語法：
	`<UPDATE_CONTROLS type = "<方向>"/>`
	
	範例：
	`<SOFTKEY_OK>`
	`< UPDATE_CONTROLS type="false"/>`
	`</SOFTKEY_OK>`

3.8 建立使用者功能表

3.8.1 建立處理循環格式

循環支援函數透過對話方塊格式允許循環呼叫的自動建立及取消編譯。

為了管理這項功能，可使用下列標籤：

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

若欲標記循環形式，必須將 **FORM** 標籤中的屬性 **type** 指定為數值 **cycle**。這項標記允許 **NC_INSTRUCTION** 進行處理。

範例

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

標籤 **NC_INSTRUCTION** 包含所要產生的循環呼叫。所有的循環參數均應使用空間固定器保留。

範例

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

標籤 **CREATE_CYCLE** 會準備儲存在空間固定器變數中的數值，並產生 NC 指令。

然後這會複製到指定變數。

標籤識別碼	說明

NC_INSTRUCTION	此標籤用於定義所要產生的 NC 指令。
	所有列出的循環參數都自動建立，成為 **FORM** 的字串變數，並提供給 **FORM**。
	
	先決條件： FORM 屬性 **type** 設定為數值 **CYCLE**。
	
	屬性：
	• **refvar**
	如果指派參考變數給標籤，則儲存在參考變數 NC 區塊中的數值，會預先指派給所有的參數。
	語法：
	`<NC_INSTRUCTION> 具有空間固定器的 NC 指令 </ NC_INSTRUCTION>`
	
	範例：
	`<let name="cyc_string" type="string"> Cycle100(0, 1000, 5)</ let>`
	…
	…
	…
	`<FORM name = "cycle100_form" type= "CYCLE">`
	`<NC_INSTRUCTION refvar= "cyc_string" >Cycle100($p1, $p2, $p3)</ NC_INSTRUCTION>`
	…
	…
	…
	`</FORM>`

標籤識別碼	說明

CREATE_CYCLE	標籤產生 NC 區塊，其語法係由 **NC_INSTRUCTION** 標籤的數值定義。
	產生 NC 指令之前，語法分析器呼叫 FORM 的 CYCLE_CREATE_EVENT 標籤。可使用此標籤計算循環參數。
	語法：
	`<CREATE_CYCLE/>`
	選項：
	若已指定參考變數，指令會將所產生的呼叫複製至此變數。
	`<create_cycle refvar="name" />`
	屬性：
	• **refvar**
	若是標籤指派了參考變數，NC 指令將複製至此變數。
	範例：
	`<LET name="cyc_string" type="string"> Cycle100(0, 1000, 5)</LET>`
	`<SOFTKEY_OK>`
	`<caption>OK</caption>`
	`<CREATE_CYCLE />`
	`<close_form />`
	`<navigation>main_menu</navigation>`
	`</SOFTKEY_OK>`
	或
	`<SOFTKEY_OK>`
	`<caption>OK</caption>`
	`<CREATE_CYCLE refvar= "cyc_string" />`
	`<close_form />`
	`<navigation>main_menu</navigation>`
	`</SOFTKEY_OK>`

3.8.2 替代字元

系統提供針對執行時間定義控制內容（屬性值）的選項。要使用這項功能，必要的內容需在區域變數中設定，此變數名稱必須轉換成為以字元 **$** 為首的屬性值的標籤

如果標籤預期字串為一屬性值或是數值，必須在變數名稱之前加上 $$$ 字元。

範例：

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

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

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

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

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

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

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

3.9 建立檔案 struct_def.xml

程序

在某些函數中，可以使用檔案 **struct_def.xml** 進行類型定義。如果遺失這個 XML 檔案，請使用以下標記來建立檔案副本，並將其整合至功能專案中。

說明

以 UTF8 格式建立 XML 檔案。

命令集

struct_def.xml

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

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

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

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

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

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

3.10 定址元件

必須建立所要資料的位址辨識碼，以定址 NC 變數、PLC 區塊、或驅動資料。 位址包含次路徑**元件名稱**與**變數位址**。 斜線必須使用，作為分隔字元。

3.10.1 PLC 定址

以路徑區段 **plc** 為開頭定址 PLC

表格 3-3　允許使用下列位址：

DBx.DB(f)	資料區塊
I(f)x	輸入
Q(f)x	輸出
M(f)x	位元記憶體
V(f)x	變數

DBx.DBXx.b	資料區塊
Ix.b	輸入
Qx.b	輸出
Mx.b	位元記憶體
Vx.b	變數

表格 3-4　資料格式 **f**：

B	位元組
W	字組
D	雙字組

資料格式辨識不適用於位元定址。

位址 **x**：

有效的 S7-200 位址辨識碼

位元定址：

b －位元編號

範例：

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

3.10.2 NC 變數定址

以路徑區段 **nck** 為開頭定址 NC 變數

本區段後面接著資料位址；結構必須符合「參數手冊 NC 變數與介面訊號」所述。

範例：

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

3.10.3 通道專屬定址

若位址符記未定義通道編號，將維持存取操作軟體在通道 1。

若有必要從專屬通道讀取資料，則將含有所需通道編號的識別碼 **u**（單元）新增至位址。

範例：

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

3.10.4 在執行時間期間產生 NC/PLC 位址

在執行時間可以產生位址識別碼。

在此情況下，可以在操作敘述、以及 nc.cap.read 與 nc.cap.write 功能中，使用字串變數的內容作為位址。

此類型的定址模式請遵守下列事項：

- 使用引號寫入變數名稱。
- 使用三個 ‚$'字元作為變數名稱的前置碼。

語法：

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

範例：

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

3.10.5 定址驅動元件

以路徑區段 **drive** 為開頭定址驅動元件。

然後指定驅動元件：

CU – 控制單元

DC – 驅動器控制（馬達模組）

CULNK – 擴充模組 (HUB)

TM – 端子模組

LM – 電源模組

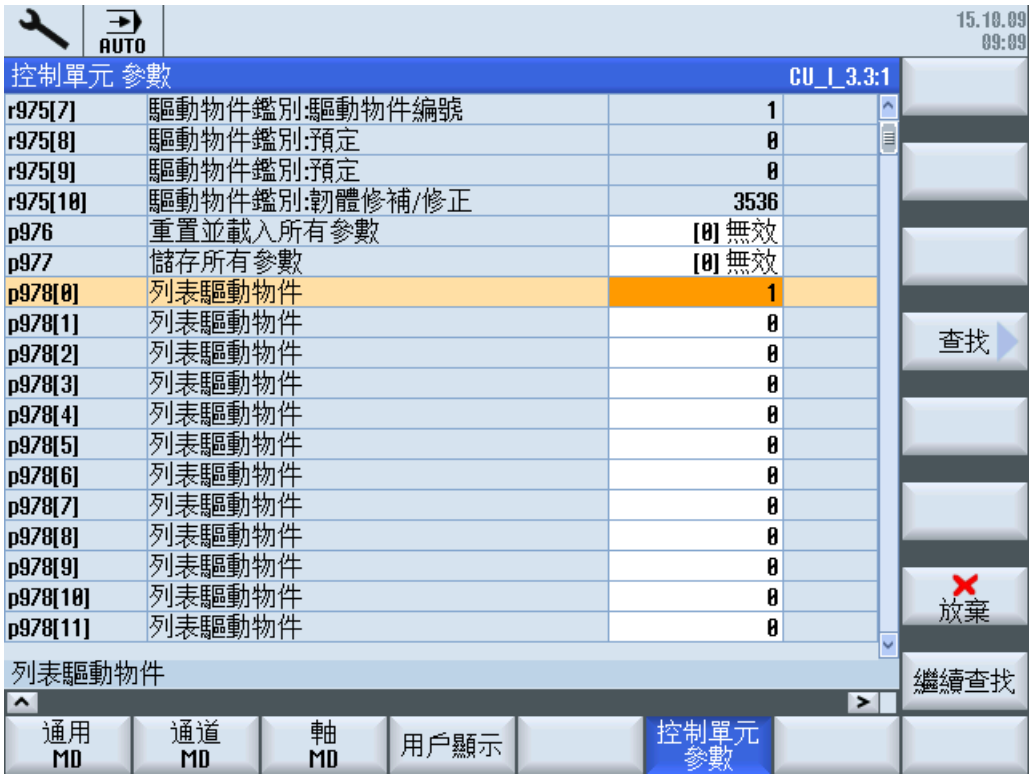
將待設定的參數加入這個區段：

範例：

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

<!– Cu 的 r0002 數值讀數 ->

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

<!– NX1 的 r0002 數值讀數 ->
<OP> r0002_content = "drive/cu/r0002[CU2]" </OP>

<!– Cu 的 p107[0] 數值讀數 ->
<OP> p107_content = "drive/cu/p107[0]" </OP>
<PRINT text="%d"> p107_content </PRINT>

<!– Cu 的 p107[0] 數值讀數 ->

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

<!– NX1 的 p107[0] 數值讀數 ->

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

驅動物件的定址：

要將個別物件定址，需要的物件必須輸入到參數後面的方括號內。



圖像 3-7　控制器的驅動器參數 p0978 [...]

參數編號[do<DO-index>]

範例：

```
p0092[do1]
```

此外，可以使用 **$<variable name>**「substitution characters」從個區域變數讀取驅動器指標。

```
z.B. DO$區域變數
```

範例：

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

間接定址：

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

3.10.6 範例：決定馬達模組的 DO 編號

類型 11（伺服）馬達模組的 DO 編號可以決定如下：

所有連接的驅動物件在相關 CU 欄位 p978 中與對應的插槽編號一併表列。元件類型編號表列在欄位 p101，而元件類型則同時表列在欄位 p107 中。

下列元件類型各別使用分隔索引：

CU – 控制單元

DC – 驅動器控制 （馬達模組）

CULNK – 擴充模組 (HUB)

TM – 端子模組

LM – 電源模組

針對每個已連接的 CU，依照遞增順序執行欄位 p107，則可決定定址索引，而每次發生所需的類型時，類型索引會隨之增量一個單位。基本值為 1。若在此欄位中找到 NX 元件，則 NX 元件的計數不再繼續，直到目前的陣列完全執行完畢為止。NX 及 CU 元件將依照所找到的順序執行。

索引的判定：含 NX 的 CUI

CUI		NX1		定址索引	
				DO	CU
p107[0]	[3]SINAMICS				1
p107[2]	[11]SERVO			1	
p107[3]	[11]SERVO			2	
p107[4]	[11]SERVO			3	
p107[5]	[254]CU-LINK				2
p107[6]	[11]SERVO			4	
		p107[1]	[11]SERVO	5	
		p107[2]	[11]SERVO	6	
		p107[3]	[11]SERVO	7	

此拓樸結構包含七個馬達模組。索引 1 至 4 用來定址指派給 CU 的馬達模組。索引 5 至 7 則用來定址 NX 的馬達模組。使用索引 1 來存取 CU。NX 藉由索引定址二次。

命令集範例：xmldial.xml 及 drv_sys_hlpfunct.xml

xmldial.xml

```
<DialogGui>

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

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

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

    <op>
     cui_idx = 0;
    </op>

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

    <function name="calculate_do_index" />

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

    <op>
     count = 0;
    </op>

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

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

    <function name="read_do_name_fast" return="do_name">count</function>
```

xmldial.xml

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

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

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

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

 </form>
</DialogGui>
```

drv_sys_hlpfunct.xml

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

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

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

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

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

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

 output:
 componentsList structure

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

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

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

 <print text="gather data" />
```

drv_sys_hlpfunct.xml

```
 <function name="ncfunc.cap.read" return="error"
rows="$count">componentsList[$cuno].cu_p978, "drive/cu/p0978[0, cu
$cuidx]"</function>
 <function name="ncfunc.cap.read" return="error"
rows="$count">componentsList[$cuno].do_p0101, "drive/cu/p0101[0, cu
$cuidx]"</function>
 <function name="ncfunc.cap.read" return="error"
rows="$count">componentsList[$cuno].do_p0107, "drive/cu/p0107[0, cu
$cuidx]"</function>

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

 <op>
  count = 0;
 </op>

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

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

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

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

 input:
 -

 output:
 componentsList array filled
```

drv_sys_hlpfunct.xml

```
 ------------------------------------------------------------------ -->

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

3.10.7 機台及設定資料的定址

驅動器與設定資料使用字元 $ 後面接著資料名稱辨識。

機台資料：

$Mx_<name[**index, AX<axis_number>**]>

設定資料：

$Sx_<name[index, AX<axis_number>]>

x：

N －一般機台或設定資料

C －通道專屬機台或設定資料

A －軸專屬機台或設定資料

索引：

對區域而言，此參數代表資料的索引。

AX<axis_number>：

需要的軸（**<axis_number>**）必須指定軸專屬資料。

此外，可以使用 **$<variable name>**「substitution characters」從區域變數讀取軸指標。

例如 AX$localvariable

範例：

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

軸的直接定址：

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

軸的間接定址：

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

3.10.8 通道專屬機械資料

若位址符記未定義通道編號，將維持存取操作軟體目前設定的通道。

若有必要從專屬通道讀取資料，則將方括號內含有所需通道編號的識別碼 **u**（單元）新增至位址。在陣列定址中，通道定義是方括號內的最後一個引數。

範例：

```
$MC_RESET_MODE_MASK
```

或

```
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]
```

```
$MC_RESET_MODE_MASK[u1]
```

或

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

3.10.9 使用者資料的定址

對使用者資料的定址是從路徑部分 **gud** 起始，然後透過指示 GUD 是否為全域，或是通道專屬的標記。位址部分後面接著 GUD 區及 GUD 名稱。

在欄位的名稱之後，所需的欄位索引應以方括號指定。

範例：

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

全域使用者資料定址

定址由路徑區段 gud 起始，後面接著區域 NCK 規格。位址區段後面接著是 GUD 區的規格：

GUD 區域	配置
sgud	Siemens GUD
mgud	機具製造商 GUD
ugud	使用者 GUD

通道專屬使用者資料定址

定址由路徑區段 gud 起始，後面接著區域 CHANNEL 規格。位址區段後面接著是 GUD 區的規格。

然後輸入 GUD 名稱。若陣列需要定址，名稱
後面要接著內有陣列索引的方括號。

範例：

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

定址多維陣列

定址多維陣列時，行索引預期後面接著欄索引。輸入的索引應以逗號逐一區隔。

以下適用於通道專屬 GUD：

- 通道編號必須註記記號 **u**（單位），接著是行/欄規格之前或之後的編號。
- 順序必須以逗號分開。
- 如未輸入通道編號，將存取第一個通道。

範例：

