SIEMENS

SINUMERIK

SINUMERIK 828D
Easy XML

프로그래밍 매뉴얼

적용 대상:

CNC 소프트웨어 버전 4.95

07/2021
6FC5398-3EP40-0LA0

법률상의 주의

경고사항

본 메뉴얼에는 여러분 자신의 안전과 재산 손실을 방지하기위해 여러분이 지켜야할 주의사항이 담겨있습니다. 여러분의 안전에 관련된 주의사항은 안전 경고 심볼로 강조되어있으며, 재산 손실에 관련된 주의사항은 안전 경고 심볼이 없습니다.

⚠ 위험
피하지 않으면 사망 또는 심각한 부상을 초래할 수 있는 절박한 위험 상황을 나타냅니다.

⚠ 경고
피하지 않으면 사망 또는 심각한 부상을 초래할 수 있는 잠재적인 위험 상황을 나타냅니다.

⚠ 주의
예방조치를 적절하게 취하지 않을 경우 경미한 인명 피해가 발생할 수 있음을 나타냅니다.

유의사항
예방조치를 적절하게 취하지 않을 경우 재산 피해가 발생할 수 있음을 나타냅니다.

여러 위험 수준이 적용될 때에는, 항상 가장 높은 레벨(낮은 번호)의 알림이 표시됩니다. 안전 경고 심볼이 인적 손실을 나타내는 경우, 재산 손실을 경고하는 또 다른 알림이 추가될 수도 있습니다.

자격을 가진 자

본서가 대상으로 하는 제품/시스템은 반드시 자격을 가진 자가 취급하는 것으로 하고, 각 조작 내용에 관련하는 문서,특히 안전상의 주의 및 경고가 준수되지 않으면 안됩니다. 자격을 가진 자란 훈련 내용 및 경험을 토대로 하면서 해당 제품/시스템의 취급에 동반하는 위험성을 인식하고, 발생할 수 있는 위해를 사전에 회피할 수 있는 자를 가리킵니다.

시멘스 제품의 올바른 사용을 위해

다음에 주의하십시오:

⚠ 경고
시멘스 제품은 카탈로그 및 부속의 기술 설명서의 지시에 따라 사용해 주십시오. 타사의 제품 또는 부품과 함께 사용하는 것은 당사의 권장 또는 허가가 있을 경우에 한합니다. 제품의 올바르고 안전한 사용을 위해 적절한 운반, 보관, 조립, 설치, 배선, 시동, 조작, 보수를 시행하고 있습니다. 사용할 때에는 허용된 범위를 꼭 지켜 주십시오. 부속의 기술 설명서에 기술되어있는 지시를 엄수해 주십시오.

상표

® 표시는 Siemens AG의 등록상표입니다. 본 문서의 기타 표시는 특정 목적으로 제삼자가 사용하는 경우, 지적 재산권을 해칠 수 있는 상표입니다.

책임의 포기

저희는 기술된 하드웨어와 소프트웨어가 본 메뉴얼의 내용물과 일치하는 것을 확인했습니다. 편차가 발생하는 것을 완전히 배제할 수는 없으므로, 완전히 동일하다고는 보장할 수 없습니다. 그렇지만, 메뉴얼의 데이터는 정기적으로 검토되며, 필요한 수정은 다음의 수정판에 반영됩니다. 품질 개선을 위한 의견은 환영합니다.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
독일

문서 주문번호: 6FC5398-3EP40-0LA0
Ⓟ 07/2021 변경될 수 있습니다

목차

소개 1

1.1 SINUMERIK 정보

단순하고 표준화된 CNC 장비에서 프리미엄 모듈형 장비 설계에 이르기까지 SINUMERIK CNC는 모든 장비 유형에 적합한 솔루션을 제공합니다. 개별 부품이든 대량 생산이든, 단순한 공작물이든 복잡한 공작물이든 상관없이 SINUMERIK은 프로토타입 제작과 공구 설계에서 금형 제작과 대규모 연속 생산에 이르는 모든 생산 영역에 고도로 동적인 자동화 솔루션을 제공합니다.

웹 사이트에서 SINUMERIK (https://www.siemens.com/sinumerik)에 대한 더욱 자세한 정보를 제공합니다.

1.2 문서 정보

대상 그룹

이 문서의 대상 독자는 다음과 같습니다.

- 프로그래머
- 프로젝트 엔지니어

이점

본 프로그래밍 매뉴얼은 대상 그룹이 XML 기반 스크립트 요소를 사용하여 고객별 및 애플리케이션별 사용자 인터페이스를 설계할 수 있도록 지원합니다.

표준 범위

이 문서에서는 표준 버전의 기능에 대해서만 설명합니다. 실제로 제공되는 시스템의 기능 범위와 다를 수 있습니다. 제공된 드라이브 시스템의 기능만 주문 문서를 참조하십시오.

이 문서에서 설명하지 않는 시스템의 다른 기능을 실행할 수 있습니다. 하지만 이 조항이 새로운 시스템이나 서비스 수행 시 이러한 기능을 제공해야 한다는 것을 의미하지는 않습니다.

이 문서에 모든 제품 유형에 대한 상세 정보가 모두 포함되어 있지는 않습니다. 또한 이 문서는 가능한 모든 유형의 설치, 작동 및 서비스/유지보수 방법에 대해 설명하고 있지는 않습니다.

장비 제조업체는 제품의 추가 사항이나 변경 사항을 직접 문서화해야 합니다.

타사 웹 사이트

이 문서에 타사 웹 사이트의 하이퍼링크가 포함될 수 있습니다. Siemens는 타사 웹 사이트와 웹 사이트 콘텐츠에 대해 책임을 지지 않습니다. Siemens는 타사 웹 사이트에 표시되는 정보를 통제하지 않으며 제공된 콘텐츠와 정보에 대해 책임을 지지 않습니다. 사용자는 이러한 콘텐츠와 정보의 사용 위험을 감수해야 합니다.

1.3 인터넷에서 제공되는 문서

1.3.1 SINUMERIK 828D 매뉴얼 개요

828D 문서 개요 (https://support.industry.siemens.com/cs/ww/en/view/109766724)에서 SINUMERIK 828D 버전 4.8 SP4 이상의 기능에 관한 종합적인 문서를 제공합니다.

문서를 PDF 및 HTML5 형식으로 표시하거나 다운로드할 수 있습니다.

문서는 다음과 같은 카테고리로 나뉩니다.

- 사용자: 작동
- 사용자: 프로그래밍
- 제조업체/서비스: 구성
- 제조업체/서비스: 스타트업
- 제조업체/서비스: 기능
- 제조업체/서비스: Safety Integrated
- SINUMERIK Integrate/MindApp
- 정보 및 교유

1.3.2 문서 개요 SINUMERIK 조작반 구성요소

문서 개요 SINUMERIK 조작반 구성요소 (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW)에서 SINUMERIK 조작반 구성요소에 관한 종합적인 문서를 제공합니다

문서를 PDF 및 HTML5 형식으로 표시하거나 다운로드할 수 있습니다.

문서는 다음과 같은 카테고리로 나뉩니다.

- 화면 조작반
- 기계 조작반
- 기계 푸시버튼 조작반
- 핸드헬드 장치/미니 핸드헬드 장치
- 추가 조작반 구성요소

SINUMERIK 개요 - 주제 페이지 (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW)에서 SINUMERIK에 관한 가장 중요한 문서, 항목 및 링크의 개요를 제공합니다.

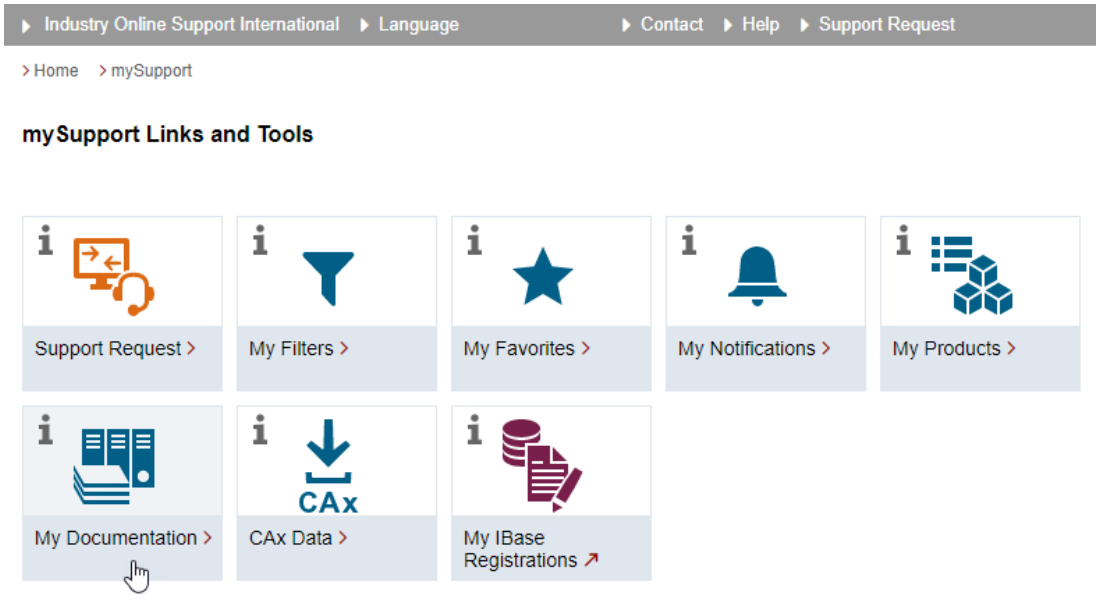
1.4 기술 문서에 대한 피드백

Siemens Industry Online Support에 게시되는 기술 문서에 대해 질문, 제안 또는 수정 사항이 있을 경우 항목 끝에 표시되는 "피드백 보내기" 링크를 사용하십시오.

1.5 mySupport 문서

"mySupport 문서" 웹 기반 시스템을 사용하면 Siemens 콘텐츠를 기반으로 자신만의 장비 문서를 편집할 수 있습니다.

애플리케이션을 시작하려면 mySupport 홈 페이지 (https://support.industry.siemens.com/cs/ww/en/my)에서 "내 문서" 타일을 클릭하십시오.



구성한 문서를 RTF, PDF 또는 XML 형식으로 내보낼 수 있습니다.

참고

mySupport 문서 애플리케이션을 지원하는 Siemens 콘텐츠에는 "구성" 링크가 있습니다.

1.6 서비스 및 지원

제품 지원

제품 정보는 인터넷에서 확인할 수 있습니다.

제품 지원 (https://support.industry.siemens.com/cs/ww/en/)

이 주소로 이동하면 다음이 제공됩니다.

- 최신 제품 정보(제품 소개)
- FAQ(자주 묻는 질문)
- 매뉴얼
- 다운로드
- 최신 제품 정보를 소개하는 뉴스레터
- 사용자와 전문가가 정보와 모범 사례를 공유하는 글로벌 포럼
- Siemens 데이터베이스의 현지 담당자 연락처(→ "연락처")
- 현장 서비스, 수리, 예비 부품 등에 관한 정보(→ "현장 서비스")

기술 지원

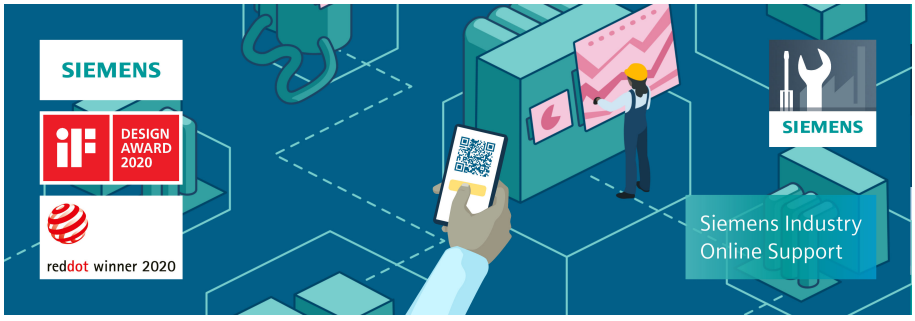
국가별 기술 지원 전화번호는 웹 사이트의 "연락처" 섹션에 있는 주소 (https://support.industry.siemens.com/cs/ww/en/sc/4868)에서 확인할 수 있습니다.

기술적인 질문이 있는 경우 "지원 요청"의 온라인 양식을 사용하십시오.

교육

다음 주소 (https://www.siemens.com/sitrain)에서 SITRAIN 관련 정보를 확인할 수 있습니다.
SITRAIN은 Siemens의 자동화 및 드라이브 제품, 시스템 및 솔루션에 대한 교육 과정을 제공합니다.

이동 중의 Siemens 지원



수상 경력이 있는 "Siemens Industry Online Support" 앱을 사용하면 언제 어디서나 Siemens Industry 제품에 관한 30만 개 이상의 문서에 액세스할 수 있습니다. 이 앱은 다음을 지원합니다.

- 프로젝트 구현 시 문제 해결
- 오류 발생 시 문제 해결
- 시스템 확장 또는 새 시스템 계획

또한 기술 포럼과 다른 전문가 문서에도 액세스할 수 있습니다.

- FAQ
- 적용 예
- 매뉴얼
- 인증서
- 제품 소개 등

"Siemens Industry Online Support" 앱은 Apple iOS 및 Android에서 사용할 수 있습니다.

명판의 데이터 매트릭스 코드

명판의 데이터 매트릭스 코드에는 특정 장치 데이터가 포함되어 있습니다. 이 코드를 스마트폰으로 읽으면 장치에 대한 기술 정보가 "Industry Online Support" 모바일 앱을 통해 표시됩니다.

1.7 중요 제품 정보

OpenSSL 사용

이 제품에는 다음과 같은 소프트웨어가 포함될 수 있습니다.

- OpenSSL 툴킷에서 사용하기 위해 OpenSSL 프로젝트에 의해 개발된 소프트웨어
- Eric Young이 개발한 암호화 소프트웨어
- Eric Young이 개발한 소프트웨어

자세한 정보는 인터넷에서 확인할 수 있습니다.

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

일반 데이터 보호 규정 준수

Siemens는 표준 데이터 보호 원칙, 특히 데이터 최소화 규칙(privacy by design)을 준수합니다.

이 제품의 경우 다음을 의미합니다.

이 제품은 개인 데이터는 처리 또는 저장하지 않고 기술 기능 데이터(예: 타임스탬프)만 처리 또는 저장합니다. 사용자가 이 데이터를 다른 데이터(예: 교대 계획)에 연결하거나 개인 관련 데이터를 동일한 데이터 매체(예: 하드 디스크)에 저장해 이 데이터를 개인화할 경우, 해당 데이터 보호 조항을 준수해야 합니다.

기본 안전 지침 2

2.1 일반 안전 지침

⚠ 경고

기타 에너지원으로 인한 감전 및 생명 위험

전압이 흐르는 부품을 만지면 심각한 부상을 당하거나 사망에 이를 수 있습니다.

- 전기 장치의 조작은 유자격자만 허용됩니다.
- 항상 국가별 안전 규칙을 준수하십시오.

일반적으로 안전을 위해 다음 단계가 적용됩니다.

1. 연결 해제를 준비하십시오. 이 절차에 의해 영향을 받을 모든 사람에게 이를 알리십시오.
2. 전원 공급 장치에서 드라이브 시스템을 분리하고 다시 켜지지 않도록 조치를 취하십시오.
3. 경고 라벨에 규정된 방전 시간이 경과할 때까지 기다리십시오.
4. 전원 연결부 사이와, 전원 연결부와 보호 도체 연결부 사이에 전압이 없는지 확인하십시오.
5. 기존의 보조 전원 공급 회로에 전류가 흐르지 않는지 확인하십시오.
6. 모터가 구동되지 않음을 확인하십시오.
7. 기타 모든 위험한 에너지원 (예: 압축 공기, 유압 시스템 또는 물) 을 확인하십시오. 에너지원을 안전 상태로 전환하십시오.
8. 드라이브 시스템이 완전히 잠겼는지 확인하십시오.

작업을 완료한 후 역순으로 작동 준비 상태로 되돌리십시오.

⚠ 경고

부적합한 전원 공급 장치의 연결로 인한 감전

장비를 부적합한 전원 공급 장치에 연결하면 노출된 콤포넌트에 위험한 전압이 흐를 수 있습니다. 위험한 전압에 접촉하면 심각한 부상을 당하거나 사망에 이를 수 있습니다.

- 모듈의 모든 연결과 터미널에는 SELV (Safety Extra Low Voltage) 또는 PELV (Protective Extra Low Voltage) 출력 전압을 제공하는 전원 공급 장치만 사용하십시오.

⚠ 경고

장비 손상으로 인한 감전

장비를 부적절하게 취급하면 장비가 손상될 수 있습니다. 손상된 장치의 경우, 외함 또는 노출된 콤포넌트에 위험한 전압이 흐를 수 있습니다. 만지면 심각한 부상을 당하거나 사망에 이를 수 있습니다.

- 운송, 보관 및 조작 중 기술 데이터에 규정된 한계 값을 준수하는지 확인하십시오.
- 손상된 장치는 사용하지 마십시오.

⚠ 경고

연결되지 않은 케이블 실드로 인한 감전

연결되지 않은 케이블 실드로 인한 정전식 누화 결합을 통해 위험한 접촉 전압이 발생할 수 있습니다.

- 적어도 케이블 실드와 한쪽 끝을 사용하지 않는 케이블 코어의 경우 접지된 하우징 전위에 연결하십시오.

⚠ 경고

접지 연결이 없는 경우의 감전

보호 등급 I의 장치에서 보호 컨덕터 연결이 없거나 부적절하게 수행된 경우, 개방되고 노출된 부품에는 만지면 심각한 부상을 당하거나 사망에 이를 수 있는 고전압이 흐를 수 있습니다.

- 장치를 해당되는 규정에 따라 접지하십시오.

유의사항

부적절한 체결 공구로 인한 장비 손상

부적절한 체결 공구나 체결 방법을 사용하면 장비 나사가 손상될 수 있습니다.

- 나사 머리와 정확하게 일치하는 드라이버만 사용하십시오.
- 기술 문서에 명시된 토크로 나사를 조이십시오.
- 동적 토크 센서와 속도 제한 시스템이 장착된 토크 렌치나 기계식 정밀 너트 러너를 사용하십시오.