```
「gud/Channel/sgud/_WP[u2, 2.0]」
```

或

```
「gud/Channel/sgud/_WP[2.0, u2]」
```

3.11 預先定義之函數

此指令集語言提供各種字串處理與標準數學功能。下列函數名稱已保留，不能重載。

函數名稱	意義

Ncfunc cap read	函數從指定位址將數值複製到區域變數中。如果讀取操作未發生錯誤，則返回的變數包含數值零。
	
	不同於操作指令，發生故障時此項函數並不會中斷指令碼操作的處理。
	
	屬性：
	• **return** - 執行狀態
	– 數值 = 0 - 無故障
	– 數值 = 1 - 無法讀取變數
	• **rows** - 待讀取陣列的額外行數（選擇性）
	若已針對參考變數而定義了陣列索引（如同這個索引所示），函數會複製已讀入目標變數中的值。
	
	語法：
	`<function name="ncfunc.cap.read" return="error"> lokale variable, "address"</function>`
	
	範例：
	`<let name="error"></let>`
	`<function name="ncfunc.cap.read" return="error"> 3, "drive/cu/p0009"</function>`
	`<if>`
	`<condition>error != 0</condition>`
	`<then>`
	`<break />`
	`</then>`
	`</if>`
	
	或
	
	`<let name="cu_p978" dim="25" ></let>`
	`...`
	`...`
	`<function name="ncfunc.cap.read" return="error" rows="23">cu_p978, "drive/cu/p0978[0, 1]"`
	`</function>`

函數名稱	意義

Ncfunc cap write	函數會將數值寫入指定的變數。如果寫入操作未發生錯誤，則返回的變數包含數值零。
	
	不同於操作指令，發生故障時此項函數並不會中斷指令碼操作的處理。
	
	屬性：
	• **return** - 執行狀態
	– 數值 = 0 - 無故障
	– 數值 = 1 - 無法讀取變數
	• **rows** - 待寫入陣列的額外行數（選擇性）
	若已針對參考變數而定義了陣列索引（如同這個索引所示），函數會將數值複製到目標變數中。
	
	語法：
	`<function name="ncfunc.cap.read" return="error"> local variable or constant, "address"</function>`
	
	範例：
	`<let name="error"></let>`
	`<function name="ncfunc.cap.write" return="error"> 0, "drive/cu/p0009"</function>`
	`<if>`
	`<condition>error != 0</condition>`
	`<then>`
	`<break />`
	`</then>`
	`</if>`
	
	或
	
	`<let name="cu_p978" dim="25" ></let>`
	`...`
	`...`
	`<function name="ncfunc.cap.write" return="error" rows="23">cu_p978, "drive/cu/p0978[0, 1]"`
	`</function>`

函數名稱	意義

Ncfunc PI-Service	工作可以使用程式叫用服務 (PI) 傳輸至 NCK。
	如果服務執行未發生錯誤，函數會在返回變數中返回數值 1。
	
	操縱刀具清單：
	_N_CREATO - 建立刀具
	_N_DELETO - 刪除刀具
	_N_CREACE - 建立刀具刀刃
	_N_DELECE - 刪除刀具刀刃
	
	啟用工件偏移量：
	_N_SETUFR - 啟用實際使用者框架
	_N_SETUDT - 啟用實際使用者資料
	
	單節搜尋：
	_N_FINDBL - 啟用單節查找
	_N_FINDAB - 取消單節查找
	
	重新設定機械參數：
	_N_CONFIG - 含啟動等級的機械參數 NEW_CONF，其中操作員按順序依次輸入，同步啟動
	
	語法：
	`<function name="ncfunc.pi_service" return="return var"> pi name, var1, var2, var3, var4, var5`
	`</function>`
	
	屬性：
	name - 函數名稱
	return- 變數名稱，執行結果將儲存：
	• 數值 == 1 – 工作執行成功
	• 數值 == 0 – 工作執行錯誤
	
	標籤值：

函數名稱	意義
	pi name - PI 服務的名稱（字串） var1 至 var5 - PI 特定引數

函數名稱	意義

Ncfunc PI-Service 續	**引數：**
	• _N_CREATO
	var1 - 刀具編號
	• _N_DELETO
	var1 - 刀具編號
	• _N_CREACE
	var1 - 刀具編號
	var2 - 刀刃編號
	• _N_DELECE
	var1 - 刀具編號
	var2 - 刀刃編號
	• _N_SETUFR
	無引數
	• _N_SETUDT
	var1- 將啟用的使用者資料區
	– 1 - 刀具偏移量資料
	– 2 - 啟用的基本框架
	– 3 - 啟用的可調整框架
	• _N_FINDBL
	var1 - 查找模式
	– 2 - 以輪廓計算查找
	– 4 - 查找單節結束點
	– 1 - 不以計算方式進行單節查找
	• _N_FINDAB
	無引數
	• _N_CONFIG
	無引數
	
	範例：
	建立刀具 – 刀具編號 3
	
	`<function name="ncfunc.pi_service">_T"_N_CREATO", 3</function>`
	
	刪除刀具 5 的刀刃 1
	
	`<function name="ncfunc.pi_service">_T"_N_DELECE", 5, 1`
	`</function>`

函數名稱	意義

ncfunc chan PI-Service	此函數以通道相關的方式執行 PI 服務。通道編號將接在 PI 服務名稱之後進行傳遞，後面接著所有其他呼叫參數。
	
	參數：
	channel - 通道編號
	
	語法：
	`<function name="ncfunc.chan_pi_service" return="error">`
	`_T"_N_SETUFR", channel, ...`
	`</function>`
	
	範例：
	`<let name="chan" >1</let>`
	`<function name="ncfunc.chan_pi_service" return="error">`
	`_T"_N_SETUFR", chan</function>`
	
	`<function name="ncfunc.chan_pi_service" return="error">`
	`_T"_N_SETUDT", chan, _T"016", _T"00000", _T"00000"</function>`

Ncfunc display resolution	此函數提供控制器中定義之浮點編號的轉換規則。字串變數必須提供為變數。
	
	請同時參閱顯示的機械參數 MD203 DISPLAY_RESOLUTION 及 MD204 DISPLAY_RESOLUTION_INCH
	
	語法：
	`<function name="ncfunc.displayresolution" return="dislay_res" />`
	
	範例：
	`<let name="dislay_res" type="string"></let>`
	...
	`<function name="ncfunc.displayresolution" return="dislay_res" />`
	
	`<control name = "cdistToGo" xpos = "210" ypos = "156" refvar="nck/Channel/GeometricAxis/progDistToGo[2]" hotlink="true" height="34" fieldtype="readonly" format="$$$dislay_res" time="superfast" color_bk="#ffffff"/>`

函數名稱	意義

Ncfunc Get drive by axis name	此函數藉由指定相關的軸名稱，回傳馬達模組的定址索引。
	若無法確定指派，則功能傳回值 -1。例如，如果尚未進行軸/驅動器指派，則可能發生這種情況。
	
	參數：
	str - 軸名稱
	
	傳回值：
	馬達模組的索引 - 寫入已判定索引的變數名稱。
	
	語法：
	`<function name="NCFUNC.GETDRVBYAX_NAME" return="<index>">`
	`<axis name></function>`
	
	範例：
	`<let name="drv_idx" />`
	`<function name="NCFUNC.GETDRVBYAX_NAME" return="drv_idx">_T"X1"`
	`</function>`
	`<control name = "c_do2" xpos = "140" ypos = "114" width = "320"`
	`fieldtype="itemlist" refvar="drive/dc/p10[`**`do$ drv_idx`**`]"`
	`hotlink="true" color_bk="#f1f1f1">`
	` <item value="0">Ready</item>`
	` <item value="1">Quick commissioning</item>`
	` <item value="2">Power unit commissioning</item>`
	` <item value="3">Motor commissioning</item>`
	` <item value="4">Encoder commissioning</item>`
	` <item value="5">Technological application/units`
	`</item>`
	` <item value="15">Data sets</item>`
	` <item value="17">Basic positioning commissioning`
	`</item>`
	` <item value="25">Commissioning the position control</item>`
	` <item value="29">Download</item>`
	` <item value="30">Parameter reset</item>`
	` <item value="95">Safety Integrated commissioning`
	`</item>`
	`</control>`
	`</init>`

函數名稱	意義

Ncfunc Get drive by axis index	此函數藉由指定相關的軸索引，回傳馬達模組的定址索引。
	若無法確定指派，則功能傳回值 -1。例如，如果尚未進行軸/驅動器指派，則可能發生這種情況。
	
	參數：
	index - 軸索引
	
	傳回值：
	馬達模組的索引 - 寫入已判定索引的變數名稱。
	
	語法：
	`<function name="NCFUNC.GETDRVBYAX_IDX" return="<index>">`
	`<axis index></function>`
	
	範例：
	`<let name="drv_idx" />`
	`<function name="NCFUNC.GETDRVBYAX_IDX" return="drv_idx">1</function>`
	`<control name = "c_do2" xpos = "140" ypos = "114" width = "320"`
	`fieldtype="itemlist" refvar="drive/dc/p10[`**`do$ drv_idx`**`]"`
	`hotlink="true" color_bk="#f1f1f1">`
	`<item value="0">Ready</item>`
	`<item value="1">Quick commissioning</item>`
	`<item value="2">Power unit commissioning</item>`
	`<item value="3">Motor commissioning</item>`
	`<item value="4">Encoder commissioning</item>`
	`<item value="5">Technological application/units`
	`</item>`
	`<item value="15">Data sets</item>`
	`<item value="17">Basic positioning commissioning`
	`</item>`
	`<item value="25">Commissioning the position control</item>`
	`<item value="29">Download</item>`
	`<item value="30">Parameter reset</item>`
	`<item value="95">Safety Integrated commissioning`
	`</item>`
	`</control>`
	`</init>`

函數名稱	意義

Ncfunc Get drive by drive name	此函數藉由指定馬達模組名稱，回傳馬達模組的定址索引。
	若無法確定指派，則功能傳回值 -1。例如，如果尚未進行軸/驅動器指派，則可能發生這種情況。
	
	參數：
	string - 馬達模組的名稱
	
	傳回值：
	馬達模組的索引 - 寫入已判定索引的變數名稱。
	
	語法：
	`<function name="NCFUNC.GETDRVBYDRV_NAME" return="<index>">`
	`<drive name></function>`
	
	範例：
	`<let name="drv_idx" />`
	`<function name="NCFUNC.GETDRVBYDRV_NAME"`
	`return="drv_idx">_T"SERVO_3.13:4"</function>`
	`<control name = "c_do2" xpos = "140" ypos = "114" width = "320"`
	`fieldtype="itemlist" refvar="drive/dc/p10[`**`do$ drv_idx`**`]"`
	`hotlink="true" color_bk="#f1f1f1">`
	` <item value="0">Ready</item>`
	` <item value="1">Quick commissioning</item>`
	` <item value="2">Power unit commissioning</item>`
	` <item value="3">Motor commissioning</item>`
	` <item value="4">Encoder commissioning</item>`
	` <item value="5">Technological application/units`
	`</item>`
	` <item value="15">Data sets</item>`
	` <item value="17">Basic positioning commissioning`
	`</item>`
	` <item value="25">Commissioning the position control</item>`
	` <item value="29">Download</item>`
	` <item value="30">Parameter reset</item>`
	` <item value="95">Safety Integrated commissioning`
	`</item>`
	`</control>`
	`</init>`

函數名稱	意義

Ncfunc Get drive by bud address	此函數藉由指定馬達模組的匯流排位址，回傳馬達模組的定址索引。
	若無法確定指派，則功能傳回值 -1。例如，如果尚未進行軸/驅動器指派，則可能發生這種情況。
	
	參數：
	bus - 匯流排編號
	slave - 從屬編號
	component - 元件編號
	
	傳回值：
	馬達模組的索引 - 寫入已判定索引的變數名稱。
	
	語法：
	`<function name="NCFUNC.GETDRVBYBUS_ADDR" return="<index>"><bus>,<slave>,<component></function>`
	
	範例：
	`<let name="drv_idx" />`
	`<function name="NCFUNC.GETDRVBYBUS_ADDR" return="drv_idx"> 3,13,4 </function>`
	`<control name = "c_do2" xpos = "140" ypos = "114" width = "320" fieldtype="itemlist" refvar="drive/dc/p10[`**`do$ drv_idx`**`]" hotlink="true" color_bk="#f1f1f1">`
	`<item value="0">Ready</item>`
	`<item value="1">Quick commissioning</item>`
	`<item value="2">Power unit commissioning</item>`
	`<item value="3">Motor commissioning</item>`
	`<item value="4">Encoder commissioning</item>`
	`<item value="5">Technological application/units </item>`
	`<item value="15">Data sets</item>`
	`<item value="17">Basic positioning commissioning </item>`
	`<item value="25">Commissioning the position control</item>`
	`<item value="29">Download</item>`
	`<item value="30">Parameter reset</item>`
	`<item value="95">Safety Integrated commissioning </item>`
	`</control>`
	`</init>`

函數名稱	意義

Ncfunc Get MD limits	該功能將機械參數的極限複製到指定的變數，資料類型為 **StructMDLimits**。結構定義包含在檔案 struct_def.xml 中。
	更多資訊請參閱章節「建立檔案 struct_def.xml (頁 155)」
	
	參數：
	範圍 - 應傳送一個字串以指定機械參數區：
	• **displayMD** - 顯示機械參數
	• **generalMD** - NC 全域機械參數
	• **channelMD** - NC 通道專屬機械參數
	• **axisMD** - NC 軸專屬機械參數
	• **generalSettingMD** - NC 全域設定資料
	• **channelSettingMD** - NC 通道專屬設定資料
	• **axisSettingMD** - NC 軸專屬設定資料
	• **channelGUD** - NC 通道專屬使用者資料
	• **globalGUD** - NC 全域使用者資料
	MD-number - 區域的機械參數編號
	變數名稱 - 功能呼叫之後，此變數包含極限值
	範圍/單位 - 選擇性（例如通道編號）
	
	語法：
	`<function name="NCFUNC.GETMD_LIMITS"><range>, <MD-number, <variabel name>, <range/unit>></function>`
	
	範例：
	`<let name="limits" type="StructMDLimits" /> <function name="NCFUNC.GETMD_LIMITS">_T"generalMD", 19220, _T"limits"</function> <op> upper_BAGS = limits.upperLimit; </op>`

函數名稱	意義

Ncfunc bico to int	此函數將 BICO 格式中指定的字串轉換成整數值（請參閱 SINAMICS）。
	
	語法：
	`<function name="ncfunc.bicotoint" return="integer variable">`
	`bico-string</function>`
	
	範例：
	`<let name="s_np0480_0" type="string"></let>`
	`<let name="i_p0480_0">0</let>`
	
	`<function name="ncfunc.bicotoint" return="i_p0480_0">s_np0480_0`
	`</function>`

Ncfunc int to bico	此函數將整數值轉換為 BICO 格式（請參閱 SINAMICS）。
	
	語法：
	`<function name="ncfunc.inttobico"`
	`return="string variable">integer variable</function>`
	
	範例：
	`<function name="ncfunc.inttobico"`
	`return="s_p0480_0">"drive/dc/p0480[0, DO2]"</function>`

Ncfunc is bico str valid	針對 BICO 格式中指定的字串，此函數會傳回數值零（請參閱 SINAMICS）。
	
	語法：
	`<function name="ncfunc.isbicostrvalid"`
	`return="integer variable">string varaible</function>`
	
	範例：
	`…`
	`<let name="s_np0480_0" type="string"></let>`
	`…`
	`…`
	`<control name = "cp0480_0" xpos = "402"`
	`ypos = "76" hotlink="true" refvar="s_np0480_0" >`
	`<property item_data="4001" />`
	`</control>`
	`…`
	`…`
	`<function name="ncfunc.isbicostrvalid"`
	`return="valid">cp0480_0</function>`

函數名稱	意義

Ncfunc password	此函數可設定或刪除來自 NC 的密碼層級。
	• 設定密碼：
	密碼必須將所要求的密碼層級設定為參數。
	• 刪除密碼：
	空白字串刪除密碼層級。
	
	語法：
	`<function name="ncfunc.password">密碼`
	`</function>`
	
	範例：
	`<let name=「密碼」type="string"></let>`
	
	`<function name="ncfunc.password" > 密碼`
	`</function>`
	`<function name="ncfunc.password" > _T"CUSTOMER"`
	`</function>`
	
	刪除密碼：
	`<function name="ncfunc.password" > _T"" </function>`

Control form color	該函數回傳字串，提供對話方塊的文字或背景色彩。
	更多資訊請參閱章節「色彩編碼 (頁 81)」
	
	範圍：
	• BACKGROUND – 要求背景的顏色值
	• TEXT – 要求文字的顏色值（前景）
	
	語法：
	`<function name="control.formcolor"`
	`return="variable">_T"區域"</function>`
	
	範例：
	`<let name="bk_color" type="string"></let>`
	
	`<function name="control.formcolor"`
	`return="bk_color">_T"BACKGROUND"</function>`

函數名稱	意義
Control local time	此函數以 7 個陣列元件將本機時間複製到欄位中。 變數名稱預期為呼叫參數。 下列項目儲存在陣列元件： • 索引 0 - 年份 • 索引 1 - 月份 • 索引 2 - 工作日 • 索引 3 - 日 • 索引 4 - 時 • 索引 5 - 分 • 索引 6 - 秒 **語法：** <function name ="control.localtime">_T"time_array" </function> **範例：** <!-- index 0 = Year 1 = Month 2 = Day of week 3 = Day 4 = Hour 5 = Minute 6 = Second --> <let name="time_array" dim="7" /> <function name ="control.localtime">_T"time_array" </function>
Control exist	若指定的控制項存在，則功能傳回值 1。 **語法：** <function name="CONTROL.EXIST" return="<return variable>"><control name></function> **範例：** <let name="ret" /> <function name="CONTROL.EXIST" return="ret">_T"edit"</function>

函數名稱	意義

Control is modified	若指定的編輯控制內容已變更，則功能傳回值 1。
	
	語法：
	`<function name="CONTROL.ISMODIFIED"`
	`return="<return variable>"><control name></function>`
	
	範例：
	`<let name="ret" /> <function name="CONTROL.ISMODIFIED"`
	`return="ret">_T"edit"</function>`

Control is visible	若指定的控制為可見，則功能傳回值 1。
	
	語法：
	`<function name="CONTROL.ISVISIBLE"`
	`return="<return variable>"><control name></function>`
	
	範例：
	`<let name="ret" /><function name="CONTROL.ISVISIBLE"`
	`return="ret">_T"edit"</function>`

Control set modified	該函數變更已指定編輯控制的已修改旗標。
	
	參數：
	control name
	• 值 = 1 - 將欄位標記為已變更
	• 值 = 0 - 將欄位標記為未變更
	
	語法：
	`<function name="CONTROL.SETMODIFIED"`
	`return="<return variable>"><control name>,`
	`<Flag></function>`
	
	範例：
	`<let name="b" >1</let>`
	`<let name="ret" />`
	`<control name="edit" xpos="10" ypos="80" width="30" refvar="b"`
	`hotlink = "true" />`
	`<function name="CONTROL.SETMODIFIED"`
	`return="ret">_T"edit", 1</function>`

函數名稱	意義

Control set working area	如果變更**捲動檢視**的內容，例如透過刪除或新增控制，則必須重新計算捲動檢視的可見區域。
	呼叫此函數可以執行計算。
	
	值：
	捲動檢視的名稱
	
	範例：
	`...`
	`<control name="area" xpos="8" ypos="72" width="554" height="320" fieldtype="scrollarea" >`
	`  <control name="test" xpos="20" ypos="40"  width="20" fieldtype="readonly" />`
	`</control >`
	
	`<control name="test1" xpos="20" ypos="80"  width="20" fieldtype="readonly" parent="area"/>`
	`<control name="test2" xpos="20" ypos="100"  width="20" fieldtype="readonly" parent="area"/>`
	
	`<function name="CONTROL.SETWORKINGAREA">_T"area"</function>`
	`...`

函數名稱	意義

比較的字串	從辭典編撰的觀點互相比較兩個字串。
	若兩個字串相同則函數傳回值為零，若第一個字串較小則傳回小於零的值，反之則傳回大於零的值。
	參數：
	str1 - 字串
	str2 - 比較字串
	
	語法：
	`<function name="string.cmp" return ="<int var>" > str1, str2 </function>`
	
	範例：
	`<let name="rval">0</let>`
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string">A brown bear hunts a brown dog.</let>`
	
	`<function name="string.cmp" return="rval"> str1, str2 </function>`
	結果：
	rval= 0

函數名稱	意義

比較的字串
無大小寫的差別
	從辭典編撰的角度比較兩個字串（不分大小寫）。
	若兩個字串相同則函數傳回值為零，若第一個字串較小則傳回小於零的值，反之則傳回大於零的值。
	
	參數：
	str1 - 字串
	str2 - 比較字串
	
	語法：
	`<function name="string.icmp" return ="<int var>" > str1, str2 </function>`
	
	範例：
	`<let name="rval">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string">A brown Bear hunts a brown Dog.</let>`
	`<function name="string.icmp" return="rval"> str1, str2 </function>`
	結果：
	rval= 0

字串左邊	此函數從字串 1 抽取第一個 nCount 字元，並複製到傳回變數。
	
	參數：
	str1 - 字串
	nCount - 字元數
	
	語法：
	`<function name="string.left" return="<result string>"> str1, nCount </function>`
	
	範例：
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let>`
	
	`<function name="string.left" return="str2"> str1, 12 </function>`
	結果：
	str2="A brown bear"

函數名稱	意義

字串右邊	此函數從字串 1 抽取最後一個 nCount 字元，並複製到傳回變數。
	
	參數：
	str1 - 字串
	nCount - 字元數
	
	語法：
	`<function name="string.right" return="<result string>"> str1, nCount </function>`
	
	範例：
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let>`
	
	`<function name="string.right" return="str2"> str1, 10 </function>`
	結果：
	str2="brown dog."

字串中間	此函數從字串 1 抽取指定數目的字元，以 iFirst 指標為起始，並複製到傳回變數。
	
	參數：
	str1 - 字串
	iFirst - 起始索引
	nCount - 字元數
	
	語法：
	`<function name="string.middle"`
	`return="<result string>"> str1, iFirst, nCount </function>`
	
	範例：
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<let name="str2" type="string"></let>`
	
	`<function name="string.middle" return="str2"> str1, 2, 5 </function>`
	結果：
	str2="brown"

函數名稱	意義

字串長度	此函數以字串提供字元數目。
	
	參數：
	str1 - 字串
	
	語法：
	`<function name="string.length" return="<int var>"> str1 </function>`
	
	範例：
	`<let name="length">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog.</let>`
	`<function name="string.length" return="length"> str1 </function>`
	結果：
	length = 31

取代的字串	此函數以新字串更換所有發現的次字串。
	
	參數：
	string - 字串變數
	find string - 欲取代的字串
	new string - 新字串
	
	語法：
	`<function name="string.replace"> string, find string, new string`
	`</function>`
	
	範例：
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	
	`<function name="string.replace" > str1, _T"a brown dog" ,`
	`_T"a big salmon"</function>`
	結果：
	str1 = "A brown bear hunts a big salmon!"

函數名稱	意義

移除的字串	此函數移除所有發現的次字串。
	
	參數：
	string - 字串變數
	remove string - 欲刪除的字串
	
	語法：
	`<function name="string.remove"> string, remove string </function>`
	
	範例：
	`<let name="index">0</let>`
	
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	`<function name="string.remove" > str1, _T"a brown dog" </function>`
	結果：
	str1 = "A brown bear hunts"

插入的字串	此函數在指定的指標插入一個字串。
	
	參數：
	string - 字串變數
	index - 索引（從零開始）
	insert string - 欲插入的字串
	
	語法：
	`<function name="string.insert"> string, index, insert string`
	`</function>`
	
	範例：
	`<let name="str1" type="string">A brown bear hunts. </let>`
	`<let name="str2" type="string">a brown dog</let>`
	
	`<function name="string.insert" > str1, 19, str2`
	`</function>`
	結果：
	str1 = "A brown bear hunts a brown dog"

函數名稱	意義

字串刪除	此函數將刪除特定字元數目，從指定起始位置的字元開始。
	
	參數：
	string - 字串變數
	start index - 起始索引（從零開始）
	nCount - 要移除的字元數目
	
	語法：
	`<function name="string.delete"> string, start index , nCount`
	`</function>`
	
	範例：
	`<let name="str1" type="string">A brown bear hunts. </let>`
	
	`<function name="string.delete" > str1, 2, 5`
	`</function>`
	結果：
	str1 = "A bear hunts"

函數名稱	意義

字串尋找	此函數搜尋傳輸的字串，直到第一個與副字串匹配的字串。
	若找到副字串，本函數提供指標到第一個字元（起始為零），若失敗則為-1。
	
	參數：
	string - 字串變數
	findstring - 要尋找的字串
	startindex – 開始索引（選擇性）
	
	語法：
	`<function name="string.find" return="<int val>"> str1, find string </function>`
	
	範例：
	`<let name="index">0</let>`
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	`<function name="string.find" return="index"> str1, _T"brown" </function>`
	
	結果：
	索引 = 2
	
	或
	`<function name="string.find" return="index"> str1, _T"brown", 1 </function>`

函數名稱	意義

字串反向尋找	此函數搜尋傳輸的字串，直到最後一個與副字串匹配的字串。
	若找到副字串，本函數提供指標到第一個字元（起始為零），若失敗則為-1。
	
	參數：
	string - 字串變數
	find string - 要尋找的字串
	startindex – 起始索引（選擇性）
	
	語法：
	`<function name="string.reversefind"`
	`return="<int val>"> str1, find string </function>`
	
	範例：
	`<let name="index">0</let>`
	`<let name="str1" type="string">A brown bear hunts a brown dog. </let>`
	`<function name="string.reversefind" return="index"> str1, _T"brown"`
	`</function>`
	
	結果：
	索引 = 21
	
	或
	`<function name="string.reversefind"`
	`return="index"> str1, _T"brown" 10 </function>`
	
	結果：
	索引 = 2

函數名稱	意義

String GetAt	此函數從指定位置讀取字元。
	參數：
	str - 字串
	index - 欲讀取的字元以零為基準索引
	
	傳回值：
	必須指定要在其中儲存字元的字串變數名稱。
	
	語法：
	`<function name="string.getat" return="<result string>"><string>, <index></function>`
	
	範例：
	`<let name="resStr" type="string" />`
	`<function name="string.getat" return="resStr">_T"brown", 2</function>`

函數名稱	意義

字串像素高度	計算字串高度(單位：像素)。使用指定的控制或格式的當前字型來執行計算。控制的名稱必須作為字串傳輸。如果指定了空字串，則計算將引用該格式。
	
	參數：
	control name
	字元字串
	字串可以直接指定為文字，也可以指定為字串變數。
	
	回傳：
	應使用整數變數的名稱，並在其中寫入高度。
	
	語法：
	`<function name="STRING.PIXELHEIGHT"`
	`return="<return variable>"><control name>,`
	`<varaible or text></function>`
	
	範例：
	計算相對於格式字型的寬度
	`<let name="pixelsh" />`
	`...`
	`<function name="STRING.PIXELHEIGHT"`
	`return="pixelsh">_T"", cedit1</function>`
	
	計算相對於控制字型的寬度
	`<function name="STRING. PIXELHEIGHT"`
	`return="pixelsh"> cedit1, cedit1</function>`

函數名稱	意義

字串像素寬度	計算字串寬度(單位：像素)。使用指定的控制或格式的當前字型來執行計算。控制的名稱必須作為字串傳輸。如果指定了空字串，則計算將引用該格式。
	
	參數：
	control name
	字元字串
	字串可以直接指定為文字，也可以指定為字串變數。
	
	回傳：
	應使用整數變數的名稱，並在其中寫入文字寬度。
	
	語法：
	`<function name="STRING.PIXELWIDTH"`
	`return="<return variable>"><control name>,`
	`<varaible or text></function>`
	
	範例：
	計算相對於格式字型的寬度
	`<let name="pixels" />`
	`...`
	`<function name="STRING. PIXELWIDTH"`
	`return="pixels">_T"", cedit1</function>`
	
	計算相對於控制字型的寬度
	`<function name="STRING.PIXELWIDTH"`
	`return="pixels"> cedit1, cedit1</function>`

函數名稱	意義

String SetAt	此函數將字元寫入指定位置。索引必須小於最大文字長度。
	
	參數：
	str - 字串
	char - 要寫入指定位置的字元
	index - 目標變數的字元字串以零為基準索引
	
	語法：
	`<function name="string.setat" ><destination string>,`
	`<character string>, <index></function>`
	
	範例：
	`<let name="str" type="string" >br_wn</string>`
	`<function name="string.setat">str, ">_T"o", 2`
	`</function>`

函數名稱	意義

String Split	此函數將字串解構為多個子字串，並將它們複製到指定的字串陣列中。在指定的分隔符號處實行分離。分隔符號並未儲存在子字串中。
	如果找到的分隔符號數大於定義的陣列大小，則此函數會自動擴展陣列。
	
	參數：
	str - 字串
	char - 包含分隔符號的變數名稱
	number - 變數上的名稱，包括執行函數後所生成的子字串編號。
	
	傳回值：
	必須指定字串陣列的名稱，其中包括執行該函數後的子字串。
	
	語法：
	`<function name=" string.split"`
	`return="<result string array>"><string>, <char>,`
	`<number></function>`
	
	範例：
	`<let name="strlist" type="string" dim="2"/>`
	`<let name="str_num" />`
	`<function name="string.split" return="strlist">`
	`_T"brown;green;blue;red", _T";", str_num </function>`

字串修剪左邊	此函數將字串的起始字元刪除。
	
	參數：
	str1 - 字串變數
	
	語法：
	`<function name="string.trimleft" > str1 </function>`
	
	範例：
	`<let name="str1" type="string">test trim left</let>`