⚠ 경고

내장 장치로부터의 화재 확산

화재가 발생하면 내장 장치의 외함은 화재와 연기의 확산을 막을 수 없습니다. 그 결과 심각한 부상을 당하거나 재산 피해가 발생할 수 있습니다.

- 화재와 연기로부터 인력을 보호할 수 있도록 내장 장치를 적절한 금속 캐비닛에 설치하거나, 다른 인력 보호 조치를 취하십시오.
- 연기는 제어되고 모니터링된 경로를 통해서만 배출되도록 하십시오.

⚠ 경고

무선 장치나 휴대 전화로 인한 예기치 않은 장비 동작

구성 요소 바로 근처에서 무선 장비나 휴대 전화를 사용하면 장비 오작동이 발생할 수 있습니다. 오작동은 장비의 기능 안전에 영향을 주어 인명을 위험에 빠뜨리거나 재산 피해를 초래할 수 있습니다.

- 따라서 구성 요소에 20 cm보다 가깝게 다가갈 경우 반드시 무선 장비나 휴대 전화를 끄십시오.
- 이미 꺼진 장비에서만 "SIEMENS Industry Online Support App"을 사용하십시오.

⚠ 경고

환기 공간 부족으로 인한 화재

불충분한 환기 공간은 콤포넌트의 과열로 인한 화재 및 연기를 초래할 수 있습니다. 이 경우 심각한 부상을 당하거나 심지어 사망에 이를 수 있습니다. 또한 가동 중단 시간이 증가하고 장치/시스템의 사용 수명이 단축될 수도 있습니다.

- 각 콤포넌트에 대해 환기 공간으로 규정된 최소 여유 공간을 준수하는지 확인하십시오.

유의사항

허용되지 않는 장착 위치로 인한 과열

허용되지 않는 위치에 장착하면 기기가 과열되어 손상될 수 있습니다.

- 허용되는 장착 위치에서만 기기를 작동하십시오.

⚠ 경고

안전 기능 비활성화로 인한 예기치 않은 장비 동작

안전 기능이 비활성화되었거나 조정되지 않으면 예기치 않은 장비 동작이 발생해 심각한 부상을 당하거나 사망에 이를 수 있습니다.

- 스타트업 전에 적절한 제품 문서에 있는 정보를 준수하십시오.
- 모든 안전 관련 콤포넌트를 포함해 전체 시스템에서 안전과 관련된 기능의 안전 검사를 수행하십시오.
- 드라이브와 자동화 작업에 사용되는 안전 기능이 적절한 설정을 통해 조정되고 활성화되어 있는지 확인하십시오.
- 기능 테스트를 수행하십시오.
- 설비는 안전과 관련된 기능이 올바르게 작동하고 있다는 것을 확인한 후에만 가동하십시오.

참고

Safety Integrated 기능에 대한 중요 안전 지침

Safety Integrated 기능을 사용하려는 경우 Safety Integrated 매뉴얼의 안전 지침을 준수해야 합니다.

⚠ 경고

파라미터 설정의 오류나 변경으로 인한 장비 오작동

부정확하거나 잘못 수정된 파라미터로 인해 기계가 고장 나거나 부상이나 인명 피해가 발생할 수 있습니다.

- 무단 액세스로부터 파라미터를 보호하십시오.
- 오작동이 발생하면 적절한 조치(예: 비상 정지 또는 비상 차단)를 취해 대응하십시오.

2.2 전기장 또는 정전기 방전으로 인한 장비 손상

정전기 민감 장치 (ESD) 란 전기장 또는 정전기 방전으로 인해 손상될 수 있는 개별 부품, 집적 회로, 모듈 또는 장치를 의미합니다.

유의사항

전기장 또는 정전기 방전으로 인한 장비 손상

전기장 또는 정전기 방전은 개별 부품, 집적 회로, 모듈 또는 장치의 손상을 통해 오작동을 초래할 수 있습니다.

- 전자 부품, 모듈 또는 장치는 원래 포장재 또는 기타 적절한 재료 (알루미늄 포일의 전도성 발포 고무) 만 사용해서 포장, 보관, 운송 및 배송하십시오.
- 콤포넌트, 모듈 및 장치는 다음 방법 중 하나로 사용자가 접지된 경우에만 만지십시오.
  - ESD 손목띠 착용
  - 전도성 바닥이 있는 ESD 영역에서 ESD 단화 또는 ESD 접지 스트랩 착용
- 전자 부품, 모듈 또는 장치는 전도성 표면 (ESD 표면을 갖춘 테이블, ESD 폼 전도체, ESD 외장재, ESD 수송 컨테이너) 위에 두어야만 합니다.

2.3 적용 예제의 보증 및 책임

적용 예제는 구속력이 없으며, 구성, 장비 또는 발생할 수 있는 상황에 대해 완전하지 않습니다. 적용 예제는 특정 고객의 솔루션을 나타내지 않으며, 일반적인 작업에 지원을 제공하는 용도로만 이용됩니다.

설명된 제품을 정확하게 작동할 책임은 사용자 자신에게 있습니다. 적용 예제는 장비를 사용, 설치, 작동 및 유지보수할 때 안전한 취급에 대한 사용자의 책임을 면제하지 않습니다.

2.4 보안 정보

Siemens는 공장, 시스템, 기계류 및 네트워크의 안전한 작동을 지원하는 산업 보안 기능과 함께 제품 및 솔루션을 제공합니다.

공장, 시스템, 기계류 및 네트워크를 사이버 위협으로부터 보호하기 위해서는 통합적인 최첨단의 산업 보안 개념을 구현하고 지속적으로 유지 관리해야 합니다. Siemens의 제품과 솔루션은 이러한 전체적인 개념의 한 요소만을 구성합니다.

고객은 고객 소유의 공장, 시스템, 기계류 및 네트워크에 대한 무단 액세스를 방지할 책임이 있습니다. 시스템, 기계류 및 구성 요소는 필요한 범위 내에서, 그리고 적절한 보안 조치가 취해진 상태(예: 방화벽 및 네트워크 세그멘테이션 사용)에서만 회사 네트워크 또는 인터넷에 연결되어야 합니다.

이행될 수 있는 산업 보안 조치에 대한 자세한 내용은 https://www.siemens.com/industrialsecurity (https://www.siemens.com/industrialsecurity)를 참조하십시오.

Siemens의 제품과 솔루션은 지속적인 개발을 통해 보안을 더욱 강화하고 있습니다. Siemens는 제품 업데이트가 나오는 즉시 적용하고 항상 최신 제품 버전을 사용할 것을 강력히 권장합니다. 더 이상 지원되지 않는 제품 버전을 사용하거나, 최신 업데이트를 적용하지 않으면 사이버 위협에 노출될 가능성이 높아집니다.

제품 업데이트에 대한 정보를 제때 받아보려면 https://www.siemens.com/industrialsecurity (https://new.siemens.com/global/en/products/services/cert.html#Subscriptions)에서 Siemens 산업 보안 RSS 피드를 구독하십시오.

추가 정보는 인터넷에서 제공합니다:

산업 안전 환경설정 매뉴얼 (https://support.industry.siemens.com/cs/ww/en/view/108862708)

⚠ 경고

악의적인 소프트웨어 조작으로 인한 안전하지 않은 작동 상태

악의적인 소프트웨어 조작(예: 바이러스, 트로이 목마, 웜)은 시스템 운영 상태를 위험에 빠뜨려 사망, 심각한 부상, 재산상의 피해를 야기할 수 있습니다.

- 소프트웨어를 최신 상태로 유지하십시오.
- 설비 또는 기계의 자동화 및 드라이브 콤포넌트를 총체적인 첨단 산업 보안 개념에 통합시키십시오.
- 반드시 모든 설치 제품들을 총체적 산업 보안 개념에 포함시키십시오.
- 바이러스 검사 프로그램 같은 적절한 보호 조치를 통해 교환 가능 저장 매체에 저장된 파일을 악성 소프트웨어로부터 보호하십시오.
- 스타트업이 완료되면 모든 보안 관련 설정을 확인하십시오.

2.5 파워 드라이브 시스템의 잔류 위험

관련 현지 규정(예: EU 기계류 지침)에 따라 기계 또는 시스템 관련 위험을 평가할 때 장비 제조사나 시스템 설치자는 드라이브 시스템의 제어 및 드라이브 콤포넌트와 관련된 다음과 같은 잔류 위험을 고려해야 합니다.

1. 다음과 같은 경우 스타트업, 작동, 유지관리 및 보수 중에 구동 중인 기계 또는 시스템 콤포넌트의 원치 않는 움직임을 유발할 수 있습니다.
   - 센서, 제어 시스템, 작동기, 케이블, 연결부 등의 하드웨어 또는 소프트웨어 오류
   - 제어 시스템 또는 드라이브의 응답 시간 오류
   - 사양에 맞지 않는 작동 또는 환경 조건
   - 결로 및 오염으로 인한 누전
   - 파라미터 설정, 프로그래밍, 배선 및 설치 오류
   - 전자 콤포넌트 가까이에서 무선 장치나 휴대 전화의 사용
   - 외부 영향 또는 손상
   - X선, 이온화(전리) 방사선 및 우주 복사
2. 화염을 포함한 이상 고온과 빛, 노이즈, 입자, 기체 등의 방출이 다음으로 인한 고장 조건에서 콤포넌트 내부와 외부에 발생할 수 있습니다.
   - 콤포넌트 고장
   - 소프트웨어 오류
   - 사양에 맞지 않는 작동 또는 환경 조건
   - 외부 영향 또는 손상
3. 다음과 같은 원인에 의해 위험한 전기 충격이 발생될 수 있습니다.
   - 콤포넌트 고장
   - 정전기 방전으로 인한 영향
   - 움직이는 모터의 유도 전압
   - 사양에 맞지 않는 작동 또는 환경 조건
   - 결로 및 오염으로 인한 누전
   - 외부 영향 또는 손상
4. 작동 중 생성되는 전기, 자기 또는 전자기장의 경우 심박조율기, 보철 또는 기타 금속 이식물을 지닌 사람에게 위험을 초래할 수 있습니다.
5. 시스템의 부적절한 작동 또는 콤포넌트의 적절치 못한 폐기로 인해 환경 오염 요소 또는 가스를 방출할 수 있습니다.
6. 리플 컨트롤 트랜스미터와 같이 네트워크에 연결된 통신 시스템이나 네트워크를 통한 데이터 통신의 영향이 발생할 수 있습니다.

드라이브 시스템 콤포넌트의 잔류 위험에 대한 자세한 내용은 사용자 기술 매뉴얼의 해당 부분을 참조하십시오.

사용자 대화 상자 생성하기 3

3.1 기능의 범위

개요

"사용자 대화 상자 생성" 함수는 개방 구조를 제공하여 사용자가 SINUMERIK Operate에서 사용자별 및 애플리케이션별 사용자 인터페이스를 개발할 수 있도록 지원합니다.

이 제어 시스템은 사용자 대화 상자 생성을 위한 XML-기준 스크립트 언어를 제공합니다.

이 스크립트 언어는 기계별 메뉴 및 대화 양식을 SINUMERIK Operate의 <CUSTOM> 영역에 표시할 수 있도록 지원합니다.

모든 대화 양식은 언어-중립성에 기초하여 설계될 수 있습니다. 그런 경우, 시스템은 수반하는 언어 데이터베이스로부터 표시되어야 하는 텍스트를 판독합니다.

사용

정의된 XML 명령은 다음과 같은 속성을 가지고 있습니다:

1. 다음 요소를 가지고 있는 대화 상자를 표시합니다.
   - 소프트 키
   - 변수
   - 텍스트 및 도움말 텍스트
   - 그래픽 및 도움말 텍스트
2. 다음을 수행하여 대화 상자를 호출합니다.
   - (시작) 소프트 키를 누름
3. 대화를 동적으로 재구성합니다.
   - 소프트 키 편집 및 삭제
   - 변수 필드 정의 및 설계
   - 디스플레이 텍스트 (언어 종속 또는 언어 중립) 를 삽입, 교환 및 삭제
   - 그래픽 삽입, 교환 및 삭제
   - 실행 가능한 스크립트 부분을 동적으로 생성하여 스크립트 처리에 포함시킵니다.

4. 아래 작업에 응답하여 운전 개시
   – 대화 상자 표시하기
   – 값 (변수) 입력하기
   – 소프트 키 선택하기
   – 대화 상자 끝내기
5. 대화간의 데이터 교환
6. 변수
   – 읽기 (NC, PLC 및 사용자 변수)
   – 쓰기 (NC, PLC 및 사용자 변수)
   – 수학, 비교 또는 논리 연산자와 결합
7. 함수 실행:
   – 서브프로그램
   – 파일 함수
   – PI 서비스
8. 사용자 그룹에 따라 사용 권한을 적용합니다.

스크립트 언어에 대한 유효 요소 (태그) 는 "XML tags" (페이지 45) 섹션에 설명되어 있습니다.

참고

아래의 "구성에 관한 기본 원칙"은 XML(Extensible Markup Language)에 관한 포괄적인 설명으로서 만들어진 것이 아닙니다. 추가 정보에 대해서는 관련 전문 서적을 참조하십시오.

3.2 환경 설정 기초

환경 설정 파일

새 사용자 인터페이스에 대한 설명은 구성 파일에 저장됩니다. 이 파일은 자동으로 해석되어 그 결과가 스크린에 표시됩니다. 환경 설정 파일은 공급된 소프트웨어에 저장되지 않고 반드시 사용자에 의해 설정된 후 로드되어야 합니다.

XML 편집기 또는 또 다른 양식의 텍스트 편집기를 사용하여 환경 설정 파일을 생성할 수 있습니다.

참고

파일 이름에는 소문자만 사용됩니다.

메뉴 트리 원리

여러 개의 상호 연결된 대화 상자가 메뉴 트리를 생성합니다. 하나의 대화 상자에서 또 다른 대화 상자로 전환할 수 있다면 링크가 존재하는 것입니다. 이 대화 상자에서 새로 정의된 수평/수직 소프트 키를 사용하여 선행 대화 또는 다른 대화 상자를 호출할 수 있습니다.

설정된 시작 소프트 키를 이용해 시작 메뉴 다음에 추가 메뉴 트리를 생성할 수 있습니다.

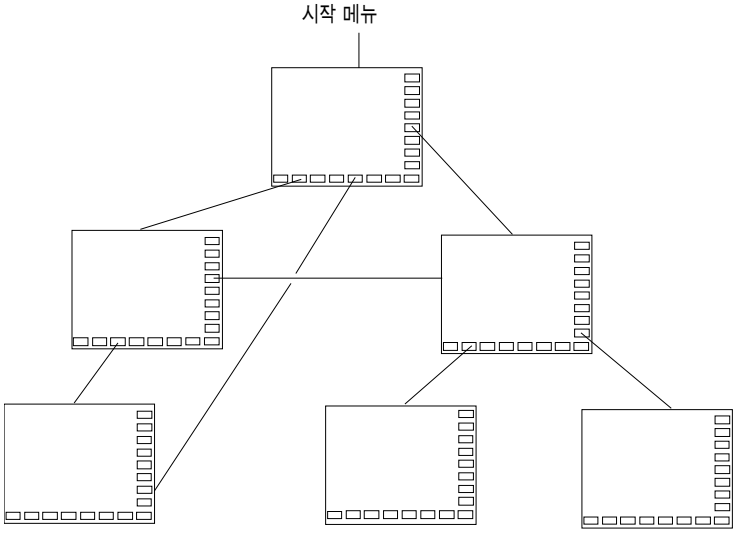


그림 3-1 사용자 대화 메뉴 트리

시작 메뉴

시작 메뉴는 "xmldial.xml" 파일에 "main"이라는 이름으로 정의됩니다. 시작 메뉴는 자체 조작 순차를 개시하는 데 사용됩니다.

메인 메뉴를 사용하면 추가 작업 실행을 위해 로드해서 사용할 자체 대화 상자나 추가 소프트키 바를 링크할 수 있습니다.

아래 그림은 제어 시스템에 있는 제조사 폴더 "System CF-Card/oem/sinumerik/hmi"를 보여줍니다.

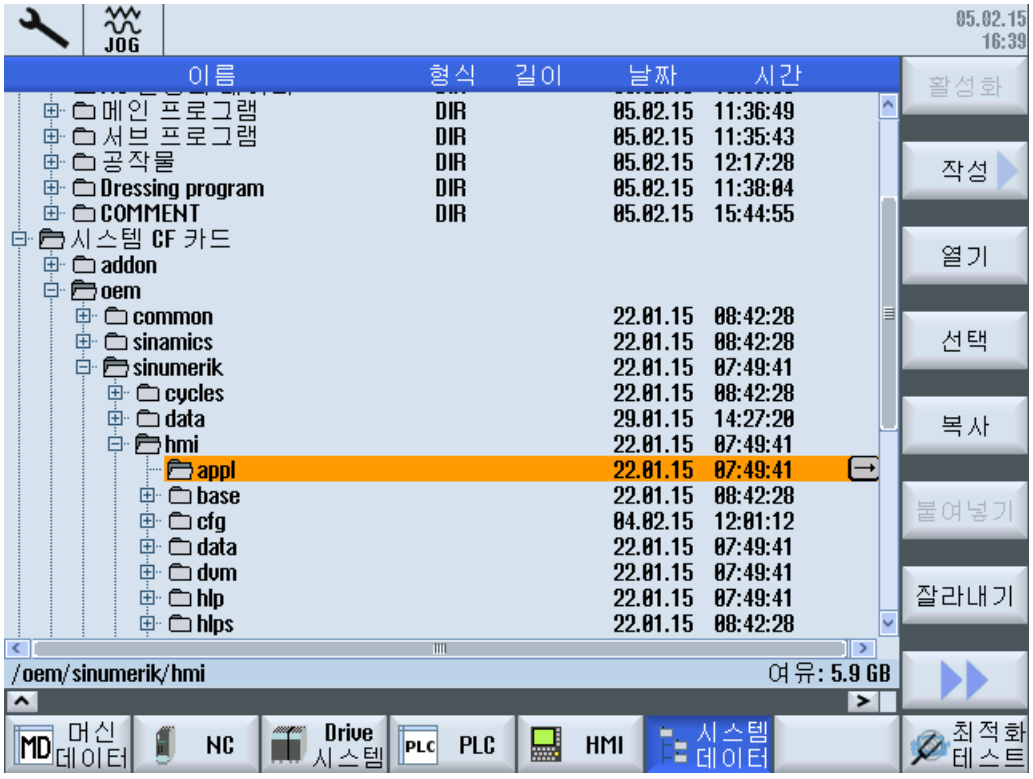


그림 3-2 OEM 디렉토리

제어 시스템의 OEM 디렉토리 "System CF-Card/oem/sinumerik/hmi"에 있는 다음 파일은 사용자 대화 상자를 구성하는 데 필요합니다.