	`<function name="string.trimleft" > str1 </function>`
	結果：
	str1 = "test trim left"

函數名稱	意義

字串修剪右邊	此函數將字串的結束字元刪除。
	
	參數：
	str1 - 字串變數
	
	語法：
	`<function name="string.trimright" > str1 </function>`
	
	範例：
	`<let name="str1" type="string"> test trim right  </let>`
	`<function name="string.trimright" > str1`
	`</function>`
	結果：
	str1 = "test trim right"

正弦	此函數計算傳輸值的正弦值，以度為單位。
	
	參數：
	double - 角度
	
	語法：
	`<function name="sin" return="<double val>"> double </function>`
	
	範例：
	`<let name= "sin_val" type="double"></let>`
	`<function name="sin" return="sin_val"> 20.0`
	`</function>`

函數名稱	意義

餘弦	此函數計算傳輸值的餘弦值，以度為單位。
	
	參數：
	double - 角度
	
	語法：
	`<function name="cos" return="<double val>"> double </function>`
	
	範例：
	`<let name= "cos_val" type="double"></let>`
	`<function name="cos" return="cos_val"> 20.0`
	`</function>`

正切	此函數計算傳輸值的正切值，以度為單位。
	
	參數：
	double - 角度
	
	語法：
	`<function name="tan" return="<double val>"> double </function>`
	
	範例：
	`<let name= "tan_val" type="double"></let>`
	`<function name="tan" return="tan_val"> 20.0`
	`</function>`

ARCSIN	此函數計算傳輸值的反正弦值，以度為單位。
	
	參數：
	double - 在 -PI/2 到 +PI/2 範圍之間的 x
	
	語法：
	`<function name="arcsin" return="<double val>"> double </function>`
	
	範例：
	`<let name= "arcsin_val" type="double"></let>`
	`<function name="arcsin" return="arcsin_val"> 20.0`
	`</function>`

函數名稱	意義

ARCOS	此函數計算傳輸值的反餘弦值，以度為單位。
	
	參數：
	double - 在 -PI/2 到 +PI/2 範圍之間的 x
	
	語法：
	`<function name="arcos" return="<double val>"> double </function>`
	
	範例：
	`<let name= "arccos_val" type="double"></let>`
	`<function name="arccos" return="arccos_val"> 20.0`
	`</function>`

ARCTAN	此函數計算傳輸值的反正切值，以度為單位。
	
	參數：
	double - y/x 的反正切
	
	語法：
	`<function name="arctan" return="<double val>"> double </function>`
	
	範例：
	`<let name= "arctan_val" type="double"></let>`
	`<function name="arctan" return="arctan_val"> 20.0`
	`</function>`

檔案處理	

函數名稱	意義

讀取檔案	此函數將指定檔案中的內容讀取至字串變數。
	可選擇性指定要讀取的字元數做為第二參數。
	
	屬性：
	name - 檔案名稱必須使用小寫字母書寫。
	透過使用 appl 或 dvm 目錄作為起點的相對路徑，存取其他目錄中的檔案。
	return - 區域變數的名稱
	
	參數：
	progname - 檔案名稱
	number of characters - 要讀取的字元，以位元組為單位（選擇性）
	
	語法：
	`<function name="doc.readfromfile"`
	`return="<string var>"> progname, number of characters`
	`</function>`
	
	範例：
	`<let name = "my_var" type="string" ></let>`
	
	NC 檔案系統
	`<function name="doc.readfromfile"`
	`return="my_var"> _T"n:\mpf\test.mpf"`
	`</function>`
	
	CF 卡
	`<function name="doc.readfromfile"`
	`return="my_var"> _T"f:\appl\test.mpf"`
	`</function>`
	
	或
	
	`<function name="doc.readfromfile"`
	`return="my_var"> _T".\test.mpf"`
	`</function>`

函數名稱	意義

寫入檔案	此函數將字串變數的內容寫入指定的檔案。
	檔案名稱必須使用小寫字母書寫。
	透過使用 appl 或 dvm 目錄作為起點的相對路徑，存取其他目錄中的檔案。
	
	參數：
	progname - 檔案名稱
	str1 - 字串
	
	語法：
	`<function name="doc.writetofile" > progname, str1 </function>`
	
	範例：
	`<let name = "my_var" type="string" > file content </let>`
	
	NC 檔案系統
	`<function name="doc.writetofile">_T"n:\mpf\test.mpf", my_var </function>`
	
	CF 卡
	`<function name="doc.writetofile">_T"f:\appl\test.mpf", my_var </function>`
	
	或
	
	`<function name="doc.writetofile">_T".\test.mpf", my_var </function>`

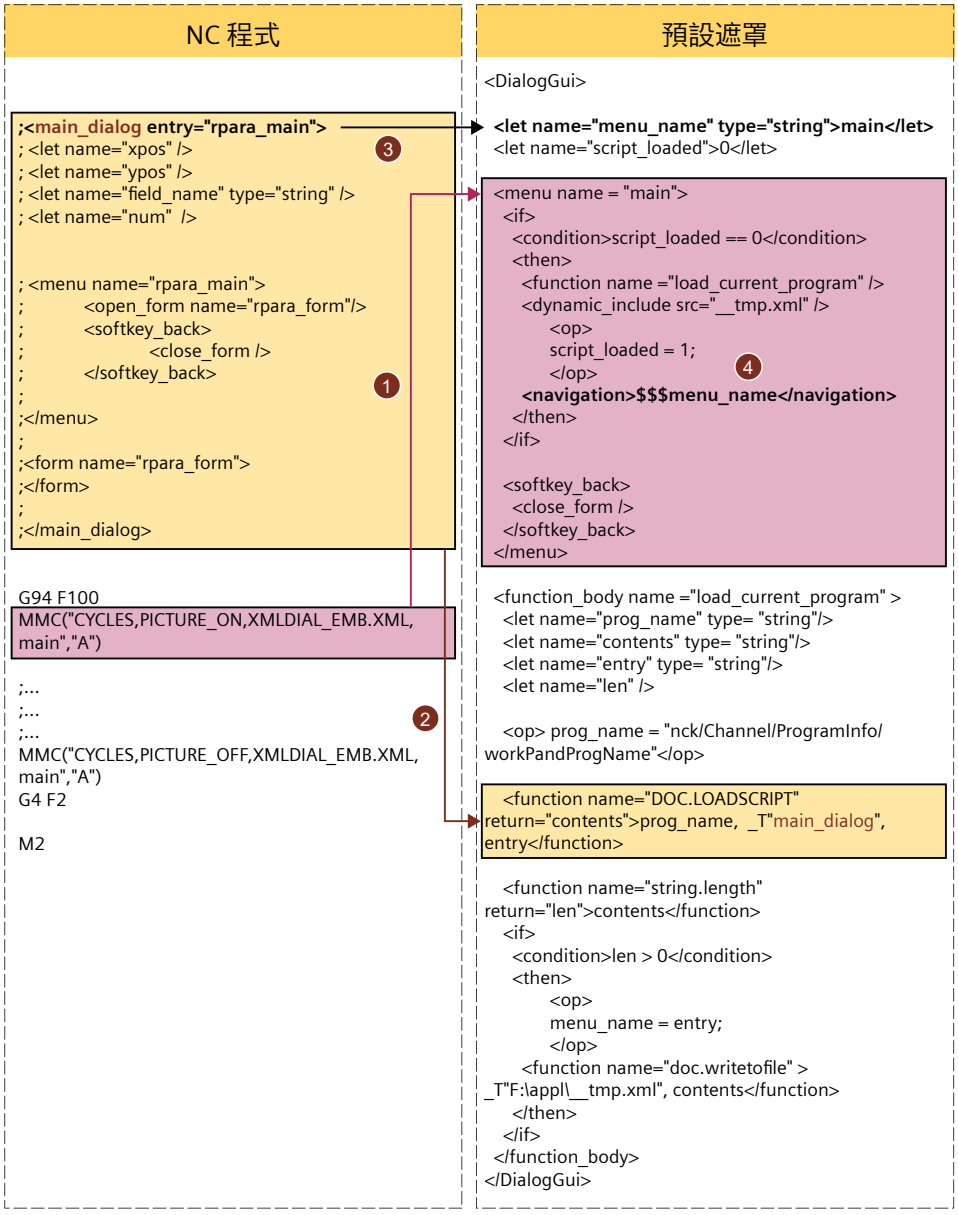
函數名稱	意義

刪除檔案	此函數從目錄中移除指定的檔案。
	檔案名稱必須使用小寫字母書寫。
	透過使用 appl 或 dvm 目錄作為起點的相對路徑，存取其他目錄中的檔案。
	
	參數：
	progname - 檔案名稱
	
	語法：
	`<function name="doc.remove" > progname </function>`
	
	範例：
	NC 檔案系統
	`<function name="doc.remove">_T"n:\mpf\test.mpf"`
	`</function>`
	
	CF 卡
	`<function name="doc.remove">_T"f:\appl\test.mpf"`
	`</function>`
	
	或
	
	`<function name="doc.remove">_T".\test.mpf"`
	`</function>`

函數名稱	意義

取出部分指令碼	此函數會將內嵌於工件程式的對話方塊說明複製至指定的區域變數。
	
	要指定的呼叫參數為程式名稱、對話方塊名稱，以及用於儲存主要功能表名稱的變數。如果在工件程式中找不到對話方塊描述的名稱，則傳回的變數將包含此描述。如果變數的內容儲存於檔案，則可以間接呼叫執行此指令碼。
	
	系統提供一個指令碼，可從作用中的工件程式取出對話方塊描述並啟用對話方塊。此指令碼可在 MMC 命令中呼叫以啟用與此工件程式有關的螢幕。
	
	語法：
	`<function name="doc.loadscript"`
	`return="<name of script variable>">progname, _T"dialog part name",`
	`main menu </function>`
	
	屬性：
	return - 儲存取出指令碼的變數
	
	參數：
	progname - 程式的完整路徑。（路徑名稱可使用 DOS 標記傳遞至函數。）
	main menu - 找到的功能表名稱會複製至此變數
	dialog part name - 嵌入對話方塊描述的標籤名稱
	
	範例：
	`<function name="doc.loadscript" return="contents">prog_name,`
	`_T"main_dialog", entry</function>`

函數名稱	意義
取出部分指令碼 續	圖示範例：



函數名稱	意義

存在	若該檔案存在，則傳回值 1。
	檔案名稱必須使用小寫字母書寫。
	透過使用 appl 或 dvm 目錄作為起點的相對路徑，存取其他目錄中的檔案。
	
	參數：
	progname - 檔案名稱
	
	語法：
	`<function name="doc.exist" return="<int_var>" > progname </function>`
	
	範例：
	`<let name ="exist">0</let>`
	
	NC 檔案系統
	`<function name="doc.exist" return="exist">_T"n:\mpf\test.mpf" </function>`
	
	CF 卡
	`<function name="doc.exist" return="exist">_T"f:\appl\test.mpf" </function>`
	
	或
	
	`<function name="doc.exist" return="exist">_T".\test.mpf" </function>`

NC 程式選擇	此函數選擇指定的程式以供執行。此程式必須儲存在 NC 檔案系統。
	
	參數：
	progname - 檔案名稱
	
	語法：
	`<function name="ncfunc.select"> progname </function>`
	
	範例：
	NC 檔案系統
	`<function name="ncfunc.select"> _T"n:\mpf\test.mpf" </function>`

函數名稱	意義

設定個別位元	此函數係用來操縱指定變數的個別位元。
	位元可加以設定或重置。
	
	語法：
	`<function name="ncfunc.bitset" refvar="address" value="set/reset" > bit0, bit1, … bit9 </function>`
	
	屬性：
	refvar - 指定變數名稱，其中位元組合必須寫入
	value – 位元值，數值範圍 0 與 1
	
	值：
	以零開頭的位元編號應以函數值傳輸。
	每個呼叫最多可以修改 10 位元。
	
	範例：
	`<function name="ncfunc.bitset" refvar="nck/Channel/Parameter/R[1]" value="1" > 0, 2, 3, 7 </function>`
	
	`<function name="ncfunc.bitset" refvar="nck/Channel/Parameter/R[1]" value="0" > 1, 4 </function>`

刪除控制器	此函數刪除指定的圖片控制器。
	
	語法：
	`<function name="control.delete"> control name </function>`
	
	屬性：
	name - 函數名稱
	
	值：
	control name – 控制項名稱
	
	範例：
	`<function name="control.delete"> _T"my_editfield" </function>`

函數名稱	意義

新增項目	此函數會在清單結尾插入新的元件。
	附註：
	此函數僅適用於控制器類型 "listbox" 及 "graphicbox"。
	
	語法：
	`<function name="control.additem"> control name, item </function>`
	
	屬性：
	name - 函數名稱
	
	值：
	control name – 控制項名稱
	item - 要插入的表示式
	itemdata - 整數值；由使用者定義
	
	範例：
	`<let name ="itemdata">1</let>`
	…
	…
	…
	`<op> item_string = _T"text1" </op>`
	`<function name="control.additem">_T"listbox1", item_string, itemdata </function>`

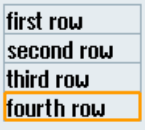
函數名稱	意義

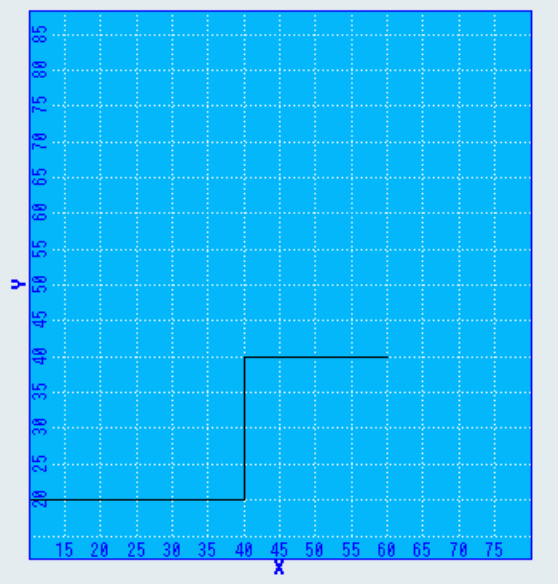
插入項目	此函數會在指定位置插入新的元件。
	附註：
	此函數僅適用於控制器類型「listbox」及「 」。
	
	語法：
	`<function name="control.insertitem"> control name, index, item, itemdata </function>`
	
	屬性：
	name - 函數名稱
	
	值：
	control name - 控制項名稱
	index - 以零開始的位置
	item - 要插入的表示式
	itemdata - 整數值；由使用者定義
	
	範例：
	`<let name ="itemdata">1</let>`
	…
	…
	…
	`<op> item_string = _T"text2" </op>`
	`<function name="control.insertitem">_T"listbox1", 1, item_string, itemdata </function>`

函數名稱	意義

刪除項目	此函數會將指定位置的元件刪除。
	附註：
	此函數僅適用於控制器類型「listbox」及「 」。
	
	語法：
	`<function name="control.deleteitem"> control name, index </function>`
	
	屬性：
	name - 函數名稱
	
	值：
	control name - 控制項名稱
	index– 從 0 開始的索引
	
	範例：
	`<function name="control.deleteitem">_T"listbox1", 1</function>`

函數名稱	意義

載入項目	此函數將表示式清單插入控制器。
	此函數僅適用於控制器類型 "listbox" 及 "graphicbox"。
	
	語法：
	`<function name="control.loaditem"> control name, list </function>`
	
	屬性：
	name - 函數名稱
	
	值：
	control name – 控制項名稱
	list- 字串變數
	變數中包含的字串必須符合以下所述之結構。
	
	清單的架構：
	清單包含一些表示式，彼此之間必須用 \\n 分隔。
	
	範例：
	在範例中，內容透過功能 control.loaditem 指派給列表方塊。
	以控制字元 \\n 分隔該行的表示式。
	`<CONTROL name = "list" xpos = "160" ypos = "32"`
	`width ="100" height="200" fieldtype="listbox"/>`
	`<let name="lb_list" type="string" />`
	`<op>`
	`lb_list = _T"first row\\n";`
	`lb_list = lb_list + _T"second row\\n";`
	`lb_list = lb_list + _T"third row\\n";`
	`lb_list = lb_list +_T"fourth row\\n";`
	`</op>`
	`<function name="control.loaditem">_T"list",`
	`lb_list</function>`
	 

函數名稱	意義
載入項目 續	**範例：** 在範例中，內容透過功能 control.loaditem 指派給圖形方塊。 以控制字元 \\n 分隔圖形元素。 <control name= "c_gbox" xpos = "6" ypos="32" width="328" height="356" fieldtype="graphicbox" /> <let name="plot_list" type="string" /> <op> plot_list = _T"l;10;20;40;20\\n"; plot_list = plot_list + _T"l;40;20;40;40\\n"; plot_list = plot_list + _T"l;40;40;60;40\\n"; plot_list = plot_list +_T"l;80;0;80;100\\n"; </op> <function name="control.loaditem">_T"c_gbox", plot_list</function> 

函數名稱	意義

空白	此函數刪除指定列表方塊或圖形方塊控制器的內容。
	
	語法：
	`<function name="control.empty"> control name,`
	`</function>`
	
	屬性：
	name - 函數名稱
	
	值：
	control name – 控制項名稱
	
	範例：
	`<function name="control.empty">_T"listbox1"`
	`</function>`

取得焦點	此函數提供控制項名稱，具有輸入焦點。
	
	語法：
	`<function name="control.getfocus" return="focus_name" />`
	
	屬性：
	name - 函數名稱
	return – 必須指定為字串變數，以便將控制項名稱複製進去。
	
	範例：
	`<let name>="focus_field" type="string"></let>`
	`<function name="control.getfocus" return="focus_field"/>`

函數名稱	意義

設定焦點	此函數將輸入焦點設定至指定的控制器。
	控制項名稱必須以函數的文字表示式傳輸。
	
	語法：
	`<function name="control.setfocus"> control name`
	`</function>`
	
	屬性：
	name - 函數名稱
	
	值：
	control name - 控制的名稱
	
	範例：
	`<function name="control.setfocus" ">_T"listbox1"`
	`</function>`

取得游標選擇	對列表方塊而言，此函數提供游標索引。
	控制項名稱必須以函數的文字表示式傳輸。
	
	語法：
	`<function name="control.getcurssel" return="var">control name`
	`</function>`
	
	範例：
	`<let name>="index"></let>`
	`<function name="control.getcurssel" return="index">_T"listbox1"`
	`</function>`

函數名稱	意義

設定游標選擇	對列表方塊而言，此函數將游標設定至適當的行。
	控制項名稱必須以函數的文字表示式傳輸。
	
	語法：
	`<function name="control.setcurssel" > control name, index`
	`</function>`
	
	範例：
	`<let name>="index">2</let>`
	`<function name="control.setcurssel" ">_T"listbox1",index`
	`</function>`

取得項目	對列表方塊而言，此函數將選取行的內容複製到指定變數。
	字串變數必須指定為參考變數。
	控制項名稱必須以函數的文字表示式傳輸。
	
	語法：
	`<function name="control.getitem" return="var"> control name, index`
	`</function>`
	
	範例：
	`<let name>="index">2</let>`
	`<let name>="item" type="string"></let>`
	
	`<function name="control.getitem" return="item" ">_T"listbox1",index`
	`</function>`

函數名稱	意義

取得項目資料	對列表方塊而言，此函數將使用者自訂的元件分配值複製到指定變數。
	對於編輯控制器而言，此函數將使用者自訂的分配值（item_data）複製到指定變數。
	整數變數必須指定為參考變數。
	控制項名稱必須以函數的文字表示式傳輸。
	
	語法：
	`<function name="control.getitemdata" return="var"> control name, index </function>`
	
	範例：
	`<let name>="index">2</let>`
	`<let name>="itemdata"></let>`
	
	`<function name="control.getitemdata" return="itemdata" ">_T"listbox1",index</function>`

函數名稱	意義
影像方塊設定	此函數用於控制**影像方塊**及**圖形方塊**等控制項的可見區。要指定的呼叫參數為控制項名稱、控制項命令，以及相關的值。 **語法：** `<function name="control.imageboxset">control name, command, command parameter</function>` **呼叫參數：** • **x_offs** 此函數將影像部分移動至指定的 X 位置。另外要指定的參數為左側角落的座標。 • **y_offs** 此函數將影像部分移動至指定的 Y 位置。另外要指定的參數為上方角落的座標。 • **xy_offs** 此函數將影像部分移動至指定的 X/Y 位置。另外要指定的參數為左側與上方角落的座標。 • **followCursor** 影像部分依照設定的游標位置。 • **zoomplus** 影像以 0.12 係數放大。 • **zoomminus** 影像以 0.12 係數縮小。 • **autozoom** 影像將自動調整尺寸以符合顯示區域的大小。 • **Update** 控制項將重新繪製。 • **SetBkColor** 此命令設定指定的背景色。另外要指定的參數為色彩編碼。 • **SetCursorRect** 以矩形指定的部分將移動至可見區域。 • **SetAnimationState** 僅套用至影像方塊控制） – **start** - 此命令會開始動畫。 – **stop** - 此命令會停止動畫。

函數名稱	意義
影像方塊集 續	**範例：** ... ... <softkey POSITION="1"> <caption>zoom%nin</caption> <function name="control.imageboxset">_T"c_gbox", _T"zoomplus"</function> </softkey> ... ... <form name = "main_form"> <init> <caption> graphicbox</caption> <control name= "c_gbox" xpos = "6" ypos="32" width="328" height="356" fieldtype="graphicbox" /> <let name="plot_list" type="string" /> <op> plot_list = _T"l;10;20;40;20\\n"; plot_list = plot_list + _T"l;40;20;40;40\\n"; plot_list = plot_list + _T"l;40;40;60;40\\n"; plot_list = plot_list +_T"l;80;0;80;100\\n"; </op> <function name="control.loaditem">_T"c_gbox", plot_list</function> </init> </form>
影像方塊取得	此函數用於查詢控制項屬性影像方塊。要指定的呼叫參數為控制項命令，以及相關的值。 **語法：** <function name="control.imageboxget">control name, command, command parameter</function> **呼叫參數：** **GetViewSize** 傳回顯示區域的大小 另外要指定的參數為結構類型 SIZE 的變數。 **範例：** <let name="view_size" type="size" /> <function name="control.imageboxget">_T"topoview", _T"**GetViewSize**", view_size</function>

函數名稱	意義

點陣圖灰暗	此函數將點陣圖尺寸複製為具有結構類型 SIZE 的變數。為定義此類型，必須在專案中包含 struct_def.xml 檔案。
	更多資訊請參閱章節「建立檔案 struct_def.xml (頁 155)」
	
	語法：
	`<function name="bitmap.dim" >name, variable type`
	`</function>`
	
	參數：
	name - 檔案路徑
	variable type - 變數類型 SIZE 的變數名稱
	
	範例：
	`<let name="bmp_size" type="size" />`
	
	`<function name="bitmap.dim" >_T"test.bmp", bmp_size`
	`</function>`

螢幕解析度	此函數將系統的絕對螢幕解析度複製為具有結構類型 SIZE 的變數。為定義此類型，必須在專案中包含 struct_def.xml 檔案。 **更多資訊**請參閱章節「建立檔案 struct_def.xml (頁 155)」 **語法：** `<function name="hmi.screen_resolution" >varialble type </function>`
取得 HMI 解析度	此函數將 SINUMERIK Operate 的螢幕解析度複製為具有結構類型 SIZE 的變數。為定義此類型，必須在專案中包含 struct_def.xml 檔案。 **更多資訊**請參閱章節「建立檔案 struct_def.xml (頁 155)」 **語法：** `<function name="hmi.get_hmi_resolution" >variable type </function>`

函數名稱	意義
取得標題高度	此函數傳回標題列高度，以畫素為單位。 **語法：** `<function name="hmi.get_caption_height" return="<return var>" />` **屬性：** **return** - 整數變數

取得字型家族	此函數提供已安裝字型的清單。列出的字型名稱用 LF 字元分隔。字串變數的名稱將作為傳回值傳遞。
	
	語法：
	`<function name="HMI.GET_FONT_FAMILIES"`
	`return="< String-Variable>" />`
	
	範例：
	`<init>`
	`...`
	` <let name="families" type="string" />`
	` <function name="HMI.GET_FONT_FAMILIES" return="families" />`
	` <control name="lb" xpos="8" ypos="40" width="260" height="140"`
	`fieldtype="listbox">`
	` </control>`
	`<function name="control.loaditem">_T"lb", families</function>`
	`...`
	`<update_controls type="true" />`
	`</init>`

Abs	此函數返回指定編號的絕對值。
	
	語法：
	`<function name="abs" return="var"> value`
	`</function>`

SDEG	此函數將指定值轉換成度數。
	
	語法：
	`<function name="sdeg" return="var"> value`
	`</function>`

函數名稱	意義

SRAD	此函數將指定值轉換成弧度。
	
	語法：
	`<function name="srad" return="var"> value`
	`</function>`

sqrt	此函數計算指定值的平方根。
	
	語法：
	`<function name="sqrt" return="var"> value`
	`</function>`

ROUND	此函數將傳輸的編號四捨五入至指定小數點位數的編號。若小數點位數的編號未指定，則此函數會考慮以小數點第一位四捨五入編號。
	
	語法：
	`<function name="round" return="var"> value, nDecimalPlaces`
	`</function>`

FLOOR	此函數提供最大可能之整數值，小於或等於傳輸值。
	
	語法：
	`<function name="floor" return="var"> value`
	`</function>`

CEIL	此函數提供最小可能之整數值，大於或等於傳輸值。
	
	語法：
	`<function name="ceil" return="var"> value`
	`</function>`

日誌	此函數計算指定值的對數。
	
	語法：
	`<function name="log" return="var"> value`
	`</function>`

LOG10	此函數計算指定值的常用（以十為底）對數。
	
	語法：
	`<function name="log10" return="var"> value`
	`</function>`

函數名稱	意義
POW	此函數計算值 "a^b"。 **語法：** `<function name="pow" return="var"> a, b </function>`
MIN	此函數比較傳輸值，並返回較低的數值。 **語法：** `<function name="min" return="var"> value1, value2 </function>`
MAX	此函數比較傳輸值，並返回較高的數值。 **語法：** `<function name="max" return="var"> value1, value2 </function>`
RANDOM	此函數返回偽隨亂數。 **語法：** `<function name="random" return="var" </function>`

函數名稱	意義
MMC	**函數：** 您可使用 MMC 命令，從 SINUMERIK Operate 中的工件程式顯示使用者定義的對話方塊格式。對話方塊格式的外觀是以純文字組態（製造商目錄中的 XML 檔案）定義。系統軟體仍然保持不變。 由於基本操作系統的變更，參數如下： XML → CYCLES 或 POPUPDLG XML_ON → PICTURE_ON XML_OFF → PICTURE_OFF **語法：** MMC（「操作區，命令，XML 指令碼檔案，功能表名稱，保留，保留，顯示時間或確認變數，保留」，「確認模式」） **說明：** • MMC 由 SINUMERIK Operate 中的工件程式以互動方式呼叫對話方塊格式。 • CYCLES 或 POPUPDLG 識別從工件程式啟動的使用者對話方塊。 • PICTURE_ON 或 PICTURE_OFF 指令：對話方塊的選擇或取消選擇 • XML file XML 指令集檔案的名稱 • Menu 功能表標籤名稱，管理欲顯示的對話方塊 • reserved 保留給 SINUMERIK Operate • reserved 保留給 SINUMERIK Operate • Time 確認模式「N」中對話方塊的顯示時間 • reserved 保留給 SINUMERIK Operate • Acknowledgement mode 「S」同步，經由「確定」軟鍵做確認 「N」非同步，在設定時間後對話方塊關閉

函數名稱	意義

MMC 續	**同步呼叫範例：**
	由於基本操作系統的變更，參數如下：
	XML → CYCLES 或 POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	NC 指令
	`MMC("POPUPDLG or CYCLES",PICTURE_ON,mmc_cmd.xml,cmd1,,,,,","S")`
	檔案： `mmc_cmd.xml`
	
	`<menu name = "cmd1">`
	`<open_form name = "cmd1_form" />`
	`<softkey_ok>`
	`<close_form />`
	`</softkey_ok>`
	`</menu>`
	
	`<form name = "cmd1_form" xpos ="12" ypos="100" width="500" height="240">`
	`<init>`
	`</init>`
	`<paint>`
	`</paint>`
	`</form>`
	
	非同步呼叫範例（未預期確認）：
	由於基本操作系統的變更，參數如下：
	XML → CYCLES 或 POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	NC 指令
	`MMC("POPUPDLG or CYCLES",PICTURE_ON,mmc_cmd.xml,cmd1,,,10,,","N")`
	
	檔案： `mmc_cmd.xml <menu name = "cmd1">`
	
	`<open_form name = "cmd1_form" />  <softkey_ok>  <close_form />`
	`</softkey_ok>`
	`</menu>`
	
	`<form name = "cmd1_form" xpos ="12" ypos="100" width="500" height="240">`
	`<init>`
	`</init>`
	`<paint>`

函數名稱	意義
	</paint> </form>

函數名稱	意義

MMC 續	**從工件程式取出部分指令碼的範例：**
	由於基本操作系統的變更，參數如下：
	XML → CYCLES 或 POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	NC 指令
	`MMC("POPUPDLG, PICTURE_ON, XMLDIAL_EMB.XML,main","S")`
	檔案：xmldial_emb.xml
	
	`<DialogGui>`
	`<let name="menu_name" type="string">main</let>`
	`<let name="script_loaded">0</let>`
	`<menu name = "main">`
	`<if>`
	`<condition>script_loaded == 0</condition>`
	`<then>`
	`<function name ="load_current_program" />`
	`<dynamic_include src="__tmp.xml" />`
	`<op>`
	`script_loaded = 1;`
	`</op>`
	`<navigation>$$$menu_name</navigation>`
	`</then>`
	`</if>`
	`<softkey_back>`
	`<close_form />`
	`</softkey_back>`
	`</menu>`
	`<function_body name ="load_current_program" >`
	`<let name="prog_name" type= "string"/>`
	`<let name="contents" type= "string"/>`
	`<let name="entry" type= "string"/>`
	`<let name="len" />`
	`<op> prog_name = "nck/Channel/ProgramInfo/workPandProgName"</op>`
	`<function name="DOC.LOADSCRIPT"`
	`return="contents">prog_name, _T"main_dialog", entry`
	`</function>`
	`<function name="string.length" return="len">contents</function>`
	`<if>`
	`<condition>len > 0</condition>`
	`<then>`
	`<op>`
	`menu_name = entry;`
	`</op>`
	`<function name="doc.writetofile" >_T"__tmp.xml", contents`
	`</function>`
	`</then>`

函數名稱	意義
	</if> </function_body> </DialogGui>

函數名稱	意義
MMC 續	程式設計範例 ;<main_dialog entry="rpara_main"> ; <let name="xpos" /> ; <let name="ypos" /> ; <let name="field_name" type="string" /> ; <let name="num" /> ; <menu name="rpara_main"> ; <open_form name="rpara_form"/> ; <softkey_back> ; <close_form /> ; </softkey_back> ;</menu> ; ;<form name="rpara_form"> ; <init> ; <caption>test mask</caption> ; <let name="count" >0</let> ; <op> ; xpos = 120; ; ypos = 34; ; ; "nck/Channel/Parameter/R[10]" = 10; ; </op> ; <!-- load the number of controls --> ; <op> ; num = "nck/Channel/Parameter/R[10]"; ; </op> ; ; <while> ; <condition>count < num</condition> ; <print name="field_name" text="edit%d">count</print> ; ; <control name = "$field_name" xpos = "$xpos" ypos = "$ypos" refvar="nck/Channel/Parameter/R[$count]" hotlink="true" /> ; <op> ; ypos = ypos +24; ; count = count +1; ; </op> ; </while> ; ; </init> ; ; <paint> ; <op> ; xpos = 8; ; ypos = 36; ; count = 0; ; </op> ; <while>

函數名稱	意義
	; <condition>count < num</condition> ; <print name="field_name" text="R-Parameter%d">count </print> ; ; <text xpos = "$xpos" ypos = "$ypos" >$$$field_name</text> ; <op> ; ypos = ypos +24; ; count = count +1; ; </op> ; </while> ; ; </paint> ; ;</form> ; ;</main_dialog> … G94 F100 MMC("POPUPDLG, PICTURE_ON, XMLDIAL_EMB.XML,main","A") … … … MMC("POPUPDLG, PICTURE_OFF, XMLDIAL_EMB.XML,main","A") G4 F2 M2
ISNUMERIC	此函數用於檢查字串的內容是否可以將其評估為編號。忽略空格及定位字元。符號及小數點仍視為有效字元。如果滿足條件，該函數將傳回值 1。 **參數：** **字串** - 字元字串 **語法：** <function name="isnumeric" return="result"><string> </function>
ROOT	該函數從編號中提取第 n 個根。 **參數：** **值** - 編號 **n** - 根的指數 **語法：** <function name="ROOT" return="result"><value>, <n></function>

3.12 多點觸控操作

3.12.1 多點觸控功能

總覽

引進多點觸控顯示器時，需要調整操作以適應顯示器的擴充功能。例如，軟鍵的剛性定位將會消除，並透過按鈕的自由定位來取代或增補。由於按鈕設計選項的多樣性，僅使用一個控制進行編程已不具任何意義，其外觀完全以操作軟體的設計規格為基礎。切換控制項（僅知有兩種狀態）可以用開關取代，例如，透過圖示顯示對應的狀態，並對點選或輕觸手勢做出反應。

顯示圖形可以具有對平移手勢做出反應的能力。為達此目的，於是擴充了 **imagebox** 控制。

說明

繪圖標籤中輸出的圖形，不會對手勢做出反應。

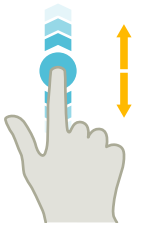
在語法分析器中提供的控制項以外，可以在指令碼中處理手指手勢，從而提供例如在對話方塊中所提供的新導覽策略之選項。手指手勢可用於放大/縮小對話方塊的內容。

簡易 XML 語法分析器支援以下手指手勢：

手指手勢

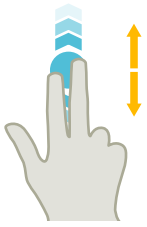
點選

- 選取視窗
- 選取物件（如 NC 集合）
- 啟動輸入欄位
  - 輸入或覆寫數值
  - 再次點選可變更數值



用一根手指垂直輕觸

- 捲動清單（如程式、刀具、零點）
- 捲動檔案（如 NC 程式）



用兩根手指垂直輕觸

- 在清單中捲動頁面（如 ZO）
- 在檔案中捲動頁面（如 NC 程式）

用三根手指垂直輕觸

- 捲動至清單開頭或結尾
- 捲動至檔案開頭或結尾

用一根手指水平輕觸

- 捲動有許多欄的清單

展開

- 放大圖形內容（如模擬、模具製作檢視）

靠攏

- 縮小圖形內容（如模擬、模具製作檢視）

用一根手指平移

- 移動圖形內容（如模擬、模具製作檢視）
- 移動清單內容

使用 **gesture_event** 標籤及 **$gestureinfo** 變數以處理手指手勢。啟用程式中已解碼手指手勢的動作。

3.12.2 程式設計手指手勢

gesture_event 標籤

說明

僅在操作面板支援多點觸控操作時才執行此標籤。

如果操作軟體識別出手指手勢，則語法分析器在 **$gestureinfo** 變數中會提供手勢資訊，並執行 **gesture_event** 標籤。變數擁有 **GestureInfoStruct** 資料類型，包含下列屬性：

屬性	值	意義
type	3	平移手勢
	4	縮小手勢
	5	點擊手勢
flag	1	縮放比例已變更
	2	旋轉角度已變更
state	1	已啟動
	2	update()
	3	已完成
item_data		啟用控制的辨識
	-1	手勢未指派給控制
	!= -1	手勢在控制中執行，建立此控制時指派這個數值
point		手勢的位置
	point.x	手勢的 X 位置
	point.y	手勢的 Y 位置
delta		手勢起點及目前事件之間的差異
	<比例縮放差異>	變更比例縮放係數時
	<角度差異>	變更旋轉角度時

這些屬性會在 **struct_def.xml** 檔案中預先定義。

其他資訊請參閱章節「建立檔案 struct_def.xml (頁 155)」

3.12.3 圖形的手勢控制

Imagebox 控制變數

使用 **Imagebox** 控制變數時，以下三個擴充可用於圖形的手勢控制：

屬性	意義/行為
rotationangle	此屬性值表示圖形的顯示角度。
setrotationmode	屬性值為 **true** 允許進行靠攏手勢
setzoommode	屬性值為 **true** 允許在顯示區域內放大/縮小或移動圖形。此屬性的預設設定為 **false**。

語法

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

點陣圖旋轉的範例

點陣圖順時針旋轉 30.5 度：

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

```
</control>
```

點陣圖縮放的範例

允許點陣圖縮放：

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

imageboxset 控制函數

使用 **control.imageboxset** 函數也可以設定屬性。

其中共有以下命令選項：

命令	意義/行為
SetRotationAngle	圖形繞著 **lparam1** 中指出的角度旋轉。
SetRotationMode	**lparam1** 的數值定義靠攏手勢相對於旋轉的評估。 數值等於 1 - 手勢由控制處理
SetZoomMode	若 **lparam1** 的數值等於 1，則評估放大/縮小的靠攏手勢或移動圖形。

3.12.4 手勢處理

message 標籤

若是縮放比例大於 1.0，顯示區域內的影像將移動。若是係數為 1.0，舉例來說，可以使用此手指手勢瀏覽整個影像清單。在這種情況下，語法分析器傳送訊息至格式，可以在 **message** 標籤中進行評估。

$message_par1 訊息參數包含控制的 Item_Data 值。**$message_par2** 訊息參數傳輸移動的手勢方向訊息。

$message_par2 可以接受下列數值：

- -1 向左滾動
- 1 向右滾動
- -2 向上滾動
- 2 向下滾動

範例

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

…
…
<control name="image_view" xpos="100" ypos="123" width="300"
height="200" fieldtype="imagebox" refvar="image_box_pict_name"
hotlink="true" >
 <property item_data="1000" />
 <property setzoommode="false" />
</control>
…
…
…
<!--
 -1 向左滾動
1 向右滾動
-2 向上滾動
2 向下滾動
```

```
 -->
<message>
 <if condition="$message_par1 == 1000">
  <then>

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

3.13 設定自己的按鈕

按鈕可以與格式整合成為動作按鈕或選擇按鈕。

3.13.1 按鈕

property 標籤

按鈕是控制元件，可以用來當作按鈕或開關（有閂鎖功能的按鈕）。該按鈕可以透過「軟鍵外觀與感覺」樣式中的屬性定義，以及自訂的樣式進行設計。相關屬性必須使用 **property** 標籤定義。

前景或背景顏色可以使用 **color_fg**、**color_bk**、**color_fg_pressed** 及 **color_bk_pressed** 屬性加以變更。

按下/鎖定或**釋放**狀態是以數值一或零表示。必須指出處理函數來評估按鈕的狀態變化。處理函數係使用控制標籤中的 **function** 屬性表示。

開關的狀態可以透過讀出控制變數或是指派參考變數加以判定。

透過觸控操作，只有當手指手勢為「放開」時，手指手勢為「按下」及「釋放」。

語法

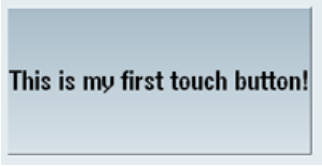
```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" >
 <caption></caption>
</control>
```

或是使用處理函數

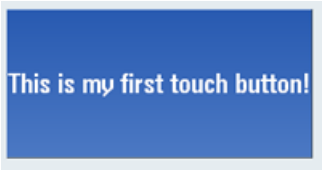
```
<control name="name" xpos="x position" ypos="y position"
fieldtype="pushbutton" function="button_handler" hotlink="true" >
 <caption></caption>
</control>
…
…
…
 <function_body name="button_handler">
…
…
…
 </function_body>
```

或

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

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



已停用



已啟動，已拴鎖

範例

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

3.13.2 按鈕功能

3.13.2.1 按鈕的子標籤

caption 標籤

程式設計師使用 **caption** 標籤指定按鈕文字，以顯示「勿按壓」狀態。預設設定中，文字為置中。文字對齊可以使用 **alignment** 屬性變更。下列屬性值可以指定：

屬性值	對齊方式
靠左	靠左對齊
靠右	靠右對齊
上方	上方
下方	下方
中心	置中

若有不同的文字欲顯示「按壓」狀態，必須使用**按壓**屬性及 **true** 值程式設計第二個標題指令。

語法

置中顯示

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

靠左對齊顯示

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

按壓顯示

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

</control>
```