도표 3-1 환경 설정 파일

파일 유형	파일 이름	의미	설정 > 시스템 데이터 영역의 저장 위치
스크립트 파일	"xmldial.xml"	이 스크립트 파일은 XML 태그를 사용해 SINUMERIK Operate의 구성된 소프트 키 메뉴와 대화 상자 양식이 <CUSTOM> 영역에 나타나는 방식을 제어합니다.	OEM 디렉토리 > 애플리케이션 서브 디렉토리 "appl"
텍스트 파일	"oem_xml_screens_xxx.ts"	고객 영역의 표준 텍스트 파일 이 텍스트 파일은 메뉴 텍스트와 개별 언어용 대화 상자 양식을 가지고 있습니다.	OEM 디렉토리 > 언어 서브 디렉토리 "lng"
비트맵	---	제어 시스템은 BMP와 PNG 포맷을 지원합니다.	OEM 디렉토리 > 서브 디렉토리 "ico"
"INCLUDE" XML 태그와 함께 "xmldial.xml" 제어 파일에 삽입된 XML 파일	예: "machine_settings.xml"	이 파일에는 SINUMERIK Operate에 대화 상자 양식 및 파라미터를 표시하기 위해 프로그램된 명령도 포함되어 있습니다.	OEM 디렉토리 > 애플리케이션 서브 디렉토리 "appl"

Easy XML 연결 포인트

Easy XML 스크립트에 다음 연결 포인트가 제공됩니다.

하드 키/소프트 키	연결 포인트
고객	표준 스크립트 파일 이름 "xmldial.xml"

소프트 키 작동 메뉴	표준 스크립트 파일 이름 "xmldial.xml"
	"slamconfig.ini" 파일에서 활성화됨
	
	예:
	`[CustomXML]`
	`TextId=SL_AM_CUSTOM`
	`SidescreenTextId=SL_SIDESCREEN_CUSTOM`
	`TextFile=slam`
	`TextContext=SlAmAreaMenu`
	`SoftkeyPosition = 12`
	`Visible=true`

사용자 메뉴	표준 스크립트 파일 이름 "menu_user.xml"
메뉴 기능	표준 스크립트 파일 이름 "menu_function.xml"

PLC 하드 키	스크립트 파일 이름은 키 설명과 관련됩니다.
	
	예:
	`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 명령어	스크립트 파일 이름은 NC 명령어 설명과 관련됩니다. **예:** `MMC("POPUPDLG, PICTURE_ON, TEST.XML, cmd1","A)`

하드 키/소프트 키	연결 포인트

예약된 작업 영역 소프트 키	스크립트 파일 이름은 메뉴 설명과 관련됩니다.
	
	예:
	`<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>`

사이드스크린	스크립트 파일 이름은 "slsidescreen.ini" 파일의 사이드스크린 설명과 관련됩니다.
Easy Extend 소프트 키	표준 스크립트 파일 이름 "agm.xml"

빠른 선택 키

PPU의 조작반 전방에 있는 빠른 선택 키 <MENU USER> 및 <MENU FUNCTION>은 어떤 사용자 대화창에도 할당될 수 있습니다. 스크립트 파일 이름 "menu_user.xml" 및 "menu_function.xml"은 이 목적에 대해 사용할 수 있습니다.

사용자는 스크립트 파일을 공구박스에서 제조사의 디렉토리로 복사할 수 있습니다. 또는 이름을 가진 자신만의 파일을 생성할 수 있습니다. "main" 이름은 입력 메뉴로 정의됩니다.

3.3 환경 설정 파일

소개

아래 그림은 제어 시스템에 있는 제조사 폴더 "System CF-Card/oem/sinumerik/hmi"를 보여 줍니다.

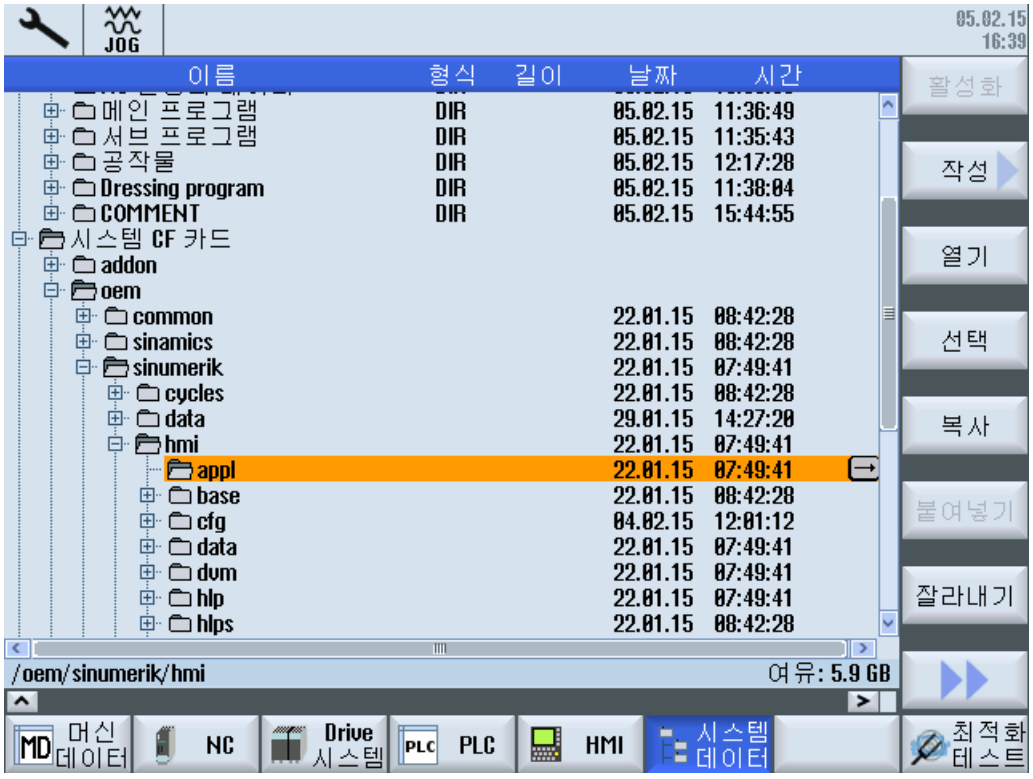


그림 3-3 제조사 폴더

제어 시스템에 있는 제조사 폴더 "System CF-Card/oem/sinumerik/hmi"에서 다음 파일은 사용자 대화 상자를 구성하는데 필요합니다.

도표 3-2 환경 설정 파일

파일 유형	파일 이름	의미	설정 > 시스템 데이터 영역의 저장 위치
스크립트 파일	"xmldial.xml"	이 스크립트 파일은 XML 태그를 사용해 SINUMERIK Operate의 구성된 소프트 키 메뉴와 대화 상자 양식이 <CUSTOM> 영역에 나타나는 방식을 제어합니다.	제조사 폴더 > 어플리케이션 서브 디렉토리 "appl"
텍스트 파일	"oem_xml_screens_xxx.ts"	이 텍스트 파일은 메뉴 텍스트와 개별 언어용 대화 상자 양식을 가지고 있습니다.	제조사 폴더 > 언어 서브 디렉토리 "lng"
비트맵		제어 시스템은 BMP와 PNG 포맷을 지원합니다.	제조사 폴더 > 서브 디렉토리 "ico" 비트맵은 제어 시스템의 화면 해상도용 서브 디렉토리에 저장됩니다. 참고: 비트맵 파일 경로가 규정되어 있을 경우, 파일을 이 디렉토리에 바로 저장할 수 있습니다.
"INCLUDE" XML 태그와 함께 "xmldial.xml" 제어 파일에 삽입된 XML 파일	예: "machine_settings.xml"	이 파일에는 SINUMERIK Operate에 대화 상자 양식 및 파라미터를 표시하기 위해 프로그램된 명령도 포함되어 있습니다.	제조사 폴더 > 어플리케이션 서브 디렉토리 "appl"

사용자 대화 상자 환경 설정 파일의 종속도

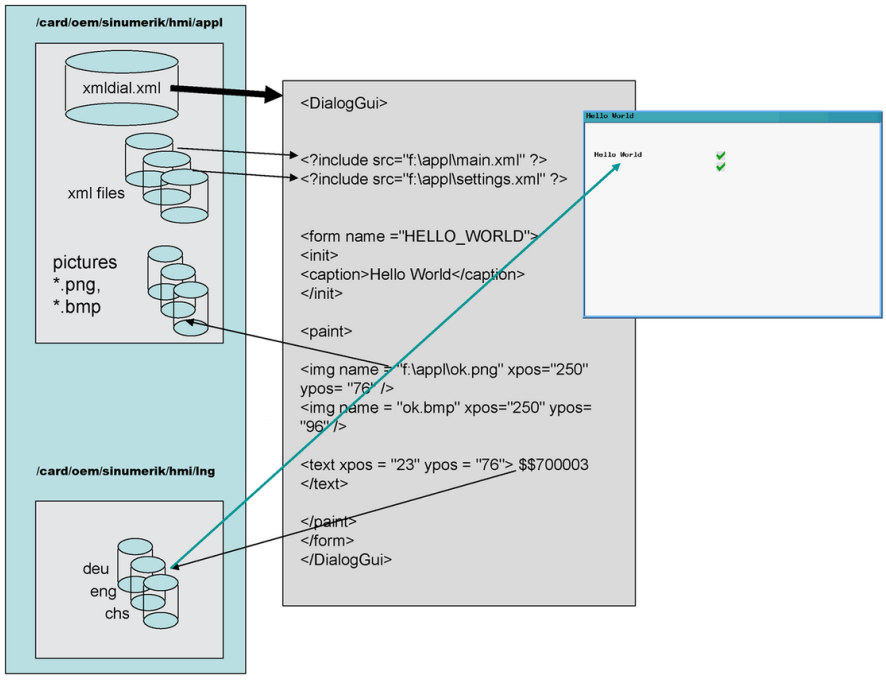


그림 3-4 종속도

로드 환경 설정

앞서 "설정 파일" 표의 "영역의 저장 위치" 열에서 설명했듯이 생성된 파일이 제조사 폴더의 해당 서브 디렉토리로 복사되어야 합니다.

참고

어플리케이션 서브 디렉토리에 스크립트 파일 "xmldial.xml"이 있을 경우 사용자는 즉시 <CUSTOM> 영역에서 이 사용자 대화 상자를 시작할 수 있습니다.
복사 프로세스를 실행 한 후 시스템의 전원을 OFF/ON 해야 합니다.

SINUMERIK Operate의 사용자 대화 상자의 예

<CUSTOM> 영역이 호출된 경우 설정된 소프트키 메뉴가 표시됩니다. 이로 인해 사용자는 환경 설정된 대화 상자 양식을 조작할 수 있습니다.

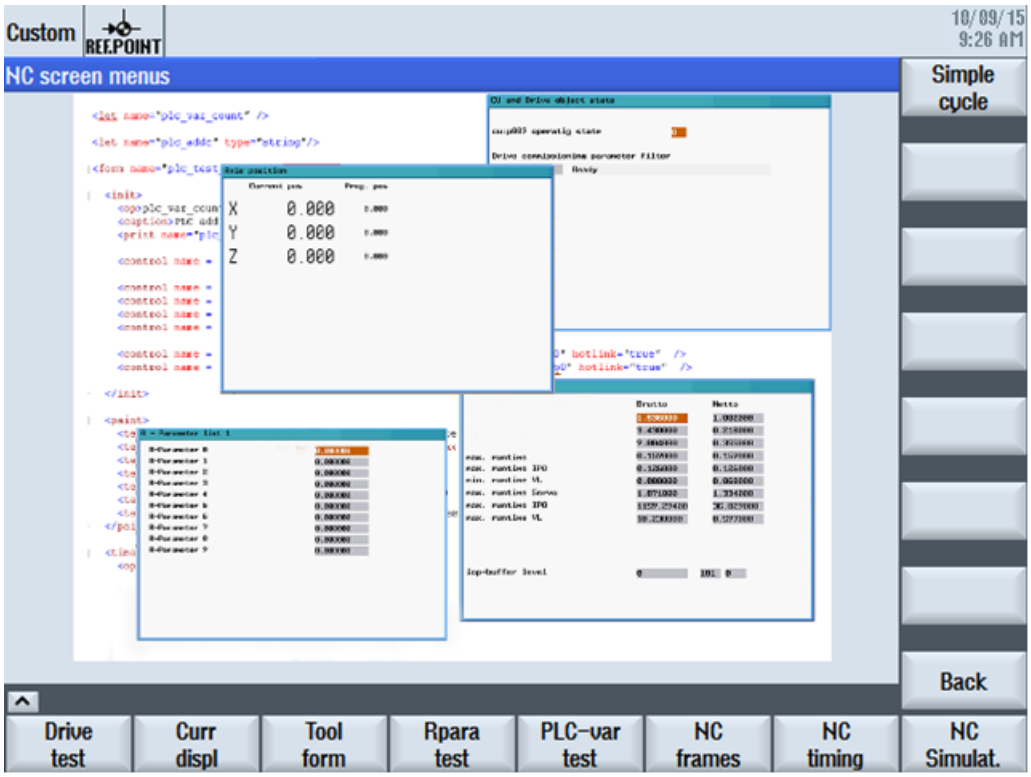


그림 3-5 <CUSTOM> 영역의 사용자 대화 상자의 예

추가 예제는 도구 상자에서 제공됩니다.

3.4 환경 설정 파일 구조

개요

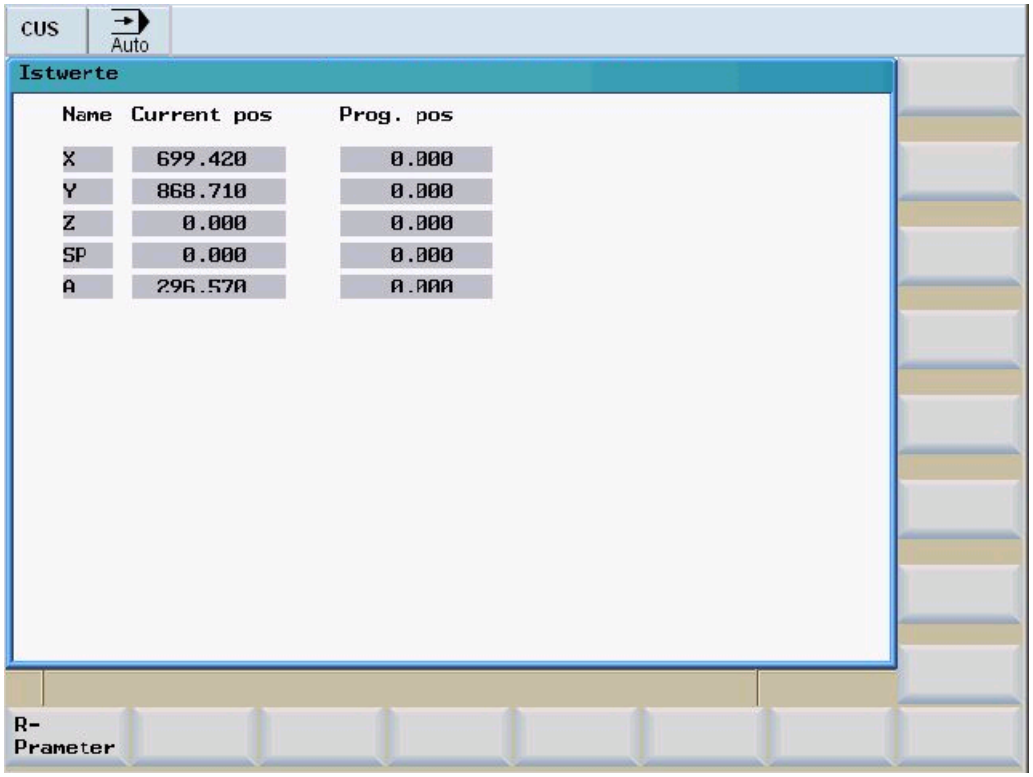
구성 파일은 다음과 같은 요소로 구성됩니다.

- 시작 소프트 키와 "main" 시작 메뉴에 대한 설명
- 대화상자 정의
- 변수 정의
- 블록 설명
- 소프트 키 바의 정의

아래 예는 "xmldial.xml" 의 XML 스크립트와 해당 스크린샷을 보여줍니다.

스크립트는 R 파라미터 목록뿐만 아니라 실제 값 및 잔여 거리 표시를 위한 대화를 포함합니다.

대화 상자 양식 섹션 "실제 값"



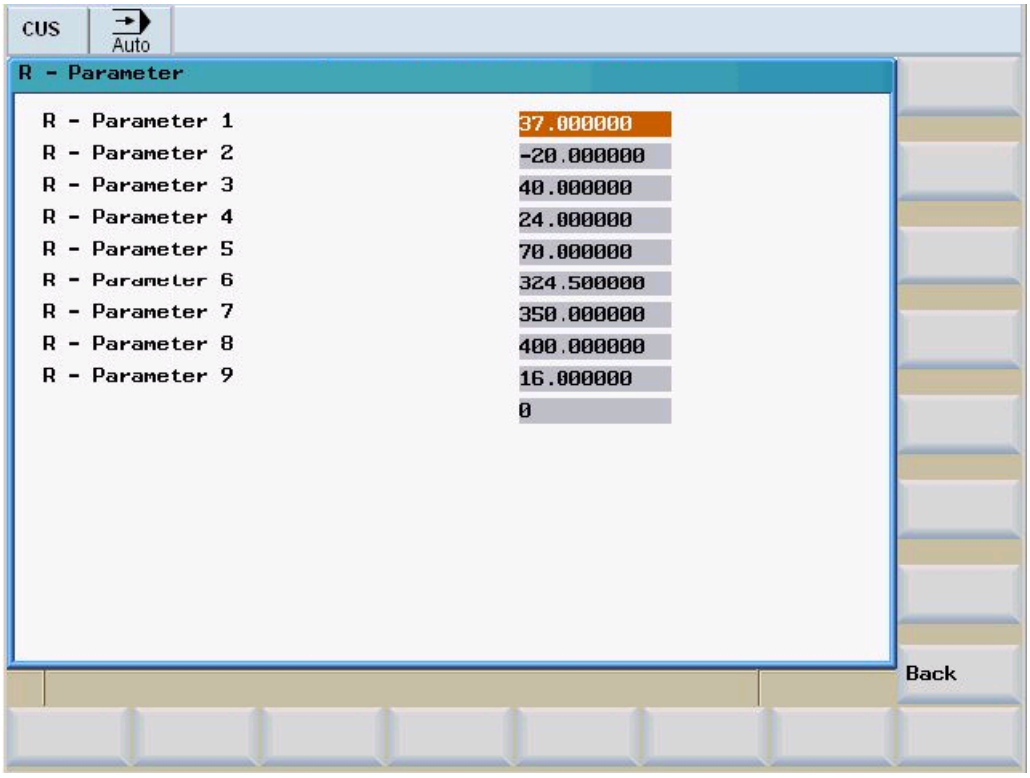
xmldial.xml

```
<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>

.....
```

대화 상자 양식 섹션 "R 파라미터"



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 언어 종속도

언어 종속 텍스트는 다음 용도로 사용됩니다:

- 소프트 키 라벨
- 헤더
- 도움말 텍스트
- 기타 텍스트

언어 종속 텍스트는 텍스트 파일로 저장됩니다.