3.13.2.2 按鈕的屬性

其他的屬性係指派給含有 **property** 標籤的控制。

判定按鈕屬性

checkable 屬性允許按鈕能成為開關使用。若要這樣做，屬性值必須設定為 **true**。

語法

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

停用開關

disabled 屬性控制按鈕是否能夠操作。若值為 **true**，控制動作即不處理。

由於指派參考變數而造成的狀態變更，將導致按鈕的更新。

語法

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

範例

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

指派圖示

預先定義的設計可以藉由指定自有圖示或按鈕顏色來全面涵蓋。圖示可以透過指派給每個狀態的按鈕，如「未按壓」、「按壓」及「停用」。若是狀態「按壓」或「停用」並未指派，控制會顯示「未按壓」狀態的圖示。

其他屬性控制每個圖示的對齊、比例縮放及透明度。

屬性	意義/行為
Picture	前景中的圖示 「未按壓」狀態的圖示名稱
PicturePressed	前景中的圖示 「按壓」狀態的圖示名稱

屬性	意義/行為
PictureDisabled	前景中的圖示 「停用」狀態的圖示名稱
BackgroundPicture	背景中的圖示 「未按壓」狀態的圖示名稱
BackgroundPicturePressed	背景中的圖示 「按壓」狀態的圖示名稱
BackgroundPictureDisabled	背景中的圖示 「停用」狀態的圖示名稱

對齊屬性

在標籤中可以指定更多的屬性，其數值可以參考至列示圖示指派的**對齊**：

屬性值	對齊方式
靠左	靠左對齊
靠右	靠右對齊
上方	上方
下方	下方
中心	置中
延伸	背景圖示： 若使用**延伸**數值，語法分析器會將圖示比例縮放至按鈕的矩形區域。 按鈕所要的任何外框都能透過以 **transparent** 屬性宣告透明色彩來產生按鈕。

3.13.2.3 按鈕的控制變數

隨後可以透過為底下所列的屬性變數指派新值，變更按鈕屬性。透過指定控制名稱及隨後的控制變數名稱，可在操作說明中指派數值。兩個名稱之間必須以句點分隔。

語法

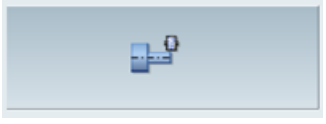
<Control-Name>.<Control-Variable>

控制變數	意義/行為
backgroundpicture	變數類型：String 背景圖示的名稱
backgroundpicturedisabled	變數類型：String 停用狀態的背景圖示名稱
backgroundpicturepressed	變數類型：String 按壓狀態的背景圖示名稱
picture	變數類型：String 前景圖示的名稱
picturedisabled	變數類型：String 停用狀態的前景圖示名稱
picturepressed	變數類型：String 按壓狀態的前景圖示名稱
picture_textalignedtopicture	變數類型：Bool 值： 1 = true 0 = false 按鈕文字與圖示對齊
picturedisabled_textalignedtopicture	變數類型：Bool 值： 1 = true 0 = false 按鈕文字與「停用狀態」圖示對齊

控制變數	意義/行為
picturepressed_textalignedtopicture	變數類型：Bool 值： 1 = true 0 = false 按鈕文字與「按壓狀態」圖示對齊
backgroundpicture_keepaspectratio	變數類型：Bool 值： 1 = true 0 = false 保留或忽略背景圖示的寬高比
backgroundpicturedisabled_keepaspectratio	變數類型：Bool 值： 1 = true 0 = false 保留或忽略「停用狀態」背景圖示的寬高比
backgroundpicturepressed_keepaspectratio	變數類型：Bool 值： 1 = true 0 = false 保留或忽略「按壓狀態」背景圖示的寬高比
picture_keepaspectratio	變數類型：Bool 值： 1 = true 0 = false 保留或忽略圖示的寬高比
picturedisabled_keepaspectratio	變數類型：Bool 值： 1 = true 0 = false 保留或忽略「停用狀態」圖示的寬高比

控制變數	意義/行為
picturepressed_keepaspectratio	變數類型：Bool 值： 1 = true 0 = false 保留或忽略「按壓狀態」圖示的寬高比
backgroundpicture_scaled	變數類型：Bool 值： 1 = true 0 = false 調整背景圖示至按鈕大小
backgroundpicturedisabled_scaled	變數類型：Bool 值： 1 = true 0 = false 調整「停用狀態」背景圖示至按鈕大小
backgroundpicturepressed_scaled	變數類型：Bool 值： 1 = true 0 = false 調整「按壓狀態」背景圖示至按鈕大小
picture_scaled	變數類型：Bool 值： 1 = true 0 = false 調整圖示至按鈕大小
picturedisabled_scaled	變數類型：Bool 值： 1 = true 0 = false 調整「停用狀態」圖示至按鈕大小
picturepressed_scaled	變數類型：Bool 調整「按壓狀態」圖示至按鈕大小

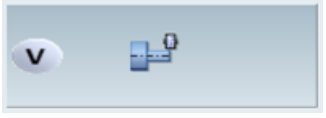
控制變數	意義/行為
backpicture_alignment	變數類型：String 參閱對齊屬性
backgroundpicturedisabled_alignment	
backgroundpicturepressed_alignment	
picture_alignment	
picturedisabled_alignment	
picturepressed_alignment	
caption	變數類型：String 「未按壓狀態」按鈕文字
captionpressed	變數類型：String 「按壓狀態」按鈕文字
caption_alignment	變數類型：String 參閱對齊屬性
captionpressed_alignment	
captiondisabled_alignment	
disabled	變數類型：Bool 值： 1 = true - 按鈕停用 0 = false - 按鈕可以操作



Picture

PicturePressed

PictureDisabled



BackgroundPicture + Picture

TextAlignedToPicture 屬性

若是屬性的值為 **true**，文字將對應於圖示對齊。

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

比例縮放屬性

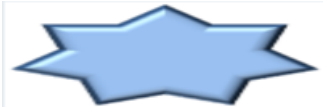
如果屬性的值為 **true**，則將影像大小調整為按鈕大小，使得最多 50% 的按鈕高度或寬度可用於圖形。

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

延伸屬性（只對背景圖示有作用）

如果屬性的值為 **true**，則將影像大小調整為按鈕大小。

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



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

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

3.13.3 開啟/關閉

開關控制為圖形化元件，透過圖示的方式發出兩個狀態的其中一個訊號。此控制可以透過觸控或使用滑鼠操作。

採用的狀態可以儲存在參考變數中，或者狀態的變更可以在指派給控制的函數中判定。函數係使用控制標籤中的 **function** 屬性指派。

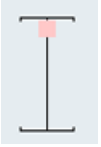
alignment 屬性啟用按鈕的垂直或水平對齊。

此控制並沒有標準設計。若是沒有指派給控制任何圖示，則只會顯示邊界框及開關位置（以點顯示）。

語法

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

對齊屬性



縱向開關：值 **vr**

橫向開關：值 **hr**

3.13.4 開關功能

3.13.4.1 開關的屬性

property 標籤

下列的開關位置可以指派給含有 **property** 標籤的控制：

屬性	意義/行為
left_position	若開關往左移動，將屬性值指派給控制變數。
right_position	若開關往右移動，將屬性值指派給控制變數。
upper_position	若開關往上移動，將屬性值指派給控制變數。
lower_position	若開關往下移動，將屬性值指派給控制變數。
disabled	該屬性控制開關是否能夠操作。 若值為 **true**，控制動作即無法評估。 由於指派參考變數而造成的狀態變更，將導致開關狀態的更新。

最後的開關設計必須藉由指定其他的屬性才能判定。這些屬性必須與屬性 **left_position**、**right_position**、**upper_position** 或 **lower_position** 一起定義。

開關所要的任何外框都能透過以 **transparent** 屬性宣告一個顏色為透明色彩來產生按鈕。

屬性	意義/行為
picture	圖示在前景中的「未按壓」狀態圖示名稱
pictureDisabled	圖示在前景中的「停用」狀態圖示名稱
transparent	屬性值包括透明顏色的色彩值。 所有指派給開關的圖示都使用此色彩值。
caption	屬性值包括與開關位置相關的文字。此文字會顯示在元件上，並且受限於開關的大小。

3.13.4.2 開關的控制變數

隨後可以透過為底下所列的控制變數指派新值，變更開關屬性。透過指定控制名稱及隨後的控制變數名稱，可在操作說明中指派數值。兩個名稱之間必須以句點分隔。

語法

<Control-Name>.<Control-Variable>

控制變數	意義/行為
Left_position	變數類型：Int 若開關往左移動，將屬性值指派給控制變數。
Right_position	變數類型：Int 若開關往右移動，將屬性值指派給控制變數。
Lower_position	變數類型：Int 若開關往下移動，將屬性值指派給控制變數。
Upper_position	變數類型：Int 若開關往上移動，將屬性值指派給控制變數。
Picture_left_position	變數類型：String 左方位置的圖示名稱
Picture_right_position	變數類型：String 右方位置的圖示名稱
Picture_upper_position	變數類型：String 上方位置的圖示名稱
Picture_lower_position	變數類型：String 下方位置的圖示名稱
Picturedisabled_left_position	變數類型：String 左方位置「停用狀態」的圖示名稱
Picturedisabled_right_position	變數類型：String 右方位置「停用狀態」的圖示名稱
Picturedisabled_upper_position	變數類型：String 上方位置「停用狀態」的圖示名稱
Picturedisabled_lower_position	變數類型：String 下方位置「停用狀態」的圖示名稱

控制變數	意義/行為
Caption_left_position	變數類型：String 左方位置的文字
Caption_right_position	變數類型：String 右方位置的文字
Caption_upper_position	變數類型：String 上方位置的文字
Caption_lower_position	變數類型：String 下方位置的文字
disabled	變數類型：Bool 值： 1 = true - 開關停用 0 = false - 開關可以操作