참고

이러한 텍스트 파일을 사용할 때는 아래 단계를 수행해야 합니다:

- 텍스트 파일을 필요한 언어에서 사용할 수 있도록 활성화합니다.
- 그런 다음 파일을 제어 시스템의 언어 디렉토리로 전송합니다.

3.6 XML 진단

시스템은 스크립트 오류의 진단과 검색을 위해 Easy XML 진단 기능을 제공합니다.

진단 기능을 사용해 XML 구문을 확인하고 프로젝트에 속한 모든 스크립트를 실행할 수 있습니다. 이를 위해 기능, 메뉴, 양식과 변수의 존재 및 유효성도 확인합니다. 오류는 목록으로 표시됩니다.

Easy XML 진단 기능은 **DialogGui** 태그의 속성이나 디스플레이 머신 데이터를 사용해 활성화할 수 있습니다.

Easy XML 진단 활성화

예:

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

또는

화면 표시 머신 데이터:

MD9113 $MM_EASY_XML_DIAGNOSE	진단 및 수정은 Easy XML 스크립트를 지원합니다.
=0	진단 활성화 안 됨
=1	구문 확인 활성화
= 0x100	메시지는 error_log.txt 파일에 추가적으로 저장됩니다.

트레이스 출력을 사용한 진단

debug 태그를 사용해 트레이스 출력을 스크립트에 통합할 수 있습니다. **DialogGui** 태그에 값이 **on**인 **debug_msg** 속성을 지정한 경우에만 데이터가 저장됩니다.

소프트 키 기능 개요

대화 상자	소프트 키	기능
메인 메뉴 "EasyXML 진단"	스크립트 시작	로드된 프로그램이 바로 시작됩니다.
	확인 시작	구문 확인이 시작됩니다. 모든 발생한 메시지를 확인할 수 있습니다. 확인 갱신이 가능합니다.
"오류" 및 "진단" 대화 상자	오류	오류와 경고에 따라 결과가 표시되고 정렬됩니다.
	경고	
	오류로 이동	오류나 경고가 발견되면 소프트 키가 표시됩니다. 소프트 키를 누르면 오류 목록에서 표시된 XML 파일이 열려 편집할 수 있습니다. 커서는 자동으로 잘못된 행에 위치합니다.
	결과 확인	모든 발생한 메시지를 확인할 수 있습니다.
"문서" 대화 상자	저장	XML 파일의 변경 사항이 저장됩니다.
	취소	XML 파일이 변경되지 않고 닫힙니다. 확인 결과가 다시 표시됩니다.

3.7 XML 식별자

3.7.1 일반 구조

대화 상자 환경 설정을 위한 스크립트 파일의 구조 및 명령

모든 대화 상자 환경 설정은 **DialogGui** 태그에 저장되어야 합니다.

```
<DialogGui>
…
</DialogGui>
```

예제:

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

</DialogGui>
```

명령

언어는 조건 명령 및 루프 제어를 실행하기 위해 다음과 같은 명령을 제공합니다:

- For 루프
- WHILE 루프
- Do while 루프
- 조건부 처리
- 스위치 및 케이스 명령
- 대화 상자 양식의 화면 제어기
- 소프트 키 설명
- 변수 정의

방법에 대한 자세한 설명은 "명령/식별자 설명 (페이지 46)" 섹션을 참조하십시오.

3.7.2 명령/식별자 설명

아래 **XML tags**는 대화 상자 및 메뉴 생성을 위해 그리고 프로그램 순차 실행을 위해 정의됩니다.

참고

큰따옴표 안의 속성 값 "<...>"은 현재 사용된 수식으로 대체해야 합니다.

예:

<DATA_LIST action="read/write/append" id="<list name>">
은 다음과 같이 프로그램합니다.

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

여러 줄의 XML 태그와 예를 복사하는 경우 원하지 않는 줄 바꿈으로 마크업이 분리되지 않도록 주의하십시오.

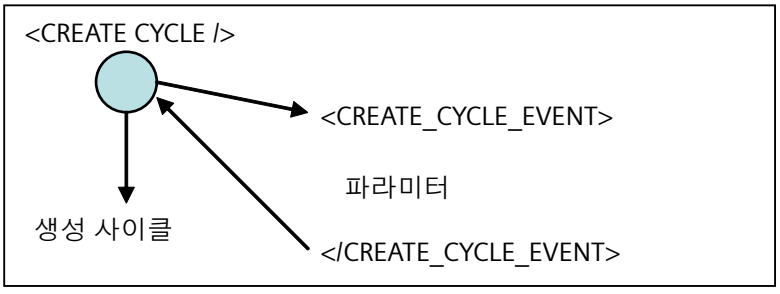
태그 이름	의미
BREAK	루프 조건부 취소.

CONDITION	이 태그는 한 줄에 프로그램되어야 합니다. 여러 줄로 표기하는 경우 태그 이름과 별도로 조건을 지정해야 합니다.
	
	예:
	`<if>`
	`<condition>test <= 2</condition>`
	`...`
	
	또는
	
	`<if>`
	`<condition>`
	`test <= 2`
	`</condition>`
	`...`

CONTROL	이 태그는 제어 요소 생성에 사용합니다. 설명은 "소프트 키 메뉴 및 대화 상자 양식 생성하기 (페이지 86)" 장을 참조하십시오.

태그 이름	의미
CONTROL_RESET	이 태그는 하나 이상의 제어 구성요소를 다시 시작할 수 있습니다. **구문:** `<CONTROL_RESET resetnc="TRUE" />` **속성:** • **RESETNC = "TRUE"** NC 콤포넌트가 재시작됩니다. • **RESETDRIVE = "TRUE"** 드라이브 콤포넌트가 재시작됩니다.

태그 이름	의미

CREATE_CYCLE_EVENT	파서가 CREATE_CYCLE 태그를 처리하기 시작하면 처음에 <CREATE_CYCLE_EVENT> 메시지가 활성 양식에 전송됩니다. 파서가 파라미터 목록 및 생성 규칙에 따라 NC 작업을 생성하기 전에 이 메시지를 이용해 싸이클 파라미터를 준비할 수 있습니다.
	 
	
	구문:
	`<CREATE_CYCLE_EVENT>`
	…
	…
	`</CREATE_CYCLE_EVENT>`
	예:
	`<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>`

태그 이름	의미
DATA	태그를 통해 NC, PLC, GUD 및 드라이브 데이터에 작성할 수 있습니다. "콤포넌트 주소 지정 (페이지 159)" 장에 주소 구성 방법이 자세히 설명되어 있습니다. **속성:** • **name** 변수 주소 **태그 값:** 태그 값으로 상수, **변수 또는 항**이 필요합니다. **구문:** <DATA name="<variable name>"> value </DATA> <DATA name="<variable name>"> <term> </DATA> **예:** <DATA name = "plc/mb170"> 1 </DATA> … <LET name = "tempVar"> 7 </LET> <!– 로컬 변수 "tempVar"의 내용을 비트 메모리 바이트 170에 씁니다. → <DATA name = "plc/mb170">$tempVar</DATA>
DATA 계속	**예:** 항의 계산. 결과는 지정된 변수에 할당됩니다. <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>

태그 이름	의미

DATA_LIST	태그는 목록에 나열된 드라이브 및 머신 데이터를 저장하거나 복원할 수 있게 합니다.
	주소는 한 줄로 나열됩니다. "콤포넌트 주소 지정 (페이지 159)" 장에 주소 구성 방법이 자세히 설명되어 있습니다.
	최대 20개의 임시 데이터 목록을 생성할 수 있습니다.
	
	속성:
	• **action**
	read – 나열된 변수의 값이 임시 메모리에 저장됩니다.
	append – 나열된 변수의 값이 기존 목록에 추가됩니다.
	write – 백업한 값을 해당 머신 데이터에 복사합니다.
	• **id**
	임시 메모리 식별을 위한 식별자
	구문:
	`<DATA_LIST action="<read/write/append>"  id="<list name>">`
	`NC/PLC 주소 컴파일`
	`</DATA_LIST>`
	
	예:
	`<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	이 태그는 Easy XML 함수의 루트 태그를 나타냅니다. 포함된 모든 명령어는 Easy XML 파서를 사용하여 편집됩니다. **속성:** 중앙 함수를 활성화하기 위해 다음 속성을 선택적으로 지정할 수 있습니다. • **diagnose** - **true** 값은 XML 진단을 활성화함 • **debug_msg** - "on" 값은 트레이스 메시지를 저장함 **자세한 정보**는 "XML 진단 (페이지 43)" 장을 참조하십시오. • **textfile** - 사용될 텍스트 파일의 이름 • **textcontext** - 사용될 텍스트 컨텍스트는 속성 텍스트 파일과 함께만 유효함 • **textfilelist** - 서로 다른 텍스트 파일이 있는 목록의 로드를 허용함 **자세한 정보**는 "프로젝트별 텍스트 파일 (페이지 290)" 장을 참조하십시오.

태그 이름	의미
DIALOGGUI 계속	• **restoresession** - **true** 값 Easy XML 프로젝트를 다시 선택하면 파서가 마지막으로 선택한 메뉴와 마지막으로 선택한 양식을 엽니다. 이 동작은 HMI가 다시 시작될 때까지 유지됩니다. **자세한 정보**는 "세션 복원 (페이지 298)" 장을 참조하십시오.
DEBUG_MSG	이 속성은 트레이스 출력의 저장을 제어합니다. 자세한 설명은 "소프트 키 메뉴 및 대화 상자 양식 생성하기 (페이지 86)" 장에서 DEBUG 태그 아래에 나오는 내용을 참조하십시오. **on** 값이 할당되면 파서가 모든 출력을 한 파일에 저장합니다. 그 결과 스크립트 실행이 느려질 수 있습니다. 기본적으로 **off** 값이 설정됩니다.

DO_WHILE	Do while 루프
	`DO`
	`  명령`
	`WHILE (Test)`
	구문:
	`<DO_WHILE>`
	명령
	…
	`<CONDITION>…</CONDITION>`
	`</DO_WHILE>`
	Do while 루프는 명령 블록 및 조건으로 구성됩니다. 먼저 명령 블록의 코드가 실행된 다음 조건이 적용됩니다. 조건이 참일 경우 함수는 코드 섹션을 다시 실행합니다. 이것은 조건이 거짓이 될 때까지 계속 반복됩니다.
	예:
	`<DO_WHILE>`
	`<DATA name = "PLC/qb11"> 15 </DATA>`
	`<CONDITIION> "plc/ib9" == 0 </CONDITION>`
	`</DO_WHILE>`

태그 이름	의미
DYNAMIC_INCLUDE	이 태그에 XML 스크립트 파일이 포함되어 있습니다. INCLUDE 태그와 달리 해당 작업을 실행하면 리드인만 먼저 실행됩니다. 대규모 프로젝트의 경우 이 태그를 사용하면 고객 영역 및/또는 싸이클 지원 로드 시간을 줄일 수 있습니다. 또한 세션 중에 항상 모든 대화 상자를 호출할 필요가 없기 때문에 평균적으로 필요한 리소스도 더 적습니다. **구문:** <DYNAMIC_INCLUDE src="path name"/> **예:** <SOFTKEY POSITION="3"> <CAPTION>MY_MENU</CAPTION> <DYNAMIC_INCLUDE src="f:\appl\sk3_script.xml"/> <NAVIGATION>MY_MENU</NAVIGATION> </SOFTKEY>
ELSE	조건에 부합하지 않는 상황에 대한 명령 (**IF, THEN, ELSE**)

태그 이름	의미

FOR	For 루프
	`for (초기화, 테스트, 계속) 명령`
	
	구문:
	`<FOR>`
	`<INIT>…</INIT>`
	`<CONDITION>…</CONDITION>`
	`<INCREMENT>…</INCREMENT>`
	명령
	…
	`</FOR>`
	For 루프는 다음과 같이 실행됩니다:
	1. 표현식 **초기화** (INIT)를 분석합니다.
	2. 표현식 **테스트** (CONDITION)를 부울 식으로 분석합니다.
	값이 거짓이면 For 루프가 종료됩니다.
	3. 후속 명령을 실행합니다.
	4. 표현식 **계속** (INCREMENT)을 분석합니다.
	5. 2를 계속합니다.
	INIT, CONDITION 및 INCREMENT 분기 내의 모든 변수는 FOR 루프 밖에서 선언되고 초기화됩니다.
	
	예:
	`<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	태그는 사용자 대화 상자에 대한 설명을 포함합니다. 설명은 "소프트 키 메뉴 및 대화 상자 양식 생성하기 (페이지 86)" 장을 참조하십시오.
HMI_RESET	HMI 재시작을 트리거하는 태그입니다. 이 명령 후 해석이 중단됩니다.

태그 이름	의미

IF	조건문 (**IF, THEN, ELSE**)
	THEN 및 ELSE 태그는 IF 태그 안에 포함시킵니다.
	IF 태그 다음의 CONDITION 태그에 실행 조건을 지정합니다. 명령 이후 처리 공정은 조작 결과에 따라 달라집니다. 실행 결과가 참이면 THEN 분기가 실행되고 ELSE 분기는 건너뜁니다. 실행 결과가 거짓이면 파서는 ELSE 분기를 실행합니다.
	
	구문:
	`<IF>`
	`<CONDITION> 조건 != 7 </CONDITION>`
	`<THEN>`
	`조건을 충족하는 경우 명령`
	`</THEN>`
	`<ELSE>`
	`조건을 충족하지 못하는 경우 명령`
	`</ELSE>`
	`</IF>`
	
	예:
	`<IF>`
	`<CONDITION> "plc/mb170" != 7 </CONDITION>`
	`<THEN>`
	`<OP> "plc/mb170" = 7 </OP>`
	`…`
	`</THEN>`
	`<ELSE>`
	`<OP> "plc/mb170" = 2 </OP>`
	`…`
	`</ELSE>`
	`</IF>`

태그 이름	의미

IF 계속	조건에서 로컬 변수만 사용하는 경우 조건을 IF 태그에서 속성으로 지정할 수 있습니다.
	
	예:
	`<let name="var1">1</let>`
	`<if condition="var == 1">`
	`<then>`
	`...`
	`...`
	`</then>`
	`</if>`
	
	참고:
	참조 변수와 결합되지 않은 컨트롤은 정수 데이터 형식으로 할당됩니다.
	예외:
	imagebox 및 **multiline** 유형 컨트롤은 **string** 데이터 형식으로 할당됩니다.

INCLUDE	명령이 XML 설명을 포함합니다. (이 표의 DYNAMIC_INCLUDE 참조) **속성:** • **src** 경로 이름이 포함되어 있습니다. **구문:** `<?INCLUDE src="<Path name>" ?>`

태그 이름	의미
LET	명령은 규정된 이름 아래에 로컬 변수를 생성합니다. **필드:** **dim** (차원) 속성을 이용해 1차원 또는 2차원 필드를 생성할 수 있습니다. 필드 인덱스는 개별 필드 요소의 주소를 지정합니다. 2차원 필드의 경우 행 인덱스를 먼저 지정한 후 열 인덱스를 지정합니다. • 1차원 필드: 인덱스 0~4 0 1 2 3 4 • 2차원 필드: 행 인덱스 0~3, 열 인덱스 0~5 0,0 0,1 0,2 0,3 0,4 0,5 1,0 1,1 1,2 1,3 1,4 1,5 2,0 2,1 2,2 2,3 2,4 2,5 3,0 3,1 3,2 3,3 3,4 3,5 **속성:** • **name** 변수 이름 • **type** 변수 유형은 정수(INT), 부호 없는 정수(UINT), 실수(double)(DOUBLE), 실수(float)(FLOAT), 문자열(STRING) 또는 STRUCT 일 수 있습니다. **typedef** 에 의해 생성된 구조를 변수 유형으로 사용할 수도 있습니다(태그 식별자 TYPEDEF 참조) 명령의 유형이 규정되어 있지 않을 경우 시스템은 정수 변수를 생성합니다. `<LET name = "VAR1" type = "INT" />` • **permanent** 속성을 **true** 로 설정하면 변수 값이 영구 저장됩니다. 이 속성은 전역 변수에만 적용됩니다. • **poweroff** 속성 값이 **true**이면 제어 시스템이 꺼질 때까지 값이 유지됩니다. **자세한 정보**는 "세션 복원 (페이지 298)" 장을 참조하십시오. • **dim** 필드 요소의 개수는 다음과 같이 지정해야 합니다. 2차원 필드의 경우 두 차원을 구분하기 위해 첫 번째 차원을 지정한 후 쉼표를 넣고 두 번째 차원을 지정합니다.

태그 이름	의미
	필드 요소는 변수 이름 다음 꺾쇠 괄호 ([]) 안에 지정된 필드 인덱스를 통해 액세스합니다. name[index] 또는 name[row,column] – 1차원 필드: dim="<요소 개수>" – 2차원 필드: dim="<행 개수>,<열 개수>" 초기화되지 않은 필드 요소에는 "0"이 사전 지정되어 있습니다.

태그 이름	의미

LET 계속	**예:**
	1차원 필드:
	`<let name="array" dim="10"></let>`
	2차원 필드:
	`<let name="list_string" dim="10,3" type="string"></let>`
	
	사전 할당:
	변수는 하나의 값으로 초기화될 수 있습니다.
	`<LET name = "VAR1" type = "INT"> 10 </LET>`
	
	NC 또는 PLC 변수를 구성하는 값이 로컬 변수에 저장될 경우 할당 연산은 자동으로 포맷을 로드된 변수의 포맷에 맞춥니다.
	• 문자열 변수의 사전 할당:
	한 줄 이상의 텍스트는 포맷된 텍스트가 하나의 값으로서 전달될 경우 문자열 변수로 할당될 수 있습니다. 행이 line feed <LF> 로 끝나려면 행 끝에 문자 "\\n" 를 추가해야 합니다.

태그 이름	의미
	<LET name = "text" type = "string"> F4000 G94\\n G1 X20\\n Z50\\n M2\\n </LET>> **필드 (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> **할당:** 지정 연산자 "="를 사용해 머신 데이터 또는 서브루틴으로 구성된 값을 변수에 지정할 수 있습니다. 변수는 상위 XML 블록의 끝에 도달할 때까지 유효합니다. 전역 변수는 **DialogGUi** 태그 다음에 직접 생성해야 합니다. 대화 상자에는 다음 규칙이 적용됩니다. • 메시지 처리를 시작하면 해당 태그가 열립니다. • 메시지 실행이 완료되면 태그가 닫힙니다. • 태그가 닫히면 태그 내의 모든 변수가 삭제됩니다.

태그 이름	의미

LET 계속	**변수 유형 struct:**
	이 변수 유형은 구조 이름을 사용하여 주소를 지정할 수 있는 변수 구성을 포함합니다. 구조는 모든 변수 유형 및 구조를 포함할 수 있습니다.
	구조 내에서 "element" 태그와 함께 변수를 선언합니다. 태그의 속성 및 초기 설정은 let 명령의 속성 및 초기 설정에 해당합니다.
	
	`<let name="name" type="struct">`
	`<element name="<Name>" type="<Variable type>" />`
	…
	…
	…
	`</let>`
	
	구조체 요소에는 변수를 생성할 때 초기 값으로 사용되는 기본값을 할당할 수 있습니다.
	자세한 정보는 "구조체 초기화" 섹션을 참조하십시오.
	변수를 생성할 때 초기 값을 지정하면 이 값을 덮어씁니다.

태그 이름	의미

LET 계속	구조의 변수에 대한 액세스는 구조 이름과 변수 이름을 이용해 수행합니다. 두 이름은 점을 사용해 구분합니다.
	
	구문:
	
	`<op>`
	`Structure_name.variable_name = value;`
	`</op>`
	
	예:
	`<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>`

태그 이름	의미

LET 계속	**구조체 초기화:**
	각 구조체 요소의 초기 값을 지정해 변수를 생성할 때 구조체를 초기화할 수 있습니다. 구조체 배열에서 각 구조체를 중괄호를 사용해 서로 구분해야 합니다.
	
	예:
	`<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" >`
	rect 변수를 생성하면 구조체 요소가 다음 값으로 초기화됩니다.
	`rect.left= 22`
	`rect.top = 122`
	`rect.right = 220`
	`rect.bottom = 56`
	
	구조체에 초기 값을 할당하면 기본값을 덮어씁니다.
	`<let name="rect" type="StructRect" >`
	`  {11,12}`
	`</let>`
	rect 변수를 생성하면 구조체 요소가 다음 값으로 초기화됩니다.
	`rect.left= 11`
	`rect.top = 12`
	`rect.right = 220`
	`rect.bottom = 56`

태그 이름	의미

LET 계속	**중첩 구조체:**
	구조체는 사용자가 정의한 변수 유형의 변수이기 때문에 구조체에서 구조체 요소로 사용될 수 있습니다.
	구조체 요소는 설명된 규칙에 따라 초기화됩니다.
	
	예:
	`<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" >`
	rect_rect_array 변수를 생성하면 구조체 요소가 다음 값으로 초기화됩니다.
	`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>`
	
	rect_rect_array 변수를 생성하면 구조체 요소가 다음 값으로 초기화됩니다.
	`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 계속	**전역 문자열 변수의 초기화:** 텍스트 식별자를 사용해 전역 문자열 변수를 초기화할 경우 표준 텍스트 파일 **oem_xml_screens_xxx.ts**나 **DialogGUI** 태그에 지정된 텍스트 파일에 대체될 식별자와 텍스트가 있어야 합니다. 애플리케이션을 로드할 때 활성 언어의 텍스트가 텍스트 식별자를 대체합니다. 애플리케이션이 활성화된 동안 언어 변경이 수행되면 전역 문자열 변수의 새로운 초기화가 수행되지 않습니다. **LANGUAGE_CHANGED** 메시지에서 텍스트를 교환할 수 있습니다.

태그 이름	의미
LOCK_OPERATING_AREA	조작 영역 전환이 잠겼습니다. 조작 영역 전환 잠금은 **UNLOCK_OPERATING_AREA** 태그로 취소합니다. **구문:** `<LOCK_OPERATING_AREA />`
MSG	연산자 콤포넌트는 태그에 표시되는 메시지를 보여줍니다. 알람 번호가 사용될 경우 대화 상자는 그 번호에 저장되어 있는 텍스트를 표시합니다. **예:** `<MSG text ="my message" />`

MSGBOX	명령은 해당 반환 값이 분기 작업에 사용될 수 있는 메시지 박스를 엽니다.
	
	구문:
	`<MSGBOX text="<Message>" caption="<caption>" retvalue="<variable name>" type="<button type>" />`
	
	속성:
	• **text**
	텍스트
	• **caption**
	헤더
	• **retvalue**
	반환 값이 복사되는 변수의 이름:
	1 – OK
	0 – CANCEL
	• **type**
	확인 옵션:
	"BTN_OK"
	"BTN_CANCEL"
	"BTN_OKCANCEL"
	"text" 또는 **"caption"** 속성에 알람 번호를 사용한 경우 메시지 상자는 이 번호에 저장된 텍스트를 표시합니다.
	
	예:
	`<MSGBOX text="텍스트 메시지" caption="정보" retvalue="result" type="BTN_OK" />`

태그 이름	의미

OP	태그가 지정된 작업을 실행합니다.
	"연산자 (페이지 84)" 장에서 설명한 작업을 실행할 수 있습니다.
	NC, PLC 및 드라이브 데이터에 액세스하려면 완전한 변수 이름이 **인용 부호** 안에 위치해야 합니다. "콤포넌트 주소 지정" (페이지 159) 장에 주소 구성 방법이 자세히 설명되어 있습니다.