橫向顯示

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

縱向顯示

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

或是指派處理函數

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

範例

icon_left.bmp

icon_right.bmp

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

透過控制變數的方式指派屬性

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

3.13.5 單選按鈕

單選按鈕啟用從數個選項做出選擇。必須為每種選項建立按鈕。屬於一個格式、群組方塊或滾軸區的所有單選按鈕，以這樣的方式彼此互動，亦即只有一個按鈕會被標記為已選擇。按鈕狀態傳輸至控制變數。

語法

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

```
 <caption>button text</caption>
</control>
```

3.13.6 核取方塊

核取方塊啟用從數個選項做出選擇。必須為每種選項建立按鈕。核取方塊的狀態傳輸至控制變數。

語法

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

3.13.7 群組方塊

群組方塊以框架及標題，將控制群組以視覺化涵蓋其中。它將邏輯上相關的控制進行分組，這些控制僅用於在群組內互動。內建控制的座標參考至標題行下方的左上角。

屬於該群組的所有控制都作為子控制內建至控制標籤中。

在指派內建控制名稱及 **Item_Data** 值時，請務必注意，它們對於屬於格式的所有控制皆必須是唯一。

群組方塊的標題係以 **caption** 標籤指定。

語法

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

3.13.8 捲軸區域

使用捲軸區域將控制項顯示在指定區域內。內建控制的座標參考至捲軸區域的左上角。若是控制建立在可見區域之外，將啟用移動可見區域。屬於該區域的所有控制都必須作為子控制內建至控制標籤中。

在指派內建控制名稱及 **Item_Data** 值時，請務必注意，它們對於屬於格式的所有控制皆必須是唯一。

語法

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

觸控操作

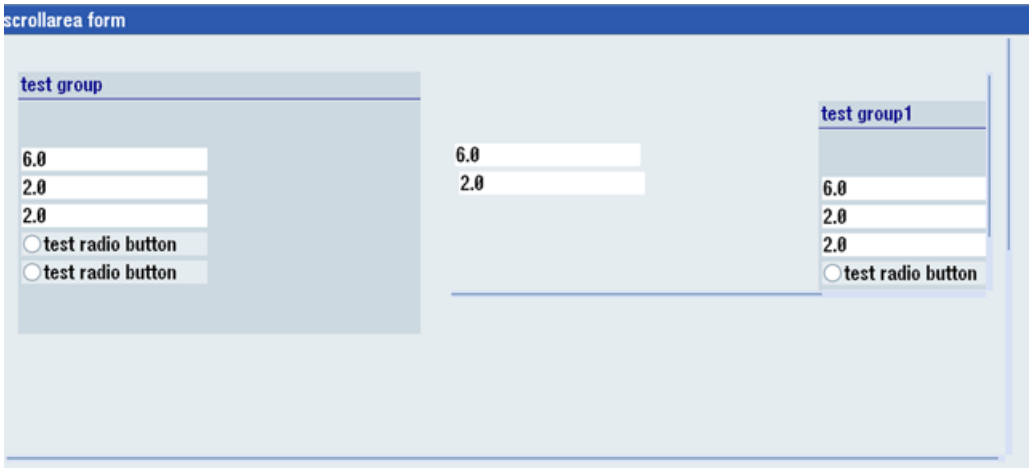
如果焦點放在當前不完全可見的控制上，則移動可見區域，直到控制可以完整顯示為止。

標號手勢

如果無法將手勢指派給內建控制，則將此手勢轉發至指令碼。

範例

巢狀捲軸區域



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

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

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

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

3.14 側邊螢幕應用

3.14.1 簡易 XML 在側邊螢幕中

簡易 XML 指令碼語言也可以用來建立側邊螢幕區域中的對話方塊。側邊螢幕元件**頁面**及**小工具**提供用來顯示對話方塊。其連接係透過 **slsidescreen.ini** 檔案產生。啟動操作軟體時，側邊螢幕元件會自動啟動，並且只有在系統關機時終止。也就是說當第一次啟動側邊螢幕元件時，語法分析器會載入指派的指令碼。

側邊螢幕區域的大小是透過從 **slsidescreen_<螢幕解析度>.ini** 檔案的配置參數來決定。欲啟用獨立的組態設定，簡易 XML 指令碼的可使用寬度是 276 個像素。簡易 XML 對話方塊的可使用高度係依據所使用的側邊螢幕元件而定。一個頁面可以提供 768 個像素。至於小工具的高度可以透過側邊螢幕管理判定，並且使用 **GETHMIResolution** 函數詢問。

指令碼語法分析器需要一個功能表來啟動形狀或分支成為其他形狀。主功能表必須命名為 **main**。側邊螢幕不支援軟鍵標籤，因此需要使用形狀的操作控制，瀏覽至其他形狀。

側邊螢幕的頁面或側邊螢幕的小工具數量，並不受限於語法分析器。

3.14.2 整合側邊螢幕對話方塊

使用 **slsidescreen.ini** 檔案整合頁面及小工具。

頁面是在 **[側邊螢幕]** 區段輸入。每個頁面都必須以關鍵字 **PAGE** 指定，其後接著流水號頁碼及必要的屬性。**name** 屬性定義頁面的物件名稱。**implementation** 屬性指定實行程式庫及等級名稱。屬性與屬性之間必須以逗號分隔。**SlSideScreenPage** 類型的頁面可以包括側邊螢幕元素。這是透過「**ELEMENT**」輸入項完成。建立 **ELEMENTxxx** 元素行的規則對應於建立頁面的規則。若是頁面僅包含使用簡易 XML 設定的對話方塊，應該將 **slagmforms.SlEESideScreenPage** 指定為 **implementation** 屬性的值。

小工具可以在 **[Element_<name>]** 區段中新增至側邊螢幕元素。

建立 **WIDGETxxx** 小工具行的規則對應於建立頁面的規則。

若是要啟動簡易 XML 側邊螢幕小工具，應該將 **slagmforms.SlEESideScreenWidget** 指定為 **implementation** 屬性的值。

語法

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

…
…

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

預設情況下，使用小工具識別碼以組成主模組名稱。

範例

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

主要模組名稱：sidescreen_proginfluence.xml

其他屬性

其他屬性可以指派給頁面或小工具，只要在檔案 **slsidescreen.ini** 中輸入區段命名的 **PAGE_<pagename>**、**ELEMENT_<elmentname>** 或 **WIDGET_<widgetname>**。

您可將下列屬性指派給頁面、元素或小工具：

屬性	意義/行為
TextId	與語言無關的文字識別碼
TextFile	文字檔名稱
TextContext	文字關聯 EASY_XML
Icon	圖示的名稱

屬性	意義/行為
maskPath	該屬性指定要使用的主要模組名稱。
maskName	該屬性指定要使用的主要功能表名稱。
focusable	該屬性決定輸入焦點是否可以設定至元素。

範例

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

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

3.14.3 語言及文字管理

欲管理與語言無關的文字，每個元件都能夠指派一個文字檔。文字檔名稱係由元件名稱、語言版本及副檔名「.ts」所組成。

應使用 **EASY_XML** 為文字關聯。

語法

<名稱>_<語言>.ts

範例

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

文字檔名稱：sidescreen_proginfluence_**eng.ts**

若未指定文字檔，語法分析器將搜尋在 **oem_xml_screens_xxx.ts** 預設文字檔內的文字識別碼。

3.14.4 側邊螢幕元件

3.14.4.1 側邊螢幕元素

若頁面連結至預設實行，則元素可以指派給此頁面。於是依此建立一個 **[Page_<page_name>]** 區段，並列出與頁面相關的所有元素。

範例

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

[Page_<page_name>]

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

每個元素需要一個 **[Element_<element name>]** 區段，在區段中列出屬性及相關的小工具。

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

3.14.4.2 側邊螢幕小工具

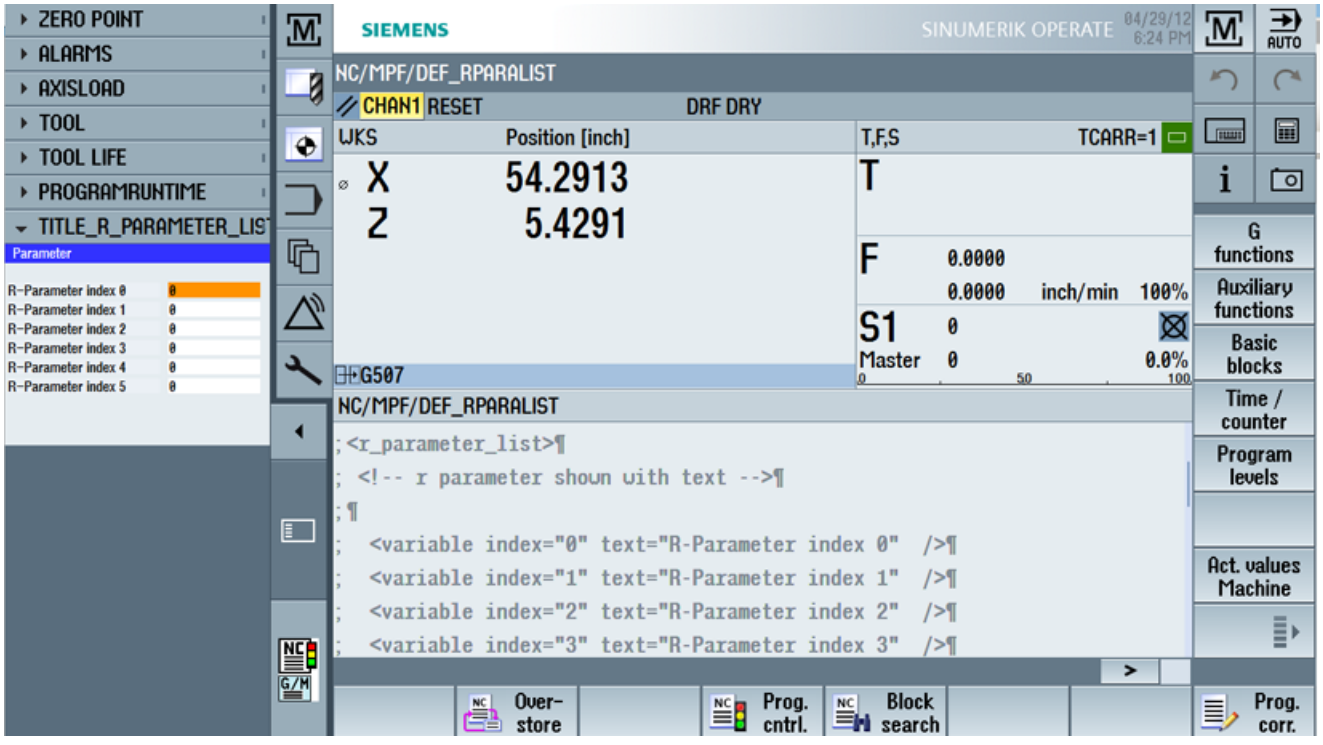
透過側邊螢幕管理指派區域給小工具，以便顯示設定的形狀。在預設狀況下，小工具並不包含輸入焦點，因此無法編輯欄位。此項行為可以藉由指定 **focusable** 屬性加以變更。小工具開啟時，語法分析器會開啟與主功能表相關的格式。在格式之中，使用者可以使用瀏覽指令分支到其他功能表。

範例

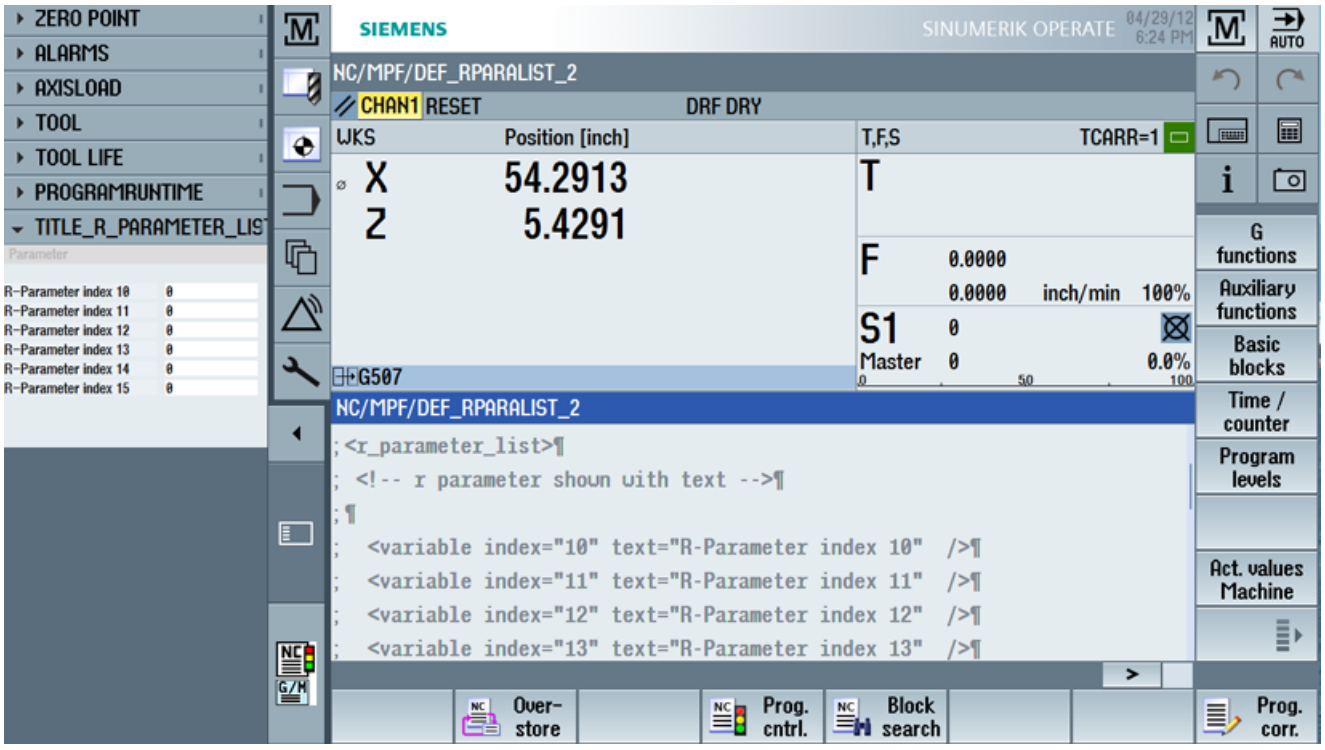
簡易 XML 指令碼搜尋在啟用工件程式中，小工具資料內容的說明。在這個特定範例中，將會顯示含 R 參數的清單說明。每個參數都能夠指派一個說明文字。

工具箱範例

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



若是選擇另一個工件程式，小工具會自動重新作動。


ZERO POINT
ALARMS
AXISLOAD
TOOL
TOOL LIFE
PROGRAMRUNTIME
TITLE_R_PARAMETER_LIS
Parameter
R-Parameter index 10 0
R-Parameter index 11 0
R-Parameter index 12 0
R-Parameter index 13 0
R-Parameter index 14 0
R-Parameter index 15 0
SIEMENS
SINUMERIK OPERATE
04/29/12
6:24 PM
AUTO
NC/MPF/DEF_RPARALIST_2
CHAN1 RESET
DRF DRY
WKS
Position [inch]
T,F,S
TCARR=1
X 54.2913
Z 5.4291
T
F 0.0000
0.0000 inch/min 100%
S1 0
Master 0 0.0%
0 50 100
G507
G functions
Auxiliary functions
Basic blocks
Time / counter
Program levels
Act. values Machine
NC/MPF/DEF_RPARALIST_2
;<r_parameter_list>¶
; <!-- r parameter shoun uith text -->¶
;¶
; <variable index="10" text="R-Parameter index 10" />¶
; <variable index="11" text="R-Parameter index 11" />¶
; <variable index="12" text="R-Parameter index 12" />¶
; <variable index="13" text="R-Parameter index 13" />¶
Over-store
Prog. cntrl.
Block search
Prog. corr.

3.14.4.3 側邊螢幕頁面

頁面係透過 **slagmforms.SIEESideScreenPage** 實行加以啟動，讓格式的整個側邊螢幕區域都能提供使用。沒有標題行，因此無法使用標題陳述來設定表頭。預設情況為頁面並不包含輸入焦點，因此無法編輯欄位。此項行為可以藉由指定 **focusable** 屬性加以變更。頁面開啟時，語法分析器會開啟與主功能表相關的格式。在格式之中，使用者可以使用瀏覽指令分支到其他功能表。