	`PLC: "PLC/MB170"`
	`NC: "NC/Channel/…"`
	
	예:
	`<LET name = "tmpVar" type="INT"> </LET>`
	`<OP> tmpVar = "plc/mb170" </OP>`
	`<OP> tmpVar = tmpVar *2 </OP>`
	`<OP> "plc/mb170" = tmpVar </OP>`
	
	연산 태그 내에 하나 이상의 등식을 사용할 수 있습니다. 세미콜론은 명령의 끝을 표시합니다.
	
	예:
	`<op>`
	`x = x+1;`
	`y = y+1;`
	`</op>`
	
	문자열 처리:
	연산 명령은 문자열을 처리하여 그 결과를 등식에 규정된 문자열 변수에 할당할 수 있습니다.
	식별자 _T는 텍스트 식 식별 수단으로서 시작 지점에 위치해야 합니다. 변수 값의 포맷도 가능합니다. 이름 _F는 포맷 규정 시작 점에 위치해야 하고 그 뒤에 포맷 명령이 따라옵니다. 그런 다음 변수에 대한 주소가 규정됩니다.
	
	예:
	`<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>`

태그 이름	의미

OPERATION	**연산**
	명령은 등식 내에서 이동할 수 있습니다.
	좌로 이동 "<<" 연산자
	<< 기능은 비트를 왼쪽으로 이동합니다. 값 및 이동 증분의 수를 직접 지정하거나 변수로 지정할 수 있습니다. 데이터 형식의 한계에 도달하면 비트는 에러 메시지를 표시하지 않고 한계를 벗어나 이동합니다.
	실행 규칙
	왼쪽에서 오른쪽으로 실행. '+' 및 '-'는 '<<' 및 '>>'보다 우선합니다.
	예:
	`<op> idx = idx << 2 </op>`
	`<op> idx = 3 + idx << 2 </op>`
	우로 이동 ">>" 연산자
	>> 기능은 비트를 오른쪽으로 이동합니다. 값 및 이동 증분의 수를 직접 지정하거나 변수로 지정할 수 있습니다. 데이터 형식의 한계에 도달하면 비트는 에러 메시지를 표시하지 않고 한계를 벗어나 이동합니다.
	실행 규칙
	왼쪽에서 오른쪽으로 실행. '+' 및 '-'는 '<<' 및 '>>'보다 우선합니다.
	예:
	`<op> idx = idx >> 2 </op>`
	`<op> idx = 3+idx >> 2</op>`

태그 이름	의미

PASSWORD	이 태그는 "EasyExtend" 함수를 위해 추가 유닛의 잠금을 해제하는 데 사용됩니다.
	명시된 문자열은 지정된 참조 변수에 기록되며 PLC 사용자 프로그램에서 반드시 처리되어야 합니다.
	
	구문:
	`<PASSWORD refVar ="<variable name>" />`
	
	속성:
	• **refVar**
	기준 변수의 이름
	• **text**
	속성을 지정하면 기본 텍스트를 대체합니다(옵션).
	
	예:
	`<PASSWORD refvar="plc/mw107" />`
	
	예 옵션:
	`<password refVar = "plc/MD108" text="Password" />`

POWER_OFF	이 메시지는 작업자에게 기계의 전원 차단을 요구하는 프롬프트를 제시합니다. 메시지 텍스트는 시스템에 영구 저장됩니다.

태그 이름	의미

PRINT	태그는 대화 상자 라인에 텍스트를 출력하거나 텍스트를 규정된 변수로 복사합니다.
	텍스트가 포맷 식별자를 가지고 있을 경우 변수 값이 적절한 장소에 삽입됩니다.
	
	구문:
	`<PRINT name="변수 이름" text="text %포맷 정의"> Variable, … </PRINT>`
	`<PRINT text="text %포맷"> Variable, … </PRINT>`
	속성:
	• **name**
	텍스트를 저장할 변수의 이름 (옵션)
	• **text**
	텍스트
	
	포맷:
	문자 "%"는 값으로 규정된 변수를 포맷합니다.
	%[Flags] [Width] [.decimal places] 유형
	• 플래그:
	출력 포맷 정의를 위한 옵션 문자:
	– 오른쪽 맞춤 또는 왼쪽 맞춤(왼쪽 맞춤의 경우 "-")
	– 리딩 제로 ("0") 추가
	– 빈칸 채우기
	• 너비:
	인수는 음수가 아닌 수에 대한 최소 출력 너비를 정의합니다. 출력되어야 하는 값이 정의된 인수보다 자리 수가 적을 경우 없는 공간은 빈칸으로 채웁니다.
	• 소수 자리:
	부동 소수점일 경우 옵션 파라미터는 소수 자리의 수를 정의합니다.
	• 유형:
	유형은 인쇄 명령을 위해 전송되는 데이터 포맷을 정의합니다. 이 문자열은 규정될 필요가 있습니다.
	– d: 정수 값
	– f: 부동 소수점 실수
	– s: 문자열

태그 이름	의미

PRINT 계속	**값:**
	값이 텍스트에 삽입되어야 하는 변수 번호.
	변수 유형은 포맷 명령의 유형 식별자와 동일해야 하고 쉼표를 사용해 분리시켜야 합니다.
	
	예:
	정보 라인에 텍스트 출력
	`<PRINT text="정보 텍스트" />`
	변수 포맷 텍스트 출력
	`<LET name="trun_dir"></LET>`
	`<PRINT text="M%d">trun_dir</PRINT>`
	변수 포맷 문자열 변수에 텍스트 출력
	`<LET name="trun_dir"></LET>`
	`<LET name="str" type="string" ></LET>`
	`<print name="str" text="M%d ">trun_dir</print>`

PROGRESS_BAR	이 태그는 진행 상태 표시줄을 열거나 닫습니다. 진행 상태 표시줄은 애플리케이션 창 아래 표시됩니다.
	
	구문:
	`<PROGRESS_BAR type="<true/false>"> value </ PROGRESS_BAR>`
	
	속성:
	• **type =** "TRUE" - 진행 상태 표시줄을 엽니다.
	• **type** = "FALSE" - 진행 상태 표시줄을 닫습니다.
	• **min**
	(옵션) - 최소값
	• **max**
	(옵션) - 최대값
	
	값:
	• **Value**
	표시줄의 작업 진행률 (%) 위치
	예:
	`<PROGRESS_BAR type="true" min="0" max="101" >20< /`
	`PROGRESS_BAR>......<PROGRESS_BAR >50< /`
	`PROGRESS_BAR>......<PROGRESS_BAR type="false" >100< /PROGRESS_BAR>`

태그 이름	의미

SEND_MESSAGE	이 태그가 두 개의 파라미터와 함께 메시지를 활성 양식으로 전송하고, 전송된 메시지는 태그 메시지 내에서 처리됩니다.
	
	구문:
	`<SEND_MESSAGE>p1, p2</SEND_MESSAGE>`
	
	예:
	`<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>`

태그 이름	의미

SHOW_CONTROL	이 태그를 사용해 컨트롤 표시 여부를 제어할 수 있습니다.
	
	구문:
	`<SHOW_CONTROL name="<name>" type="<type>" />`
	
	속성:
	• **name**
	컨트롤의 이름
	• **type =** "TRUE" - 컨트롤을 표시합니다.
	• **type** = "FALSE" - 컨트롤을 표시하지 않습니다(숨김).
	
	예:
	`<SHOW_CONTROL name="myEditfield" type="false" />`
	…
	…
	`<SHOW_CONTROL name="myEditfield" type="true" />`

SLEEP	이 태그는 지정된 기간 동안 스크립트 실행을 인터럽트합니다. 인터럽트 시간은 전송된 값에 기본 시간 50 ms를 곱한 시간입니다.
	
	구문:
	`<SLEEP value="인터럽트 시간" />`
	
	예:
	대기 시간, 1.5초
	`<SLEEP value="30" />`

STOP	이 지점에서 해석이 취소됩니다.

태그 이름	의미

SWITCH	**SWITCH** 명령은 여러 가지 선택이 가능하다는 뜻입니다. 용어는 일단 평가된 다음 수 많은 상수와 비교됩니다. 수식이 상수와 일치하면 **CASE** 명령에 지정된 명령이 실행됩니다.
	수식과 일치하는 상수가 없으면 **DEFAULT** 명령이 실행됩니다.
	
	구문:
	`<SWITCH>`
	`<CONDITION> 값 </CONDITION>`
	`<CASE value="<상수 1>">`
	명령
	…
	`</CASE>`
	`<CASE value="<상수 2>">`
	명령
	…
	`</CASE>`
	`<DEFAULT>`
	명령
	…
	`</DEFAULT>`
	`</SWITCH>`

태그 이름	의미

SWITCHTOAREA	SWITCHTOAREA 태그는 고객 영역에서 지정된 조작 영역으로 변경됩니다.
	파라미터는 속성 값으로 지정됩니다.
	
	구문:
	`<switchToArea name="area" args="argument "/>`
	
	속성:
	name
	다음 이름은 조작 영역에서 선언됩니다.
	• **AreaMachine** - 기계
	• **AreaParameter**-파라메타
	• **AreaProgramEdit** - 편집기
	• **AreaProgramManager** - 프로그램 관리자
	• **AreaDiagnosis** - 진단
	• **AreaStartup** - 설정
	args (예비)
	대화 상자 활성화를 위해 런타임 인수가 조작 영역으로 전달될 수 있습니다.
	
	예:
	`<switchToArea name="AreaMachine" />`

SWITCHTODYNAMICTARGET	이전 대화 상자가 동적 점프 대상으로 정의되면, SWITCHTODYNAMICTARGET 태그가 이 대화 상자를 활성화하고 스크립트 실행을 종료합니다. **구문:** `<switchToDynamicTarget />`
THEN	조건을 충족하는 경우 명령(**IF, THEN, ELSE**)

태그 이름	의미
TYPE_CAST	로컬 변수의 데이터 유형을 변환하는 태그입니다. **구문:** <type_cast name="variable name" type=" new type" /> **속성:** • **name** 변수 이름 • **type** 새 데이터 유형이 변수에 지정됩니다. • **convert** 새 데이터 유형이 변수에 지정됩니다. 변수 값도 새 데이터 유형으로 변환됩니다.

태그 이름	의미

TYPEDEF	데이터 유형의 새 식별자는 이 태그로 정의할 수 있습니다. 이렇게 하면 구조 정의의 경우 데이터 유형을 한 번만 정의하고 LET 명령에서 데이터 유형으로 사용할 수 있다는 장점이 있습니다.
	식별자와 유형을 속성으로 제공해야 합니다.
	파서는 구조 정의 사양만을 지원합니다.
	
	유형 정의에서 "element" 태그로 변수를 선언합니다. 태그의 속성은 let 명령의 속성에 해당합니다.
	
	`<typedef name="<identifier>" type="struct">`
	` <element name="<name>" type="<variable type>" />`
	…
	…
	…
	`</typedef>`
	
	정의 후 식별자를 LET 명령에서 데이터 유형으로 사용할 수 있습니다.
	
	`<let name="<variable name>" type="<identifier>"></let>`
	
	Example:
	`<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>`

태그 이름	의미
TYPEDEF 계속	일부 사전 정의된 함수는 구조체 유형 RECT, POINT 또는 SIZE 의 변수를 호출 파라미터로 예상합니다. 이 구조체는 struct_def.xml 파일에서 정의됩니다. **자세한 정보**는 "struct_def.xml 파일 생성 (페이지 157)" 장을 참조하십시오. **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	조작 영역 전환 잠금 취소 **구문:** `<UNLOCK_OPERATING_AREA />`

태그 이름	의미

WAITING	이 태그는 NC 또는 드라이브 리셋 후 구성요소가 핫 리스타트 (hot restart) 할 때까지 기다립니다.
	
	속성:
	• **WAITINGFORNC** = "TRUE" - NC가 재시작될 때까지 시스템이 대기합니다.
	• **WAITINGFORDRIVE** = "TRUE" - 드라이브가 재시작될 때까지 시스템이 대기합니다.
	
	구문:
	`<WAITING WAITINGFORNC = "TRUE"/>`
	
	예:
	...
	`<CONTROL_RESET resetnc = "true" resetdrive = "true"/>`
	`<WAITING waitingfornc = "true" waitingfordrive = "true" />`
	...

WHILE	WHILE 루프
	`WHILE (Test)`
	`  명령`
	
	구문:
	`<WHILE>`
	`<CONDITION>…</CONDITION>`
	명령
	…
	`</WHILE>`
	While 루프는 조건에 부합할 때 반복적으로 순차 명령을 실행합니다. 이 조건은 명령 순차가 실행되기 전에 테스트됩니다.
	
	예:
	`<WHILE>`
	`<CONDITION> "plc/ib9" == 0 </CONDITION>`
	`<DATA name = "PLC/qb11"> 15 </DATA>`
	`</WHILE>`

태그 이름	의미
XML_PARSER	"XML_PARSER" 태그는 XML 파일의 구문 분석에 사용할 수 있습니다. 파서는 XML 파일을 해석하여 정의된 콜백 함수를 호출합니다. 각 콜백 함수는 사전 정의된 이벤트에 속합니다. 프로그래머는 이 함수 내의 XML 데이터를 처리할 수 있습니다. **사전 정의된 이벤트:** • 시작 문서 파서가 문서를 열고 구문 분석을 시작합니다. • 끝 문서 파서가 문서를 닫습니다. • 시작 요소 파서가 요소를 발견하여 모든 속성 및 속성 값을 포함하는 목록을 생성합니다. 이 목록은 콜백 함수로 전달됩니다. • 끝 요소 요소의 끝을 발견했습니다. • 문자 파서가 요소의 모든 문자를 전달합니다. • 에러 파서가 구문 에러를 발견했습니다. 이벤트가 발생하면 파서는 콜백 함수를 호출하고 함수 반환값을 확인합니다. 함수가 "true" 값을 반환하면 파서가 프로세스를 계속 진행합니다.

태그 이름	의미

XML_PARSER 계속	**인터페이스**
	name 속성의 값은 XML 파일의 경로를 포함합니다.
	이벤트를 콜백 함수에 할당하려면 다음 속성을 지정해야 합니다.
	표준
	startElementHandler
	endElementHandler
	charactersHandler
	옵션
	errorHandler
	documentHandler
	속성의 값은 콜백 함수의 이름을 정의합니다.
	
	예:
	`<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>`

태그 이름	의미

XML_PARSER 계속	파서는 콜백 함수가 이벤트 데이터에 액세스할 수 있도록 변수도 제공합니다.
	
	startElementHandler:
	함수 파라미터
	tag_name - 태그 이름
	num - 발견된 속성의 개수
	시스템 변수
	$xmlAttribute num 가 요소 개수를 표시하는 문자열 배열입니다.
	$xmlValue 0-num 속성 값 범위를 포함하는 문자열 배열입니다.
	
	예:
	`<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:
	함수 파라미터
	tag_name - 태그 이름
	
	예:
	`<function_body name="endElementHandler" parameter="tag_name">`
	`<print text="name %s"> tag_name </print>`
	`</function_body>`

태그 이름	의미

XML_PARSER 계속	**charactersHandler:**
	시스템 변수
	$xmlCharacters 데이터를 포함하는 문자열
	$xmlCharactersStart 항상 0
	$xmlCharactersLength 바이트 수
	
	예:
	`<function_body name="charactersHandler" return="true" >`
	`<print text="chars %s"> $xmlCharacters </print>`
	`</function_body>`
	
	documentHandler:
	함수 파라미터
	state 1 시작 문서, 2 끝 문서
	
	errorHandler:
	시스템 변수
	$xmlErrorString 유효하지 않은 행을 포함합니다 (시스템 변수).
	
	예:
	`<function_body name="errorHandler" >`
	`<print text="error %s">$xmlErrorString</print>`
	`</function_body>`
	
	콜백 결과:
	$return 1 (참) 이면 파서가 파일의 구문 분석을 계속 진행합니다.

3.7.3 색 코딩

색 속성은 HTML 언어에 색 코딩 계통을 이용합니다.

구문에 따르면 색상 지정은 "#" (hash) 문자와 16진수 중 6개의 숫자로 구성되는데, 각각의 색은 2자리 숫자로 표시됩니다.

R – 적색

G - 녹색

B - 청색

#RRGGBB

예:

색 = "#ff0011"

예제 색상:

빨간색	녹색	파란색	노란색	흰색	검은색
#FF0000	#00FF00	#0000FF	#ffff00	#FFFFFF	#000000

3.7.4 특수 XML 구문

XML 구문에 특수 의미가 있는 특수 문자를 일반 XML 편집기로 정확하게 표시해야 할 경우 그 문자를 다시 써야 합니다.

다음 문자가 영향을 받습니다:

문자	XML 표시법
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;

3.7.5 HTML 특수 문자

문자의 표현이나 제어 문자의 사용을 위해 HTML 특수 문자 평가가 제공됩니다.

코드 페이지 Latin 1의 10진수 및 16진수 인코딩을 사용할 수 있습니다.

예

문자	설명	표기법 10진수	표기법 16진수
"	인용 부호	"	"
LF	줄 바꿈

3.7.6 연산자

연산 명령은 다음 연산자를 처리합니다.

연산자	의미
=	할당
==	같음
<, <	보다 작음
>, >	보다 큼
<=, <=	보다 작거나 같음
>=, >=	보다 크거나 같음
|	비트 단위의 OR 연산
||	논리 OR 연산
&, &	비트 단위의 AND 연산
&&, &&	논리 AND 연산
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈
!	Not
!=	같지 않음

연산 명령은 왼쪽부터 오른쪽으로 처리됩니다. 하위 용어 실행을 위한 우선 순위를 정의하기 위해 특정 환경 하에서 용어를 괄호에 넣을 수도 있습니다.

3.7.7 시스템 변수

시스템 변수는 모든 스크립트에서 사용할 수 있는 변수이고, 이들은 파서와 스크립트 실행 사이 데이터 교환에 사용됩니다.

다음 표는 태그에 대한 개요를 보여주는데, 이 태그를 위해 변수는 자동으로 생성됩니다.

변수 이름	의미	태그에서 유효함
$actionresult	파서가 이벤트를 처리해야 하는지에 관한, 파서쪽으로의 신호	KEY_EVENT
$focus_name	입력 초점을 갖는 필드 이름을 포함함	FOCUS_IN INDEX_CHANGED EDIT_CHANGED
$focus_item_data	필드에 할당된 항목 데이터의 숫자값을 포함함	
$gestureinfo	손가락 동작의 경우, 파서는 변수에서 동작 정보를 제공하고 gesture_event 태그를 실행합니다.	GESTURE_EVENT
$return	변수는 하위 함수의 반환값을 갖고 있음	FUNCTION_BODY
$message_par1 $message_par2	SEND_MESSAGE 함수의 호출 파라미터를 포함함	MESSAGE
$xmlAttribute	발견된 태그 속성 목록을 포함함	XML_PARSER startElementHandler
$xmlValue	발견된 태그값 목록을 포함함	
$xmlCharacters	데이터 흐름을 포함함	XML_PARSER charactersHandler
$xmlCharactersStart	시작 색인을 포함함:	
$xmlCharactersLength	데이터 흐름에 저장된 문자 개수를 포함함	
$mouse_event.type $mouse_event.x $mouse_event.y $mouse_event.id $mouse_event.buttons $mouse_event.button	마우스 이벤트의 파라미터를 전송하기 위한 구조	MOUSE_EVENT

3.7.8 소프트 키 메뉴 및 대화 상자 양식 생성하기

XML 설명에 이름이 "main"인 메뉴 태그가 있는 경우에만 대화 상자 양식을 통합할 수 있습니다. 이 태그는 <CUSTOM> 영역이 활성화된 후에 시스템에 의해 호출됩니다. 추가적인 대화 상자 양식을 분기할 수 있고 태그 내에 정의된 대화 상자를 활성화할 수 있습니다.

```
<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>
```

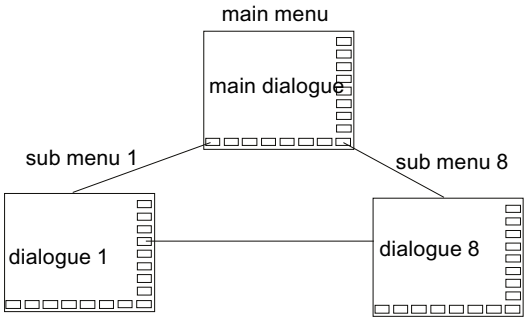


그림 3-6 메뉴 구조

태그 이름	의미
FORM	태그는 사용자 대화 상자에 대한 설명을 포함합니다. 관련 태그는 메뉴 생성 및 대화 상자 마스크에 대한 섹션에 설명되어 있습니다. **구문:** `<FORM name="<dialog name>" color="#ff0000">` **속성:** • **color** 대화 상자 마스크의 배경색 **자세한 정보**는 "색 코딩 (페이지 82)" 장을 참조하십시오. – 기본값은 하얀색 • **name** 양식 식별자 • **type** 허용되는 값은 *cycle* 입니다. 이 값이 사용자 싸이클 화면 양식을 지정합니다. • **xpos** 대화 상자 좌측 상단의 X 위치 (옵션) • **ypos** 대화 상자 좌측 상단의 Y 위치 (옵션) • **width** 픽셀 단위로 X 방향으로 확장 (옵션) • **height** 픽셀 단위로 Y 방향으로 확장 (옵션)

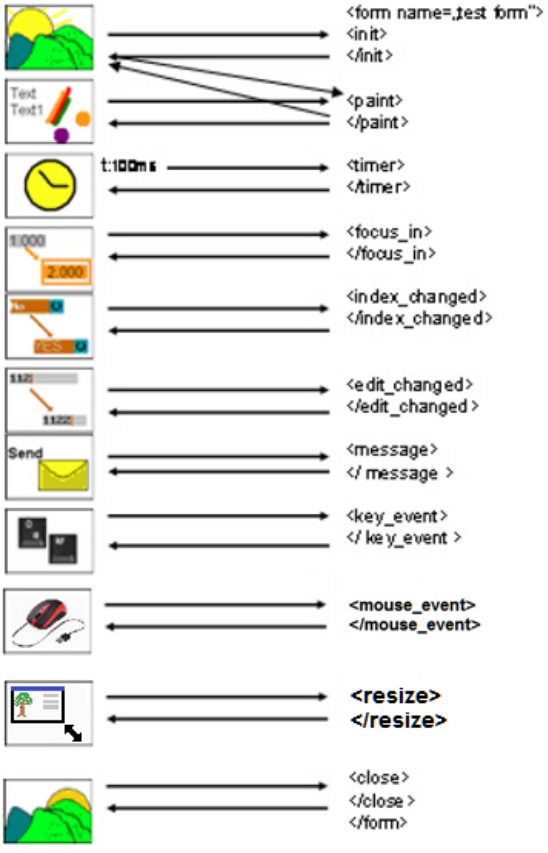
태그 이름	의미

FORM 계속	AUTOSCALE_CONTENT 속성을 사용해 양식의 컨트롤이 서로 다른 화면 해상도에서 동작하는 방식을 정의할 수 있습니다. 기본적으로 컨트롤은 이미 설정된 화면 해상도에 맞게 자동으로 조정됩니다. 위치 지정과 치수 측정을 직접 제어하고 싶으면 양식 태그에서 속성으로 OFF 값을 설정하십시오.
	
	속성:
	autoscale_content
	
	값:
	• **on**
	컨트롤의 좌표는 화면 해상도에 맞게 자동으로 조정됩니다(기본 설정).
	• **off**
	컨트롤의 좌표는 변하지 않은 상태로 사용됩니다.
	
	예:
	`<form name="main_form" autoscale_content="off">`
	…
	…
	…
	`</form>`

FORM 계속	속성: • **textfile** 이 속성을 사용해 양식 관련 텍스트 파일을 지정할 수 있습니다. 시스템은 표준 텍스트 컨텍스트로 식별자 EASY_XML을 사용합니다. • **textcontext** 이 속성을 사용해 사용될 텍스트 컨텍스트의 식별자를 지정할 수 있습니다. 이 속성은 **textfile** 속성과 함께만 유효합니다.

태그 이름	의미

FORM 계속	표준 레이아웃이나 easyscreen.ini 파일에 저장된 레이아웃을 사용하려면 LAYOUT 속성을 사용하십시오. 입력할 값은 사용될 레이아웃의 이름입니다.
	참고:
	호출 대화 상자가 숨겨져 있지 않기 때문에 팝업 창의 기본 설정을 변경하는 것이 좋습니다.
	
	속성:
	layout
	
	구문:
	`<form name="양식의 이름" layout="레이아웃의 이름" >`
	…
	…
	`</form>`
	
	예:
	`<form name="form0" layout="slstandardscreenlayout.`
	`SlStandardScreenLayout.LowerForm" >`
	…
	…
	`</form>`
	
	미리 정의된 화면 양식 위치 및 치수 외에 easyscreen.ini 파일에 자체 정의도 저장할 수 있습니다.
	easyscreen.ini의 항목:
	`[640x480]`
	`MyPanel = x:=0, y:=220, width:=340, height:=174`
	`[800x480]`
	`MyPanel = x:=0, y:=220, width:=420, height:=174`
	
	예:
	`<form name="form0" layout="MyPanel" >`
	…
	…
	`</form>`

태그 이름	의미
FORM 계속	대화 상자 메시지: • **INIT** • **PAINT** • **TIMER** • **CLOSE** • **FOCUS_IN** • **INDEX_CHANGED** • **EDIT_CHANGED** • **GESTURE_EVENT** • **KEY_EVENT** • **MESSAGE** • **MOUSE_EVENT** • **RESIZE** • **CHANNEL_CHANGED** • **LANGUAGE_CHANGED**
FORM 계속	

태그 이름	의미

FORM 계속	**구문:**
	`<FORM name = "<dialog name>" color = "#ff0000">`
	예:
	`<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	**대화 상자 메시지** 대화 상자가 생성된 후 바로 실행되는 태그입니다. 대화 상자 양식에 대한 모든 입력 요소 및 "핫링크"는 여기에 생성되어야 합니다.

태그 이름	의미

KEY_EVENT	대화 상자 메시지
	키보드 이벤트를 평가하는 KEY_EVENT 태그를 양식에 통합할 수 있습니다. 양식에 이 태그를 사용할 수 있는 경우 시스템이 MF2 키보드 코드를 활성 양식으로 전송합니다. 변수 **$actionresult** 가 0으로 설정되어 있지 않으면 시스템은 이어서 키보드 이벤트를 처리합니다.
	키보드 코드는 변수 **$keycode** 에 정수 값으로 제공됩니다.
	
	예:
	변수 **exclude_key** 에 입력된 문자는 입력 스트림에서 필터링해야 합니다.
	
	`<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>`

태그 이름	의미

MOUSE_EVENT	태그가 마우스 이벤트의 처리를 위해 스크립트로 링크될 수 있습니다. 마우스로 다음 작업을 수행할 때 실행됩니다.
	• 버튼 누름
	• 버튼 누름 해제
	• 마우스 이동
	파서는 구조체에서 정보를 제공하고 다음 요소를 사용해 구조체 변수 **$mouse_event** 를 생성합니다.
	
	구조체 요소:
	• **type**
	작업 코딩
	– 2 - 버튼 누름
	– 3 - 버튼 누름 해제
	– 5 - 마우스 이동
	• **x**
	커서의 X 위치(픽셀); 현재 화면 해상도 대비
	• **y**
	커서의 Y 위치(픽셀); 현재 화면 해상도 대비
	• **id**
	식별자
	– -1, 위치를 컨트롤에 지정할 수 없는 경우
	– != -1, 마우스 커서가 컨트롤 안에 있는 경우, 속성 **idemdata** 의 내용이 반환됩니다.
	• **button**
	이벤트 발생 시점에 버튼의 상태를 포함합니다.
	– 0 - 버튼 없음
	– 1 - 왼쪽 버튼
	– 2 - 오른쪽 버튼
	– 4 - 가운데 버튼
	버튼은 비트별 OR 연산으로 연결될 수 있습니다.
	
	예:
	`<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>`

태그 이름	의미

MESSAGE	**대화 상자 메시지**
	스크립트에서 Send_message 연산이 실행되면 파서가 태그 메시지를 처리합니다. 변수 P1 및 P2는 변수 $message_par1 및 $message_par2가 제공합니다 ("SEND_MESSAGE" 태그 참조).
	
	구문:
	`<MESSAGE>`
	`</MESSAGE>`
	
	예:
	`<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>`

태그 이름	의미

SEND_MESSAGE	이 태그가 두 개의 파라미터와 함께 메시지를 활성 양식으로 전송하고, 전송된 메시지는 태그 메시지 내에서 처리됩니다 (MESSAGE 참조).
	
	구문:
	`<SEND_MESSAGE>p1, p2</SEND_MESSAGE>`
	
	예:
	`<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>`

태그 이름	의미
RESIZE	**대화 상자 메시지** 태그가 RESIZE 이벤트의 처리를 위해 스크립트로 링크될 수 있습니다. 이 이벤트는 동적 해상도 전환에 의해 생성됩니다. 

태그 이름	의미

RESIZE 계속	**예:**
	`<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>`

태그 이름	의미

CHANNEL_CHANGED	**대화 상자 메시지**
	이 태그를 사용해 채널 변경 이벤트를 처리할 수 있습니다. 사용자가 채널 어드밴스 버튼을 누르면 실행됩니다.
	
	구문:
	`<channel_changed>`
	`</channel_changed>`
	
	예:
	채널이 변경되면 선택한 채널의 데이터가 표시될 수 있도록 양식이 닫혔다가 다시 열립니다.
	`<channel_changed>`
	`  <open_form name = "form1" reopen="true"/>`
	`</channel_changed>`
	
	또는
	
	채널이 변경되면 또 다른 메뉴가 활성화됩니다.
	`<channel_changed>`
	`  <navigation>menu2</navigation>`
	`</channel_changed>`

LANGUAGE_CHANGED	**대화 상자 메시지**
	이 태그를 사용해 언어 전환 요청을 처리할 수 있습니다. 사용자가 화면 양식이 열려 있는 동안 언어를 전환하면 실행됩니다.
	
	구문:
	`<language_changed>`
	`</language_changed>`
	
	예:
	언어가 전환되면 제어 시스템에 활성화된 언어의 텍스트 요소를 제공하기 위해 양식이 닫혔다가 다시 열립니다.
	`<language_changed>`
	`  <open_form name = "form1" reopen="true"/>`
	`</language_changed>`

태그 이름	의미

FOCUS_IN	**대화 상자 메시지**
	시스템이 제어에 초점을 맞출 경우에 호출되는 태그 입니다. 시스템은 컨트롤을 식별하기 위해 컨트롤의 이름을 변수 **$focus_name** 에 복사하고 **item_data** 속성의 값을 변수 **$focus_item_data** 에 복사합니다.
	예를 들어 이 메시지는 초점 위치에 따라 이미지를 출력하는 데 사용될 수 있습니다.
	예:
	`<focus_in>`
	`<PRINT text="focus on filed:%s, %d">$focus_name,`
	`$focus_item_data </PRINT>`
	`</focus_in>`

PAINT	**대화 상자 메시지** 대화 상자가 표시되면 실행되는 태그입니다. 대화 상자에 표시되는 모든 텍스트 및 이미지는 여기 규정되어야 합니다. 또한 시스템이 대화 상자의 일부를 다시 표시하도록 지정한 경우에도 이 태그가 실행됩니다. 예를 들어 상위 창을 닫으면 이 태그가 실행됩니다.
TIMER	**대화 상자 메시지** 주기적으로 실행되는 태그입니다. 각 양식에 타이머 태그를 매 100 ms마다 시작하는 타이머가 지정됩니다.

태그 이름	의미

CAPTION	이 태그는 대화 상자 제목을 포함합니다.
	이 태그는 INIT 태그 내에 사용해야 합니다.
	제목 표시줄은 여러 개의 열로 나눌 수 있습니다.
	제목 표시줄을 나누려면 텍스트를 출력하기 전에 "define_section" 속성으로 "caption" 태그를 프로그래밍해야 합니다.
	"define_section" 속성은 열의 개수를 지정합니다.
	
	"property" 속성은 각 열의 길이, 시작 위치, 텍스트 정렬 등을 정의하기 위해 사용합니다. 위치 및 길이 사양은 양식의 너비를 기준으로 하는 백분율 값입니다.
	"value" 속성의 값은 열 번호를 나타냅니다 (0으로 시작).
	
	`<caption define_sections="3">`
	`<property value="0" length="20" position="2" alignment="left" />`
	`<property value="1" length="20" position="79" alignment="right" />`
	`</caption>`
	
	텍스트를 열에 할당하려면 열 인덱스로 index 속성을 추가로 지정하면 됩니다.
	`<caption index="0">column1</caption>`
	`<caption index="1">column2</caption>`
	
	구문:
	`<CAPTION>Titel</CAPTION>`
	
	예:
	`<CAPTION>my first dialogue</CAPTION>`

CLOSE	**대화 상자 메시지** 이 태그는 대화 상자를 닫기 전에 실행됩니다.

태그 이름	의미

CLOSE_FORM	활성 대화 상자를 닫는 태그입니다.
	이 명령은 MMC 명령으로 대화 상자를 연 뒤 사용자에게 소프트 키로 대화 상자를 닫을 수 있도록 할 경우에만 필요합니다. 보통 대화 상자는 자동으로 관리되기 때문에 명령을 통해 닫을 필요가 없습니다.
	
	구문:
	`<CLOSE_FORM/>`
	
	예:
	`<softkey_ok>`
	` <caption>OK</caption>`
	` <CLOSE_FORM />`
	` <navigation>main_menu</navigation>`
	`</softkey_ok>`

태그 이름	의미
CONTROL	이 태그는 제어 요소 생성에 사용합니다. **구문:** `<CONTROL name = "<control name>" xpos = "<X 위치>" ypos = "<Y 위치>" refvar = "<NC 변수>" hotlink = "true" format = "<서식>" />` **참고:** 속성 값 설정에 사용되는 변수는 전역 변수로 선언되어야 합니다. **속성:** • **name** 필드 식별자. 이 식별자는 로컬 변수도 나타내기 때문에 양식에 동시에 여러 번 사용해서는 안 됩니다. • **xpos** 좌측 상단의 X 위치 • **ypos** 좌측 상단의 Y 위치 • **fieldtype** 필드 유형 유형이 규정되지 않은 경우, 필드는 편집 필드로 설정합니다. – **edit** 데이터를 변경할 수 있습니다. – **readonly** 데이터를 변경할 수 없습니다. **combobox** 필드가 숫자 값 대신 해당 식별자를 표시합니다. **combobox** 필드 유형을 선택한 경우 표시할 표현식을 필드에 지정해야 합니다. <ITEM> 태그가 이런 목적에 사용됩니다. 콤보 박스는 현재 선택된 텍스트의 인덱스를 컨트롤의 변수에 저장합니다 (**refvar** 속성 참조). 컨트롤이 생성되면 **addItem** 또는 **insertItem** 함수를 사용해 요소를 추가로 삽입할 수 있습니다. – **progressbar** 값 범위가 0~100인 진행 상태 표시줄을 표시합니다. 밸리 (valley) 값과 피크 값 속성을 사용하여 디스플레이 될 데이터 범위를 조정할 수 있습니다.

태그 이름	의미

CONTROL 계속	• **fieldtype edit**및 **readonly**
	필드 유형 **edit** 및 **readonly**의 경우 **multiline** 속성을 사용해 텍스트를 여러 줄에 표시할 수 있습니다. 줄 바꿈은 단어 경계에서 발생하거나 줄 바꿈 문자 <LF>를 사용해 발생합니다.
	예:
	`<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>`

태그 이름	의미

CONTROL 계속	• **fieldtype**
	– **graphicbox**
	두 번째 점선 그래픽 컨트롤을 생성하는 필드 유형입니다. <ITEM> 태그를 사용해 컨트롤에 그래픽 요소를 삽입할 수 있습니다. 상자의 너비 및 높이는 **width** 및 **height** 파라미터로 지정합니다.
	컨트롤이 생성되면 **addItem** 또는 **insertItem** 함수를 사용해 요소를 추가로 삽입할 수 있습니다. 이 컨트롤에서는 **itemdata** 파라미터를 평가하지 않습니다.
	기준 변수가 컨트롤에 지정되면 값은 100 ms 시간 기준으로 기록됩니다. 최대 10000개의 값을 기록할 수 있습니다. **max** 속성을 사용하면 값이 끝없이 기록됩니다. 최대 기록 시간에 도달하면 가장 오래된 값이 삭제됩니다. 기록 시간은 밀리초 단위로 지정해야 합니다.
	값이 이산 시간에 저장되더라도 컨트롤이 연결된 선으로 표시합니다.**channel** 속성은 최대 3개의 추가 기준 변수의 기록을 허용합니다. 인덱스는 1부터 시작합니다. 인덱스 0은 컨트롤 태그에 지정된 변수의 속성을 설정하는데 사용됩니다.
	예:
	`<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>`

태그 이름	의미
CONTROL 계속	**graphicbox 예:** `<CONTROL name= "graphic" xpos = "8" ypos="23" width="300" height="352" fieldtype="graphicbox" />` • 요소 추가: 요소는 **additem** 또는 **loaditem** 함수를 사용하여 추가합니다. 다음과 같은 두 번째 요소를 사용할 수 있습니다. – 라인 - l(inc) – 부채꼴 - c(ircle) – 포인트 - p(oint) **요소의 구조** <Element type>; 좌표 • 라인: l; xs; ys; xe, ye l - 라인 표시 Xs - X 시작 위치 Ys - Y 시작 위치 Xe - X 끝 위치 Ye - Y 끝 위치 • 원: C, xs, ys, xe, ye, cc_x, cc_y, r C - 부채꼴 표시 Xs - X 시작 위치 Ys - Y 시작 위치 Xe - X 끝 위치 Ye - Y 끝 위치 Cc_x - X 좌표, 원 중심점 CC_y - Y 좌표, 원 중심점 • 반경: R • 점: P, x, y P - 점 표시 X - X 위치 Y - Y 위치 • 그래픽 삭제: 내용은 **비우기** 함수를 사용하여 삭제합니다.

태그 이름	의미
CONTROL 계속	다음 필드 유형은 "사용자 지정 버튼 구성 (페이지 245)" 섹션에 설명되어 있습니다. • **fieldtype** – **pushbutton** 이 필드 유형은 푸시버튼이나 스위치로 사용됩니다. – **radiobutton** 이 필드 유형은 여러 옵션 중 하나를 선택할 수 있도록 합니다. – **checkbox** 이 필드 유형은 여러 옵션을 선택할 수 있도록 합니다. – **groupbox** 이 필드 유형은 컨트롤 그룹을 프레임 및 타이틀과 함께 시각적으로 묶습니다. – **switch** 이 필드 유형은 아이콘을 이용해 두 상태 중 하나에 대한 신호를 보내는 그래픽 요소입니다. – **scrollarea** 이 필드 유형은 지정 영역 내의 컨트롤 표시에 사용됩니다.

태그 이름	의미
CONTROL 계속	• **fieldtype** – **listbox** 빈 목록 상자 컨트롤을 생성하는 필드 유형입니다. <ITEM> 태그를 사용해 목록 상자 요소를 목록 상자에 삽입할 수 있습니다. ITEM 태그에 **value** 속성을 사용하면 이 요소에 고유 값을 지정할 수 있습니다. 이 속성은 예를 들어 요소 식별에 사용할 수 있습니다. 목록 상자의 너비 및 높이는 **width** 및 **height** 파라미터로 지정합니다. 컨트롤이 생성되면 **addItem**, **insertItem** 또는 **loadItem** 함수를 사용해 목록 상자 요소를 추가로 삽입할 수 있습니다. 컨트롤이 생성되면 **addItem** 또는 **insertItem** 함수를 사용해 목록 상자 요소를 추가로 삽입할 수 있습니다. 이 컨트롤에서는 **itemdata** 파라미터를 평가하지 않습니다. – **itemlist** 숫자 값 대신 해당 식별자를 표시하는 정적 컨트롤을 생성하는 필드 유형입니다. 필드의 식별자는 <ITEM> 태그를 사용해 지정할 수 있습니다. • **item_data** 사용자별 정수 값이 속성에 지정될 수 있습니다. 이 값은 초점 필드 확인을 위한 FOCUS_IN 메시지의 일부입니다. • **refvar** 필드와 연결될 수 있는 기준 변수의 식별자 (옵션). • **hotlink** = "TRUE" " 기준 변수의 값이 변경되면 필드가 자동 업데이트됩니다 (옵션). • **format** 지정된 변수의 디스플레이 형식을 정의하는 속성입니다. 데이터 서식 지정(옵션) 방법은 print-Tag 를 참조하십시오. 또한 **Format** 속성을 사용해 정수 값을 다른 숫자 시스템으로 표시하거나 필드 유형 **edit** 및 **readonly**를 위한 Boolean 값으로 표시할 수 있습니다. 문자 수 및 정렬을 사용한 서식 지정은 지원되지 않습니다. 다음과 같은 표현 유형을 사용할 수 있습니다. • **%o** - 8진수 • **%x** - 16진수 • **%n** - 2진수 • **%b** - bool

태그 이름	의미

CONTROL 계속	**열 생성 예, 속성:**
	`<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>`

CONTROL 계속	**속성 형식 예:**
	`<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>`

태그 이름	의미
CONTROL 계속	**속성:** • **color_bk** 컨트롤의 배경 색상을 설정하는 속성입니다. • **color_fg** 컨트롤의 전경 색상을 설정하는 속성입니다. **자세한 정보**는 "색 코딩 (페이지 82)" 장을 참조하십시오. • **display_format** 지정된 변수의 처리 형식을 정의하는 속성입니다. PLC 실수 변수는 더블 워드를 읽어 액세스하기 때문에 PLC 실수 변수에 액세스할 때는 반드시 이 속성을 사용해야 합니다. 허용되는 데이터 형식은 다음과 같습니다. – FLOAT – INT – DOUBLE – STRING 표현식 (예: 표시할 텍스트 또는 그래픽 요소) 를 목록 상자, 그래픽 상자 또는 콤보 박스에 지정: **구문:** <ITEM>표현식</ITEM> <ITEM value ="<값>">표현식</ITEM>

태그 이름	의미

CONTROL 계속	**예:**
	`<CONTROL name = "button1" xpos = "10" ypos = "10" fieldtype = "combobox ">`
	`<ITEM>text1</ITEM>`