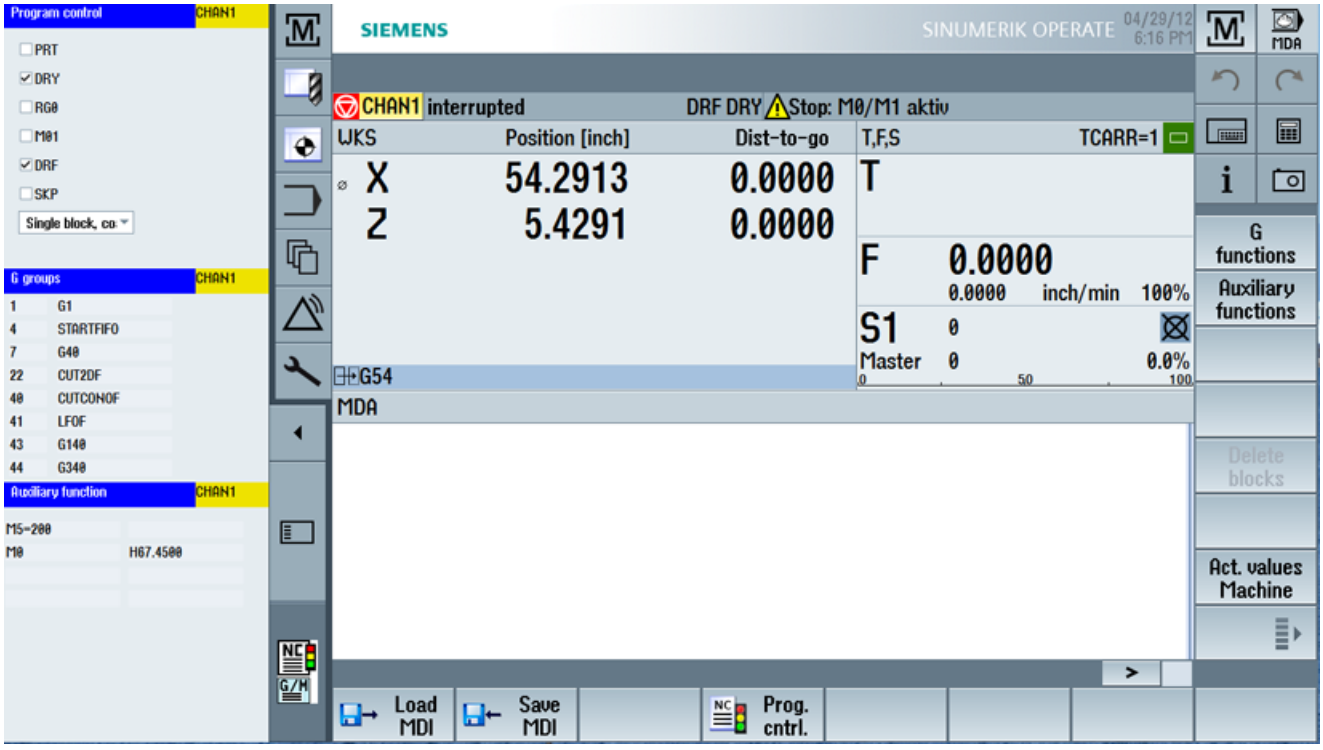
範例

側邊螢幕頁面顯示為只有顯示簡易 XML 格式的頁面。

在此範例中，其他資訊將顯示在側邊螢幕區域。

工具箱範例

Custom Screen Sample\side_screen_examples\sidecreenpage
\sidescreen_proginfluence.xml




```
[Sidescreen]
PROPERTY001= name:=minimizable, type:=bool, value:="true"
```

```
PROPERTY002= name:=buttonBarVisible, type:=bool, value:="true"
PAGE001= name:=WidgetsPage, implementation:=SlSideScreenPage
PAGE004= name:=sidescreen_proginfluence,
implementation:=slagmforms.SlEESideScreenPage

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

文字檔案：sidescreen_proginfluence_deu.ts

3.15 透過 PLC 硬鍵選擇對話方塊

應用領域

以下功能可由 PLC 在作業軟體中初始化：

- 選擇操作區。
- 在操作區中選擇特定情形
- 執行已設定於軟鍵的功能

硬鍵

所有按鍵 (包含 PLC 按鍵) 隨後均成為硬鍵。最多可定義 254 個硬鍵。適用於以下位置：

按鍵數字	應用
按鍵 1 - 按鍵 9	操作面板前的按鍵
按鍵 10 - 按鍵 49	保留
按鍵 50 - 按鍵 254 按鍵 50 - 按鍵 81	PLC 按鍵： 保留給 OEM
按鍵 255	已由控制資訊預先指定。

硬鍵 1 - 9 已預先指定如下：

按鍵指定		動作/作用
HK1	MACHINE（機台）	選擇「機台」操作區域，最後對話方塊
HK2	PROGRAM（程式）	選擇「程式」操作區，最後對話方塊或最後程式
HK3	OFFSET（偏移量）	選擇「參數」操作區，最後對話方塊
HK4	PROGRAM MANAGER（程式管理員）	選擇「程式」操作區，「程式管理員」基本畫面 無功能
HK5	ALARM（警報）	選擇「診斷」操作區，「警報清單」對話方塊
HK6	CUSTOM（自訂）	選擇「自訂」操作區
HK7	MENU SELECT（功能表選擇）	選擇「主功能表」

按鍵指定		動作/作用
HK8	MENU FUNCTION（功能表功能）	選擇「功能」操作區
HK9	MENU USER（功能表使用者）	選擇「使用者」操作區

設定

規劃設定已實現於 [keyconfiguration] 區段中的 systemconfiguration.ini 組態檔案中。每一行均定義各硬鍵事件的內容。硬鍵事件是特定硬鍵的第 n 個動作。例如，特定硬鍵的第二及第三個動作可產生不同的回應。

systemconfiguration.ini 組態檔案中項目可由使用者指定的設定值覆寫。目錄 [系統使用者目錄]/cfg 及 [系統 oem 目錄]/cfg 可用於此用途。

用於設定硬鍵事件的文字行有下列結構：

```
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：硬鍵編號、值域：1 – 254

n：事件編號 - 對應硬鍵的第 n 個動作，數值範圍：0 – 9

需求

PLC 使用者程式必須滿足下列要求：

僅處理一個硬鍵。因此，只有在作業軟體承認了先前的要求之後，才能接受新的要求。如果 PLC 使用者程式從 MCP 鍵衍生硬鍵，則必須為這些硬鍵提供足夠的暫存空間，以確保不會漏失快速的按鍵。

PLC 介面

PLC 介面中有提供用於選擇硬鍵的區域。區域位於 DB19.DBB10。在此，PLC 可直接指定按鍵值，範圍為 50 至 254。

作業軟體的確認動作包括兩個步驟。此程序是必要的，如此當連續輸入兩次相同的按鍵碼時，作業軟體即可正確識別兩個個別的事件。在第一個步驟中，控制資訊 255 將寫入位元組 DB19.DBB10。此已定義的虛擬按鍵啟用可讓 HMI 識別每個獨特的 PLC 按鍵順序。控制資訊

對於 PLC 使用者程式是無意義的，並且不得變更。在第二個步驟中，實際的承認動作會在 PLC 清除 DB19.DBB10 時進行。從這個時間點開始，PLC 使用者程式即可指定新的硬鍵。同時，目前的硬鍵要求將在作業軟體中處理。

範例

設定檔：

```
; configuration of OP hardkeys (KEY1-KEY9) and
; PLC hardkeys (KEY50-KEY254)
[keyconfiguration]
; MACHINE key (hardkey block)
KEY1.0 = area:=AreaMachine, dialog:=SlMachine
; PROGRAM key (hardkey block)
KEY2.0 = area:=AreaProgramEdit
; OFFSET key (hardkey block)
KEY3.0 = area:=AreaParameter
; PROGRAM MANAGER key (hardkey block)
KEY4.0 = area:=AreaProgramManager
; ALARM key (hardkey block)
KEY5.0 = area:=AreaDiagnosis, dialog:=SlDgDialog,
cmdline:="-slGfwHmiScreen SlDgAeAlarmsScreen"
; CUSTOM (hardkey block)
KEY6.0 = area:=Custom
; MACHINE key (optional, Shift + F10)
KEY7.0 = area:=AreaMachine, dialog:=SlMachine,
cmdline:="-MKey"
; MENU USER (hardkey block, Ctrl + F10)
KEY10.0 = area:=MenuUser
; MENU FUNCTION (hardkey block, Ctrl + Shift + F10)
KEY11.0 = area:=MenuFunction
KEY50.0 = area:=CustomXML, dialog:=SlEECustomDialog
```

區域及對話方塊識別碼位於 [系統 Siemens 目錄]/cfg 的 systemconfiguration.ini。

如果只使用 **SlEECustomDialog**，剖析器將預期應用程式目錄內的 **xmldial.xml** 檔案和名稱為 **main** 的功能表標籤。其他檔案名稱或主功能表名稱可用加入 **cmdline** 按鍵建立。-

mainModule 參數定義要載入的 XML 說明。剖析器預期此指令碼內的主功能表。**-entry** 參數定義主功能表的名稱。若缺少此參數，剖析器將使用 **main** 名稱來啟動功能。必須永遠加入值為 **slagmdialog.hmi** 的 **-conf** 參數。

以下為其中一個例子：

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

使用者介面

systemconfiguration.ini 檔案中的以下項目允許在上述功能中使用 Easy XML。

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

3.16 將使用者對話方塊指派給保留的軟鍵

用來擴充現有功能的所有標準操作區都保留了軟鍵。對應的檔案儲存於下列系統目錄，用來啟動及連結使用者專案的 OEM 軟鍵：

/siemens/sinumerik/hmi/template/cfg

這些檔案包含操作區專屬的 XML 說明，會將其轉譯並且在操作軟體啟動時，整合至操作區的功能表樹狀結構中。

對於個別應用程式的整合，對應於操作區的檔案，將從範本目錄複製到以下目錄中，並進行修改：

/oem/sinumerik/hmi/template/cfg

3.16.1 定義語言中立的軟鍵文字

TEXTFILE 標籤包含文字檔的檔案名稱，可用來指派給 OEM 區域。檔名指定時沒有語言及副檔名。

語法

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

範例

oem_xml_screens_deu.ts

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

所要顯示的軟鍵文字是在 **softkey** 標籤中的 **property** 標籤管理。該 OEM 數值將被所要的文字 ID 取代。**translationContext** 屬性參考文字檔內已指派文字的區域。在 easyXML 的情況下，指定 EASY_XML 上下文。

語法

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

範例

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

3.16.2 指派簡易 XML 區域

導覽目標也必須在 **softkey** 標籤中定義。因此，導覽目標 **CustomXML** 必須輸入至 **area** 標籤的 **name** 屬性中。屬性 **dialog** 及 **screen** 會自動定義，因此可以將這些參數刪除。

語法

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

範例

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

若使用導覽目標 **area**，語法分析器將以檔案 xmldial.xml 為啟動模組，以及 **main** 功能表為功能表管理的輸入內容點。為了能夠同時提供多重應用程式，使用 **switchToDialog** 函數。因此，必須使用 FUNCTION 標籤取代 NAVIGATION 標籤。**CustomXML** 區域、**SlEECustomDialog** 對話方塊，及啟動模組與主功能表的名稱，均需傳輸至此函數當作呼叫引數。

然後可以使用 **switchToArea** 標籤，從 easyXML 返回標準使用者區域。

語法

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

範例專案

sldgarea_oem.xml

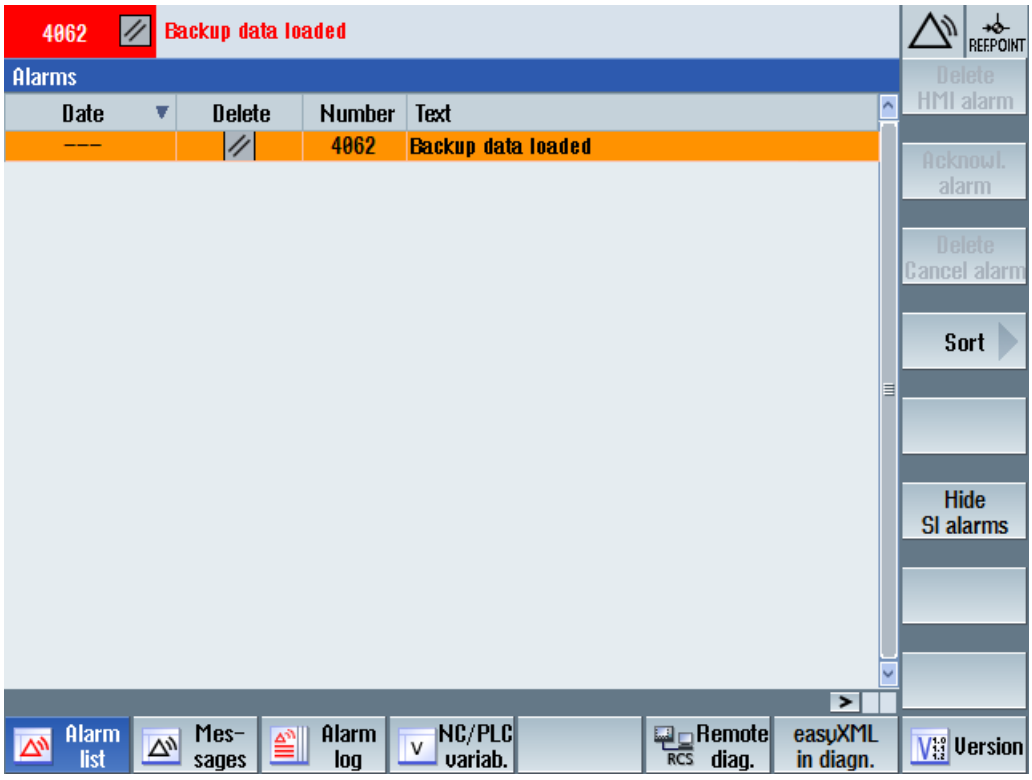
```
<!DOCTYPE DIALOG>
<DIALOG>
<TEXTFILE>oem_xml_screens</TEXTFILE>
<!-- ========================================================================
-->
<!-- OEM softkey number 7 on first horizontal main menu in area diagnosis --
>
<!-- ========================================================================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

或

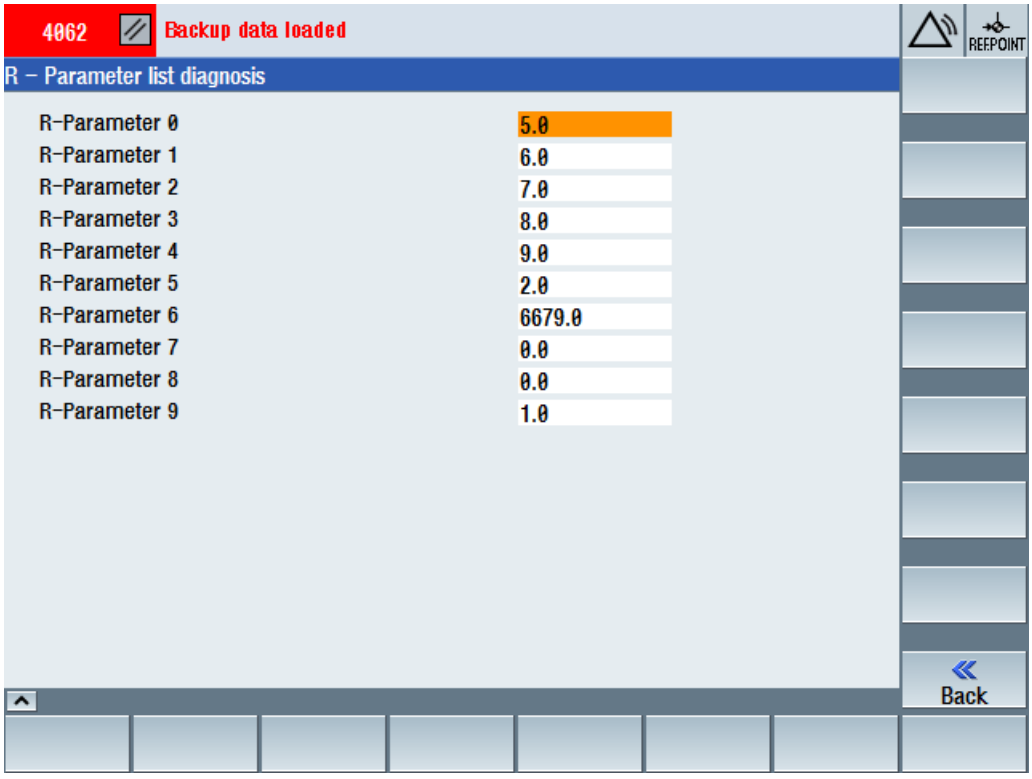
```
<!DOCTYPE DIALOG>
<DIALOG
<!-- ========================================================================
-->
<!-- OEM softkey number 7 on first horizontal main menu in area diagnosis --
>
<!-- ========================================================================
-->
<TEXTFILE>oem_xml_screens</TEXTFILE>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

dg.xml

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


4062
Backup data loaded
REFPOINT
Alarms
Date
Delete
Number
Text

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


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

3.16.3 標籤說明

softkey 標籤

在 **softkey** 標籤之外，軟鍵狀態可以使用 POSITION 屬性，用 **state** 屬性設定。狀態必須設定的軟鍵編號，應指定為屬性值。

範例

```
<menu name = "menu_sktest">

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

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

typedef 標籤

說明

必須移除範例中元素的預先分派。

類型定義內的變數以**元素**標籤做宣告。標籤屬性對應於 **let** 指令的屬性。建立變數時，元素可以預先分派。

範例

RECT：

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

POINT：

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

SIZE：

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

let 標籤

欄位元素的編號係以 **dim** 屬性指定。二維欄位的第一維度與第二維度是以逗號分隔。欄位元件的存取係透過欄位索引，在變數名稱後以方括號指定。陣列的大小可以直接定義，或是透過變數的內容。

範例

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

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

或

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

或

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

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

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

3.17 建立語言中立文字

對話方塊格式中使用的所有文字都是可以管理的語言中立，其中文字識別碼用於對話方塊中，在執行期間，則以目前選用語言的文字取代之。此文字與 UTF8 格式的文字檔中的每種可用語言的文字識別碼儲存在一起。

預設狀況下，檔案 oem_xml_screens_<語言代碼>.ts 會指派給簡易 XML 函數當作文字檔。將 EASY_XML 識別碼指定為文字關聯。

將兩個美元符號拿來當作指令碼中文字識別碼的前置碼，以標記測試識別碼。

結構

oem_xml_screens_deu.ts

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

範例

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

3.18 專案專屬的文字檔

3.18.1 建立專案專屬的文字檔

管理個別的專案時，可以將個別的文字檔用於專案、每個格式及每個功能表。TEXTFILE 屬性會在對應的標籤中提供通知。

文字檔的名稱將作為屬性值傳輸，不含語言識別碼及副檔名。在格式或功能表關閉之前，可以繼續存取文字。如果使用標籤 **DialogGui** 載入文字檔，則可以存取內容，直到退出專案。若是多次使用文字內容，則在上次載入的檔案中搜索文字識別碼。每個文字檔案可以使用一個文字內容。

範例

main_form_screens_deu.ts

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

啟用

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

或

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

或

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

範例對話方塊

main_form_screens_deu.ts

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

main2_sktext_screens_deu.ts

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

form2_screens_deu.ts

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

對話方塊指令碼

xmldial.xml

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

3.18.2 指定文字關聯

可以透過獨立命名的區域名稱，在文字檔中對文字進行分組。區域識別碼是在 **context** 標籤內，用 **name** 標籤定義。

語法

```
<context>
```

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

範例

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

如果使用特定的上下文，則透過 TEXTCONTEXT 屬性，將上下文的名稱與專案、格式或功能表的文字檔檔名一起指定。在設定新的上下文之前，仍然可以存取儲存在上下文中的文字。因此，為專案設定的上下文將替換為為格式或功能表設定的上下文。

專案範例

form2_screens_deu.ts

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

using_textcontext.xml

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

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

3.18.3 指定 Textfilelist

如果要為一個專案使用多個文字檔案，則可以使用 **textfilelist** 屬性將清單中所需文字檔案的名稱轉發給解析器。

文件檔案名稱的表示法是逐行實現的，必須以換行符做結尾。文字檔案的名稱應為不含語言副檔名及檔案副檔名的檔案名稱。

範例

出現在清單中的檔案 `form2_screens_deu.ts` 應為 `form2_screens`。

textfilelist.ini

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

啟用

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

3.19 還原工作階段

關閉 Easy XML 專案時，將啟用所有本地變數，並卸載命令集指令。如果要在關閉控制後保留變數的資料，則將這些變數指定為 **permanent** 屬性。必須在關閉電源之前，將所要保留的資料指定為 **poweroff** 屬性。

如果再次選擇 Easy XML 應用，則可以在功能表中控制更新選擇應用之後所要啟動的功能表或格式。編程人員有責任確保提供這方面所需的所有資料。

如果在 **DialogGUI** 標籤中設定了 **restoresession** 屬性，則解析器將組織重新選擇上次開啟的功能表及上次開啟的格式。編程人員有責任提供必要的資料。

產生調試對話方塊

4

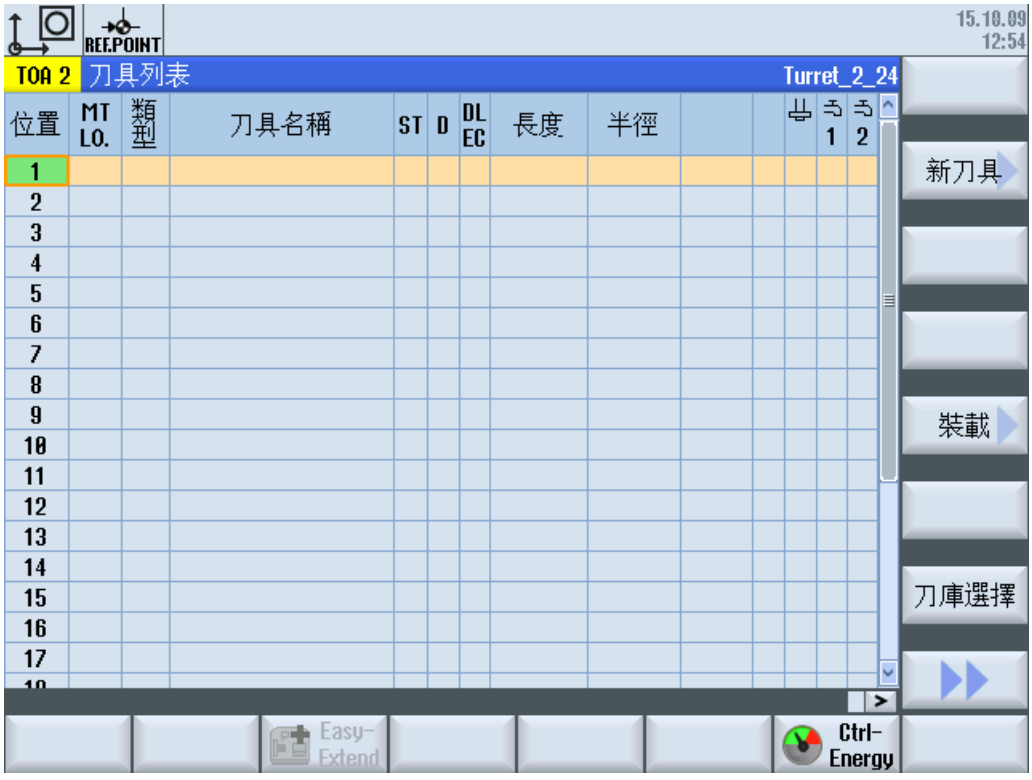
4.1 功能概觀

用途

「簡易擴充」功能可提供簡單的調試、啟用、停用或測試選擇性配備等作業方式。控制系統顯示一份包含可用設備與裝置之狀態的清單。系統最多可管理 64 個裝置。

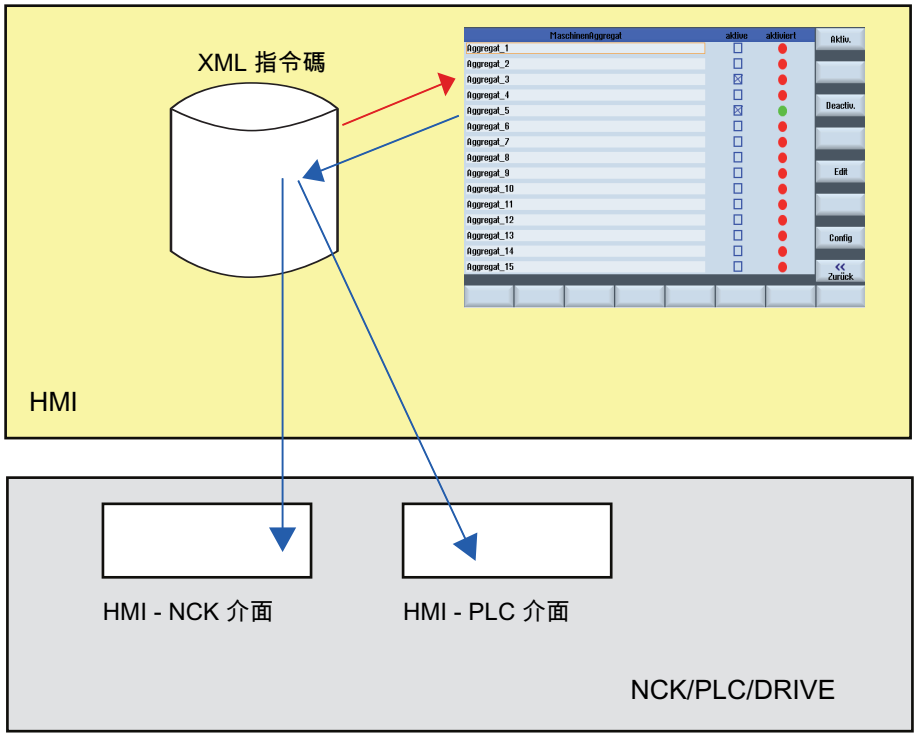
使用軟鍵將裝置啟用及停用。

「簡易擴充」功能可在「參數」操作區中取得。



圖像 4-1 主畫面「參數」

設定



圖像 4-2 「簡易擴充」的運作方式

若要使用「簡易擴充」功能，必須由機台製造商設定好下列功能：

- **PLC ↔ HMI 介面**
  透過使用者介面和 PLC 之間的介面管理選配裝置。
- **指令碼處理**
  機具製造商將調試、啟用、停用或測試裝置的執行程序儲存在指令碼陳述中。
- **參數對話方塊 (選用)**
  參數對話方塊顯示儲存在指令碼檔案中的裝置資訊。

檔案儲存

屬於「簡易擴充」的檔案儲存於系統 CF 卡的「dvm」目錄中（機台製造商）。

檔案	名稱	目標目錄
文字檔案	oem_aggregate_xxx.ts	\oem\sinumerik\hmi\lng
指令碼檔案	agm.xml	\oem\sinumerik\hmi\dvm

檔案	名稱	目標目錄
封存檔	任意	\oem\sinumerik\hmi\dvm\archives
PLC 使用者程式	任意	PLC

4.2 PLC 使用者程式的設定

載入設定資料

將建立好的設定資料連同指令碼和文字檔一起傳送到控制器的製造商目錄中。此外，對應的 PLC 使用者程式也必須載入。

設備編程設定

使用 PLC 使用者程式，透過資料區塊 DB9905，在操作員元件和 PLC 之間進行通訊，其中保留了 128 個字組供裝置管理使用。資料單節指定器的說明可參閱章節「PLC_INTERFACE (頁 310)」。

PLC 字組從裝置 1 開始指派：

資料區塊	裝置名稱
DB9905.DBB0	裝置 1
DB9905.DBB4	裝置 2
...	...
DB9905.DBB192	裝置 49
DB9905.DBB196	裝置 50

以 4 個位元組表示每一個裝置的狀態，各位元組含義如下：

位元組	Bit	說明	
0	0	== 1	裝置已經啟動 (HMI 確認)
	1	== 1	準備將裝置啟用 (HMI 請求)
	2	== 1	準備將裝置停用 (HMI 請求)
	3-7	保留	
1	0-7	保留	
2	0	== 1	裝置生效 (PLC 確認)
	1	== 1	裝置發生錯誤
	2-7	保留	
3	0-7	裝置的唯一識別碼	

檔案儲存

Easy Extend 檔案儲存在系統 CompactFlash 卡的「oem」〈製造商〉及「oem_i」〈個別〉目錄中。

說明

檔案名稱只能包含小寫字母。

檔案	名稱	目標目錄
文字檔案	oem_aggregate_xxx .ts	/oem/sinumerik/hmi/lng/ /oem_i/sinumerik/hmi/lng/
指令碼檔案	agm.xml	/oem/sinumerik/hmi/dvm /oem_i/sinumerik/hmi/dvm
封存檔	任意	/oem/sinumerik/hmi/dvm/archives /oem_i/sinumerik/hmi/dvm/archives
PLC 使用者程式	任意	PLC

一般順序

機具製造商必須執行下列步驟，以便提供所需的參數：

1. 在 PLC 上建立 PLC 使用者程式，用以在啟動時將裝置啟用。
2. 執行「標準機台」調試，然後將參數儲存在序列啟動封存 (series startup archive) 中。
3. 安裝裝置，進行調試，然後將參數讀出，成為差異序列啟動封存 (differential series startup archive)。

   說明

   變更機台設定

   假使有必要編輯驅動機台參數，此編輯必須先在控制器中實施。必須針對所有裝置和裝置群重複此程序。

增加軸

如果機台上有增添機台軸，則以固定的順序安裝驅動物件 (DO) 是很重要的，因為序列啟動封存中包含機具製造商的參考機台機群，順序一旦改變，序列啟動封存便無法適用。

針對「控制器元件」(control components) 建議選擇下列設定：

- NC 資料
- PLC 資料
- 驅動資料
  - ACX 格式 (二進位)

4.3 使用者介面的畫面顯示

使用者介面上的對話方塊

下列對話方塊可供「簡易擴充」功能使用：

- 控制器提供一個**可設定的對話方塊**，其中顯示可用的裝置。
- 若首次調試尚未執行過，則控制器開啟**調試對話方塊**。

如果已經程式設計調試程序（XML 指令：「START_UP」），並且裝置仍未調試，則控制器會開始調試步驟。

此程序包括，在讀入儲存於指令碼檔中的序列啟動封存之前，先執行完整的資料備份。

發生錯誤時，調試工程師可決定是否要回復 (roll back) 調試程序，還是要手動改正機台設定中可能的錯誤。

- 可利用「Cancel」功能早期取消調試。控制器會將先前儲存的調試檔案複製回來。

若機台必須在調試成功完成之後關機，則可使用 XML 陳述 "POWER_OFF" 將對應的訊息輸出到控制器上。

4.4 建立語言相關文字

設定檔的結構

包含語言相關文字的 XML 檔案必須以 UTF8 格式建立：

範例

oem_aggregate_eng.ts

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

oem_aggregate_deu.ts

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TS>
<TS>
<context>
    <name>my_context</name>
    <message>
    <source>DEVICE_ONE</source>
    <translation>First device</translation>
    </message>
    <source>DEVICE_TWO</source>
    <translation>Second device</translation>
    </message>
</context>
</TS>
```

4.5 動力單元範例

啟用驅動器物件

以下所要啟用的驅動器物件已經由機具製造商調試過並再予以停用，藉以將這些軸列為可選配備。

若要將軸啟用，請執行下列步驟：

- 以 P0105 啟用驅動器物件。
- 啟用通道機械資料中的第二軸。
- 以 P0971 備份驅動器機台參數。
- 等候參數寫入。
- 重新啟動 NCK 和驅動器。

編程設定：

```
<DEVICE>
  <list_id>1</list_id>
  <name> 「啟動驅動器」</name>

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

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

</DEVICE>
```

啟用由 PLC 控制的裝置

利用輸出位元組 10 進行裝置定址，利用輸入位元組 9 發訊給 PLC 告知參數記錄已就緒。

將輸出位元組設為指定的啟用編碼。While 迴圈等候裝置的參數記錄就緒。

編程設定：

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

4.6 指令碼語言

說明

於 建立使用者對話方塊 (頁 25) 功能中所說明的所有指令碼元件組成「簡易擴充」功能的基礎。定義其他指令碼元件可以管理其他裝置。

指令碼程式的各部份

指令碼區分為下列區域：

- 「簡易擴充」識別碼
- 定義裝置能夠執行動作的框架
- 用於裝置的識別碼
- 用於調試裝置的識別碼
- 用於啟用裝置的識別碼
- 用於停用裝置的識別碼
- 用於測試裝置的識別碼
- 用於參數對話方塊的識別碼

以下章節詳細說明各標籤。

說明

識別碼 <tag>	含義
AGM	用於「啟動精靈」的識別碼
DEVICE	用於描述裝置的識別碼。 **屬性：** • **option_bit** 針對管理選項，指派一個固定的位元編號給裝置。
NAME	此識別碼指定要在對話方塊中顯示的裝置名稱。 若使用文字參照，則對話方塊顯示針對此識別碼儲存的文字。
START_UP	此識別碼包含關於裝置調試程序的說明。

識別碼 <tag>	含義
SET_ACTIVE	此識別碼包含關於裝置啟用程序的說明。 **屬性：** • **timeout** 屬性允許以秒為單位指定逾時。經過這個時間之後如果指令碼仍未完成，系統將中斷處理。
SET_INACTIVE	此識別碼包含關於裝置關機程序的說明。 **屬性：** • **timeout** 屬性允許以秒為單位指定逾時。經過這個時間之後如果指令碼仍未完成，系統將中斷處理。
TEST	此識別碼包含用於測試裝置操作能力的陳述。 **屬性：** • **timeout** 屬性允許以秒為單位指定逾時。經過這個時間之後如果指令碼仍未完成，系統將中斷處理。
UID	唯一的數值識別碼，用於識別 PLC↔ HMI 介面中的裝置。
VERSION	用於版本的識別碼

函數執行之否定確認

利用自動提供的變數「$actionresult」，系統可將否定執行結果告知 XML 剖析器。若數值設為零，則剖析器取消執行中的函數處理。

範例

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>                                  用於「啟動精靈」的識別碼
<DEVICE>
  <NAME> 裝置 1 </NAME>                用於裝置的識別碼
  <START_UP>                           用於調試裝置的識別碼
  …
  </START_UP>
  <SET_ACTIVE>                         用於啟用裝置的識別碼
  …
  </SET_ACTIVE>
```

```
    <SET_INACTIVE>                    用於停用裝置的識別碼
    …
    </SET_INACTIVE>
    <TEST>                            用於測試裝置的識別碼
    …
    </TEST>
  </DEVICE>
  …
  </AGM>
```

4.6.1 CONTROL_RESET

說明

使用此識別碼，可重新啟動控制元件。 必須控制器已經恢復循環操作，指令碼 (script) 才會繼續執行。

編程設定

識別碼：	**CONTROL_RESET**	
語法：	<CONTROL_RESET resetnc="TRUE" />	
屬性：	resetnc="true"	NC 元件重新啟動。
	resetdrive="true"	驅動元件重新啟動。

4.6.2 FILE

說明

識別碼啟用標準備檔以便讀入或建立。

- 讀入封存：
  讀入封存時，必須指定封存的檔案名稱。
- 建立封存：
  若指定屬性 create= "true" ，則此功能會以指定的名稱建立一個標準封存 (*.arc)，並把檔案儲存在 .../dvm/archives 目錄中。

編程設定

識別碼：	**FILE**	
語法：	<file name ="<archive name>" />	
	<file name ="<archive name>" create="true" class="<data classes>" group="<area>" />	
屬性：	name	用於檔案名稱的識別碼

create	以指定的名稱在 …/dvm/archives/ 目錄中建立調試封存。
group	指定要包含在封存中的資料群組。 若要儲存多個資料群組，則各群組之間應以空格分開。 下列資料群組可包含在封存中： • NC • PLC • HMI • DRIVES

範例

```
<!-- 建立資料類別群組 -->
<file name="user.arc" create="true"
group="nc plc hmi" />

<!–將備檔讀入控制器中->
<file name="user.arc" />
```

4.6.3 OPTION_MD

說明

使用此識別碼，可重新定義可選的機台參數。 交貨時，系統是使用 MD14510 $MN_USER_DATA_INT[0] 到 $MN_USER_DATA_INT[3]。

若 PLC 使用者程式管理這些可選參數，則必須在資料區塊或 GUD 中提供適當的資料字組。

此參數由位元構成。 從位元 0 開始，以固定的方式將各位元指派給表列裝置，亦即，位元 0 指派給裝置 1，位元 1 指派給裝置 2 等。如果所管理的裝置超過 16 個，則透過區域索引指派裝置群組 1-3 的位址標記。

說明

轉換值域

MD14510 $MN_USER_DATA_INT[i] 的值域是從 -32768 到 +32767。若要透過機台參數對話方塊逐一將裝置啟用，必須將位元的組合轉換成十進位值。

編程設定

識別碼： **OPTION_MD**

語法： 區域 0：

<option_md name = "Address identifier of the data" />

或：

<option_md name = "Address identifier of the data" index= "0"/>

區域 1 到 3：

<option_md name = "Address identifier of the data" index= "Area index"/>

屬性：		
	name	用於位址的識別碼，例如 $MN_USER_DATA_INT[0]
	index	用於區域索引的識別碼： 0 (預設值) 裝置 1 到 16 1: 裝置 17 到 32 2: 裝置 33 到 48 3: 裝置 49 到 64

4.6.4 PLC_INTERFACE

說明

使用此識別碼，可重新定義 PLC ↔ HMI 介面。系統預期 128 個可定址的字組。

預設值：DB9905

編程設定

識別碼： **PLC_INTERFACE**

語法： <plc_interface name = "Address identifier of the data" />

屬性：		
	name	用於位址的識別碼，例如 "plc/mb170"

範例：`plc/mb170`

4.6.5 POWER_OFF

說明

此識別碼用於產生提示操作者關閉機台的訊息。 本訊息文字永久儲存在系統中。

編程設定

識別碼：	**POWER_OFF**
語法：	<power_off />
屬性：	--

4.6.6 WAITING（等待中）

說明

NC 或 驅動器重置之後，須等候這些元件分別重新啟動。

編程設定

識別碼：	**WAITING**	
語法：	<WAITING WAITINGFORNC ="TRUE" />	
屬性：	waitingfornc="true"	等 NC 重新啟動。
	waitingfordrive="true"	等驅動器重新啟動。

4.6.7 用於對話方塊的 XML 標記

用於設定參數的對話方塊

可針對每一個裝置設定一個對話方塊，以便能夠在執行時間設定或輸出額外的參數。按 "Additional parameters" 軟鍵來顯示此對話方塊。

章節 建立使用者對話方塊 (頁 25) 中所說明的所有命令集元件可以用來產生對話方塊。

範例

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>
<DEVICE>
  <NAME> 裝置 1 </NAME>
  <START_UP>
  …
  </START_UP>
  <SET_ACTIVE>
  …
  </SET_ACTIVE>
…
  <FORM name="<對話框名稱>">                使用者對話方塊識別碼
    <INIT>
      <CONTROL name = "edit1" .../>      輸入欄位識別碼
    </INIT>
    <PAINT>                                文字或圖像顯示識別碼
      <TEXT>hello world !</TEXT>
    </PAINT>
  </FORM>

</DEVICE>
…
</AGM>
```

4.6.8 SOFTKEY_OK、SOFTKEY_CANCEL

說明

識別碼 SOFTKEY_OK 在利用 "OK" 軟鍵關閉對話方塊時將標準行為覆寫。 識別碼 SOFTKEY_CANCEL 在利用 "CANCEL" 軟鍵關閉對話方塊時將標準行為覆寫。

下列功能可在此識別碼內執行：

- 資料操控
- 條件式處理程序
- 迴圈處理

編程設定

識別碼：	**SOFTKEY_OK**
語法：	<SOFTKEY_OK>
	...
	</SOFTKEY_OK>
識別碼：	**SOFTKEY_CANCEL**
語法：	<SOFTKEY_CANCEL>
	...
	</SOFTKEY_CANCEL>

索引