	`<ITEM>text2</ITEM>`
	`<ITEM>text3</ITEM>`
	`<ITEM>text4</ITEM>`
	`</CONTROL>`
	임의 정수 값을 표현식에 지정할 경우 속성 **value = "값"**을 태그에 추가해야 합니다.
	제어 변수는 이제 일련 번호라기 보다는 항목의 지정 값을 포함합니다.
	
	예:
	`<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>
	상태 진행 바의 예:
	`<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>`
	목록 상자의 예:
	`<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" />`
	• 요소 추가:
	요소는 **additem** 또는 **loaditem** 함수를 사용하여 추가합니다.
	• 내용 삭제:
	내용은 **empty** 함수를 사용하여 삭제합니다.
	
	`<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>`

태그 이름	의미

CONTROL 계속	**예 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 계속	이미지 상자 컨트롤은 비트맵 또는 GIF 형식의 그림을 관리합니다. 그림이 표시된 영역보다 크면 컨트롤이 스크롤 막대를 표시합니다. 표시 영역을 제어하기 위해 시스템은 CONTROL.IMAGEBOXSET 함수를 제공합니다. **자세한 정보**는 "사전 정의된 기능 (페이지 174)" 장을 참조하십시오. **구문:** `<function name="control.imageboxget"> ... </function>`
CONTROL 계속	**속성:** • **disable** 이 속성은 편집 컨트롤에서 입력을 금지/허용합니다. • **tooltip** 커서가 컨트롤에 위치하면 정보 텍스트가 표시됩니다. • **factor** 변환 계수 • **font** 글꼴 정의 • **fontpixelsize** 문자 높이 - 값은 픽셀 수로 지정되고 640x480 픽셀의 해상도를 나타냅니다. 표시된 문자 높이는 실제 해상도와 기준 해상도의 비율로부터 결정됩니다. **예:** `<control name="c2" xpos="180" ypos="120" fontname="arial" fontpixelsize="26" height="100" type="string" />`

태그 이름	의미
CONTROL 계속	**속성:** • **fontname** 이 속성은 사용될 글꼴을 지정하는 데 사용됩니다. 설치되는 글꼴은 HMI.GET_FONT_FAMILIES 함수를 사용해 결정할 수 있습니다. **값:** 글꼴 이름 다음과 같은 Siemens 글꼴이 시스템에 설치됩니다. • Siemens AD CH Simplified • Siemens AD CH Traditional • Siemens AD JP • Siemens AD KS • Siemens AD Mono • Siemens AD Mono CH Simplified • Siemens AD Mono CH Traditional • Siemens AD Mono JP • Siemens AD Mono KS • Siemens AD Mono TH • Siemens AD Mono VN • Siemens AD Sans • Siemens AD TH • Siemens AD VN • Siemens Logo • Siemens Sans Cond Global • Siemens Sans Cond Mono • Siemens Sans Global MS Windows 기반 시스템에는 추가 글꼴이 포함될 수 있습니다. **예:** `<control name="c2" xpos="180" ypos="120" fontname="arial" fontpixelsize="26" height="100" type="string" />`

태그 이름	의미
CONTROL 계속	**작성 후 제어 시스템 변경** 제어 태그는 작성 후 기존 제어 시스템의 속성을 변경합니다. 태그에 변경할 시스템의 이름과 새 속성을 지정해야 합니다. 변경은 양식 태그 내에서만 실행할 수 있습니다. 다음과 같은 속성을 설정할 수 있습니다. 예: • name • xpos • ypos • width • height • color_bk • color_fg • access level • fieldtype • itemdata • min • max • default • disable • tooltip • font • factor 참조 값은 수정할 수 없습니다. 소프트 키 이벤트에 의한 트리거로 속성을 변경할 경우 메시지 전송 태그가 이 요청을 양식 컨텍스트로 전달해야 합니다. 메시지 태그는 메시지를 가져오는데 사용됩니다.

태그 이름	의미

CONTROL 계속	**조작 명령의 컨트롤 변경**
	런타임 중 컨트롤 속성 변경은 조작 명령에서도 가능합니다. 따라서 컨트롤 이름과 새 값을 지정할 속성을 정의해야 합니다. 속성은 점으로 컨트롤 이름과 구분됩니다.
	구문:
	<이름>.<속성>
	
	예:
	`...`
	`...`
	`<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>`

태그 이름	의미

HELP_CONTEXT	이 태그는 호출해야 하는 도움말 주제를 정의합니다. INIT 블록에 프로그램 해야 합니다.
	808/802D sl 시스템
	속성에 지정된 이름에 접두어 XmlUserDlg_ 를 붙인 다음 도움말 시스템으로 전송합니다. 도움말 파일의 관련 구조는 도움말 주제 중 온라인 도움말 생성의 구조를 적용해야 합니다.
	도움말 시스템 활성화 순서:
	1. "정보" 키를 누릅니다.
	2. 대화 상자가 표현식 "my_dlg_help" 를 제공합니다.
	3. 파서가 표현식을 "XmlUserDlg_my_dlg_help" 로 변환합니다.
	4. 도움말 시스템이 활성화됩니다.
	5. 검색어 "XmlUserDlg_my_dlg_help" 를 제출합니다.
	840D sl/828D 시스템
	도움말 설명서의 레이아웃과 생성에 관한 **추가 정보**:
	SINUMERIK Operate 스타트업 매뉴얼
	
	속성:
	• **name**
	특정에 지정된 이름은 도움말 시스템으로 전달됩니다. **entry** 태그에 도움말 설명서에서 사용되는 파일 이름을 지정해야 합니다.
	• **anchor**
	이 속성을 사용해 키워드를 입력할 수 있습니다.
	
	구문:
	`<HELP_CONTEXT name="<file name>" anchor="key word"/>`
	
	예:
	`<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>`

태그 이름	의미
DATA_ACCESS	사용자 입력이 저장 중일 때 대화 상자 양식의 반응 행동을 제어하는 태그입니다. 반응 행동은 INIT 태그에 정의해야 합니다. 태그가 사용되지 않으면 입력은 각 사례에 버퍼링됩니다. 예외: **hotlink** 속성을 **true** 로 설정한 컨트롤은 항상 직접 기록하거나 읽어옵니다. **속성:** • **type** = "TRUE" – 입력 값을 버퍼링하지 않습니다. 대화 상자 양식은 입력 값을 기준 변수에 바로 복사합니다. • **type** = "FALSE" – **UPDATE_CONTROLS** type = "FALSE" 태그가 있는 기준 변수에만 값을 복사합니다. **예:** `<DATA_ACCESS type = "true" />`

태그 이름	의미

DEBUG	이 태그는 스크립트에 프로그램할 수 있는 트레이스 출력을 저장하는 데 사용됩니다. **text** 속성은 저장될 텍스트를 기록하는 데 사용됩니다. 이 속성과 속성 값은 인쇄 명령의 **text** 속성과 값에 해당합니다.
	기본적으로 debug 태그는 시스템 속도를 늦추기 때문에 실행되지 않습니다. debug 출력을 활성화하려면 **DialogGui** 또는 **AGM** 태그에서 **on** 값으로 **debug_msg** 속성을 프로그램하십시오.
	debug 출력은 다음 파일에 파서 오류 메시지와 함께 저장됩니다.
	card/oem/sinumerik/hmi/dvm/log/dvm.log
	
	구문:
	`<debug text="<텍스트는 인쇄 명령 참조>">변수는 인쇄 명령 참조</debug>`
	
	Easy XML 예:
	`<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>`
	
	dvm.log 파일의 항목:
	`date: 10/22/ 2018 time: 09:38:44:351`
	`Debug xmldial.xml 17: main_form control test_bool`
	`created value:255`
	
	Easy Extend 예:
	`<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>`
	
	dvm.log 파일의 항목:
	`date: 10/22/ 2018 time: 09:38:44:351`
	`Debug xmldial.xml 17: main_form control test_bool`
	`created value:255`

태그 이름	의미

EDIT_CHANGED	**대화 상자 메시지**
	편집 컨트롤의 내용이 변경되면 이 태그가 호출됩니다.
	시스템은 컨트롤을 식별하기 위해 컨트롤의 이름을 변수 **$focus_name** 에 복사하고 **item_data** 속성의 값을 변수 **$focus_item_data** 에 복사합니다.
	
	예:
	`<EDIT_CHANGED>`
	`<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>`
	…
	…
	`</EDIT_CHANGED>`

GESTURE_EVENT	이 태그는 멀티 터치 조작을 위한 손가락 동작 수행에 사용됩니다. 이 태그는 "멀티터치 조작 (페이지 237)" 섹션에 설명되어 있습니다.

INDEX_CHANGED	**대화 상자 메시지**
	작업자가 콤보 박스 선택을 변경하면 이 태그가 호출됩니다.
	시스템은 컨트롤을 식별하기 위해 컨트롤의 이름을 변수
	$focus_name 에 복사하고 **item_data** 속성의 값을 변수 **$focus_item_data** 에 복사합니다.
	참고:
	이때 컨트롤에 지정된 기준 변수는 컨트롤 변수에 맞게 조정되지 않기 때문에 기준 변수는 선택을 변경하기 전 콤보 박스의 인덱스를 포함하게 됩니다.
	
	예:
	`<INDEX_CHANGED>`
	`<print text="index changed filed:%s, %d"> $focus_name, $focus_item_data </print>`
	…
	…
	`</INDEX_CHANGED>`

태그 이름	의미
MENU	이 태그는 열어야 하는 소프트 키 설명과 대화 상자를 포함하는 메뉴를 정의합니다. **속성:** • **name** 메뉴 이름 **구문:** <MENU name = "<menu name>"> … <open_form …> … <SOFTKEY …> </SOFTKEY> </MENU>

태그 이름	의미

NAVIGATION	이 태그는 호출해야 하는 메뉴를 정의합니다. 소프트 키 블록, 메뉴 블록 및 양식 내에서 사용될 수 있습니다. 변수 이름이 값으로 태그에 지정되면 파서가 변수에 저장된 메뉴를 활성화합니다.
	메뉴 블록에서 탐색은 명령의 위치에 있습니다. 후속 명령은 더 이상 실행되지 않습니다.
	
	참고:
	변수 내용으로 탐색 대상을 판단하도록 하려면 이 변수가 전역 변수로 선언되어야 합니다.
	
	구문:
	`<NAVIGATION>menu name</NAVIGATION>`
	
	예:
	`<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>

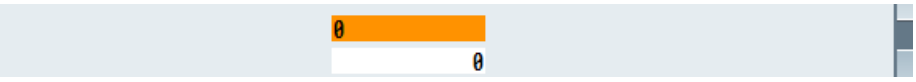
태그 이름	의미

OPEN_FORM	이름 아래에 있는 대화 상자 양식을 여는 태그입니다.
	지정된 화면 양식이 이미 활성화된 경우 이 태그는 실행되지 않습니다.
	reopen 속성을 입력해 화면 양식이 다시 열리도록 강제할 수 있습니다. 이는 시스템 상태가 변경되어 화면 양식 내용의 재배치가 필요한 경우에 필요할 수 있습니다.
	
	속성:
	• **name**
	대화 상자 양식 이름
	• **reopen**
	속성 값이 **true**로 설정된 경우 **open_form** 태그가 다시 호출되면 화면 양식이 활성 양식인지 확인합니다. 이 경우 파서가 활성 양식을 닫은 다음 다시 엽니다.
	
	구문:
	`<OPEN_FORM name = "<form name>" />`
	
	예:
	`<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>`

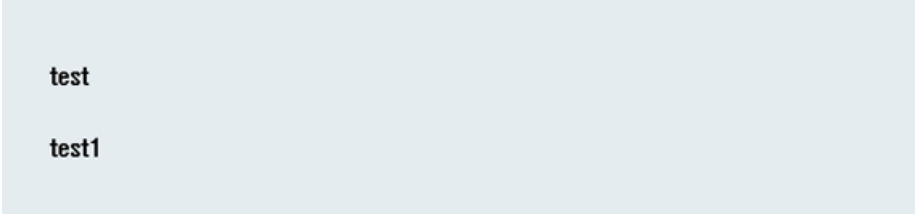
태그 이름	의미

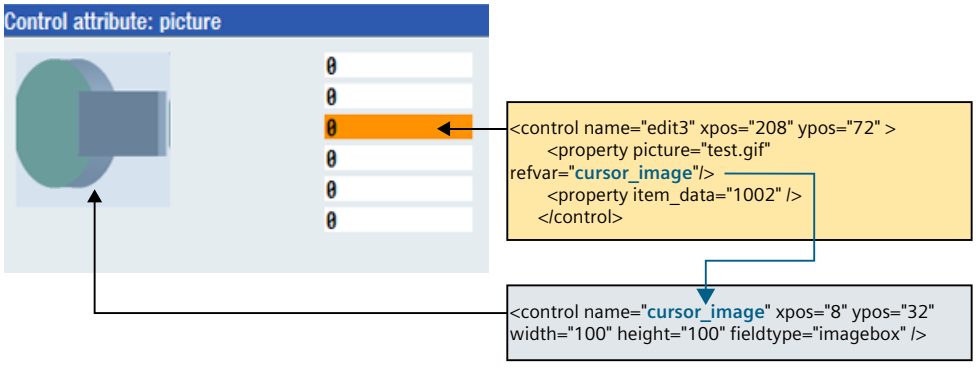
PROPERTY	이 태그는 화면 제어를 위한 추가 속성 정의에 사용할 수 있습니다. 태그는 컨트롤 태그에 포함됩니다.
	
	속성:
	• **max** = "<최대값>"
	• **min** = "<최소값>"
	• **default** = "<기본값>"
	• **factor** = "변환 계수"
	• **color_bk** = "<배경 색상 코드>"
	• **color_fg** = "<폰트 색상 코드>"
	• **password** = "<true>" - 입력한 문자를 "*"로 표시합니다.
	• **multiline** = "<true>" - 편집 컨트롤에 여러 줄을 입력하도록 허용합니다.
	• **disable** = "<true/false>" - 편집 컨트롤에 입력을 차단/허용합니다.
	• **transparent** = "비트맵의 투명 색상"
	자세한 정보는 "색 코딩 (페이지 82)" 장을 참조하십시오.
	• **tooltip** = 커서가 컨트롤에 설정된 경우 정보 텍스트가 표시됩니다.
	구문: <property tooltip="note text" />
	• **abscissa** = "첫 번째 좌표 축의 이름"(그래픽 상자에만 허용)
	• **ordinate** = "두 번째 좌표 축의 이름"(그래픽 상자에만 허용)
	• **fix_time_area** = "on" - **min** 및 **max** 속성은 신호가 표시되는 기간을 정의합니다. 표시된 신호는 오른쪽에서 왼쪽으로 이동합니다.
	예:
	`<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>`

태그 이름	의미

PROPERTY 계속	**속성:**
	alignment - 이 속성은 왼쪽 또는 오른쪽 텍스트 정렬을 허용합니다. 기본적으로 텍스트는 왼쪽 정렬입니다.
	
	값:
	• **left**
	• **right**
	
	구문:
	`alignment="<right/left>"`
	
	예:
	`<control name="edit2" xpos="208" ypos="52" >`
	`<property alignment="right"/>`
	`</control>`
	
	 

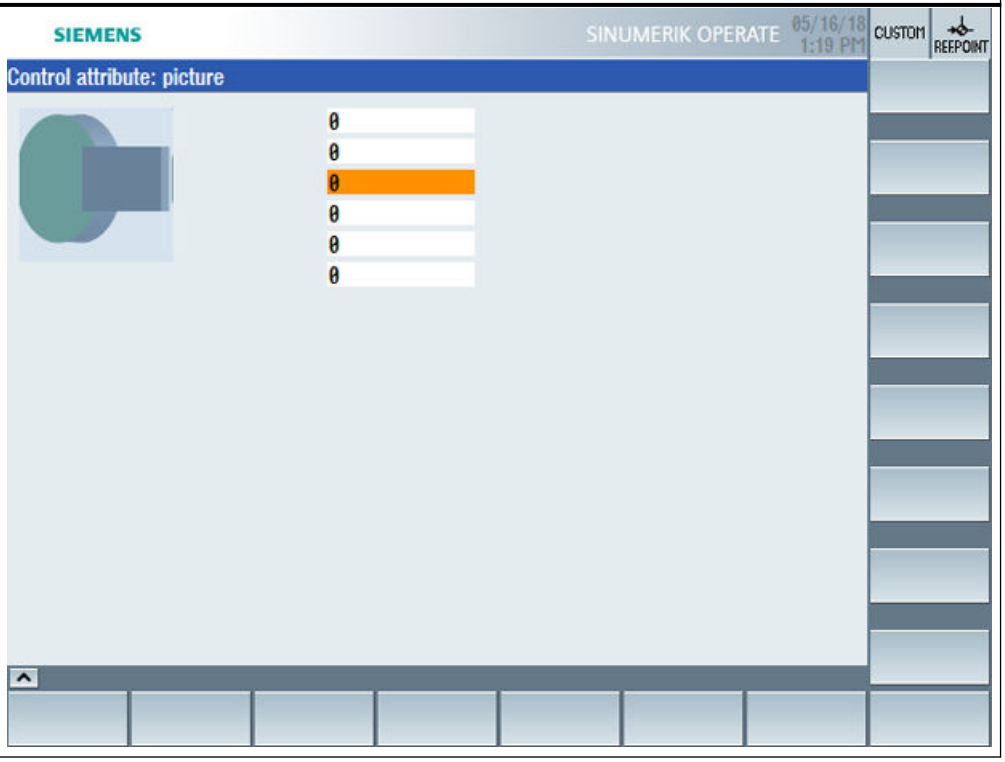
태그 이름	의미

PROPERTY 계속	**속성:**
	parent - 이 속성은 제어 시스템의 구성원 자격을 결정합니다. 기본적으로 컨트롤이 양식에 할당됩니다. 컨트롤을 스크롤 보기에 할당하려면 부모 창으로 지정해야 합니다.
	
	값:
	부모 창의 이름
	
	예:
	`...`
	`<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" />`
	`...`

태그 이름	의미
PROPERTY 계속	**속성:** **picture** - 이 속성은 표시될 그래픽을 지정합니다. 입력 포커스가 이 필드에 설정된 경우 그래픽 이름이 변수에 저장되거나 지정된 그래픽이 표시됩니다. 이 속성은 데이터 싱크를 정의할 수 있도록 **refvar** 속성과 함께 사용됩니다. 그래픽이나 애니메이션을 표시하려면 **imagebox** 유형 컨트롤의 이름이 지정되어 있어야 합니다. 이 컨트롤은 그래픽 관리를 담당합니다. 로컬 변수의 이름이 지정되는 경우 이 변수는 반드시 **String** 데이터 유형이어야 합니다. 그래픽은 새 이름이 기준 변수에 할당될 때까지 계속 표시됩니다. **구문:** `<property picture="<Name der Grafik>" refvar="<Datensenke>"/>`
PROPERTY 계속	**필드 연결 예:** 

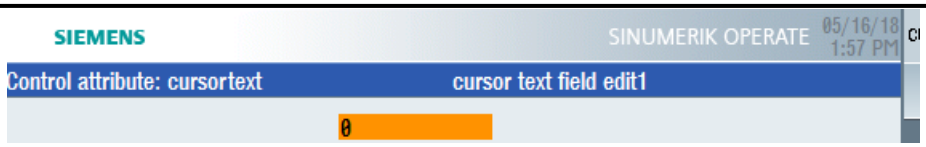
태그 이름	의미

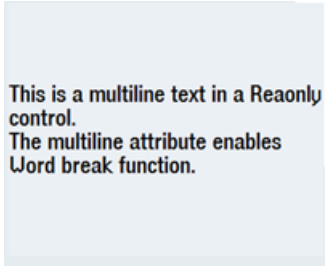
PROPERTY 계속	**예:**
	`<let name="cursor_icon" type="string" />`
	` <form name="main_form1">`
	`  <init>`
	`  <caption>Control attribute: picture</caption>`
	`  <control name="cursor_image" xpos="8" ypos="32" width="100" height="100" fieldtype="imagebox" />`
	`  <control name="edit1" xpos="208" ypos="32" >`
	`  <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>`

태그 이름	의미
	 SIEMENS SINUMERIK OPERATE 05/16/18 1:19 PM CUSTOM REFPOINT Control attribute: picture 0 0 0 0 0 0

태그 이름	의미

PROPERTY 계속	**속성:**
	cursortext - 이 속성은 표시될 커서 텍스트를 정의합니다.
	
	입력 포커스가 이 필드에 설정되면 텍스트가 제목 표시줄에 표시됩니다.
	이 속성 외에 텍스트 정렬과 커서 텍스트 영역을 정의하는 2개의 속성이 추가로 지정될 수 있습니다. 텍스트는 표준으로 왼쪽 정렬됩니다. 커서 텍스트 영역은 기본으로 제목 표시줄 너비의 50%를 차지합니다.
	
	`alignment="<right/left>"` - 텍스트 정렬 오른쪽 맞춤/왼쪽 맞춤
	`length="<Width>"` - 커서 텍스트에 필요한 제목 표시줄의 비율
	
	구문:
	`<property cursortext="<text>" />`
	또는
	`<property cursortext="<text>" alignment="<right/left>" />`
	또는
	`<property cursortext="text2" alignment="r="<text>" alignment="<right/left>" length="<비율>"/>`
	
	예:
	`<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>`

태그 이름	의미
	

PROPERTY 계속	**속성:**
	multiline - 이 속성은 편집 또는 읽기 전용 컨트롤에서 텍스트의 여러 줄 출력을 허용합니다.
	
	구문:
	`<property multiline="true" />`
	
	예:
	`...`
	`...`
	`<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>`
	
	 

태그 이름	의미
PROPERTY 계속	**속성:** • **help_context** 이 속성을 사용해 도움말 주제를 컨트롤에 할당할 수 있습니다. **entry** 태그에 도움말 설명서에서 사용되는 파일 이름을 지정해야 합니다. • **anchor** 이 속성을 사용해 키워드를 입력할 수 있습니다. 이 속성은 **help_context** 속성과 함께만 유효합니다. **구문:** `<property help_context="<file name>" anchor="<key word>"/>` **예:** `<control name = "cpwd" xpos = "322" ypos = "64" > <property password="true" /> <property help_context="chapter_1.html" anchor="Keyword_2" /> </control>`
SOFTKEY	이 태그는 소프트 키의 속성 및 반응을 정의합니다. **속성:** • **position** 소프트 키 번호. 1-8은 수평 소프트 키, 9-16은 수직 소프트 키 다음 속성은 아래와 같이 하면 유효하게 됩니다. • **type** 소프트 키 속성을 정의합니다. user_controled - 소프트 키 표시 방법을 정의하는 스크립트입니다. toggle_softkey - 소프트 키를 눌렀을 때와 누르지 않았을 때 소프트 키를 다르게 표시합니다. • **refvar** 반드시 toggle_softkey 와 함께 사용해야 합니다. 실제 소프트 키 속성은 기준 변수에 복사됩니다. 변수 유형은 "String"으로 지정해야 합니다. 이 변수에 소프트 키를 누르거나 누르지 않았을 때, 또는 소프트 키가 잠겼을 때 속성이 포함됩니다 (state 태그 참조). • **picture** 이 속성을 사용하면 비트맵을 소프트 키 좌측에 정렬시켜 출력할 수 있습니다. 전체 경로 이름을 지정해야 합니다. 표시 가능한 텍스트 문자의 개수는 비트맵 너비에 맞게 줄어듭니다.

태그 이름	의미
SOFTKEY 계속	소프트 키 블록 내에 다음과 같은 추가 작업을 정의할 수 있습니다. • **picturealignment** 이 속성으로 이미지 방향을 지정합니다. 이미지는 기본적으로 소프트 키의 왼편에 정렬됩니다. 정렬을 위해 다음 값을 지정할 수 있습니다. – **top** 위 정렬 – **bottom** 아래 정렬 – **left** 왼쪽 정렬 – **right** 오른쪽 정렬 – **center** 가운데 정렬 • **caption** 소프트 키 텍스트 • **state** 반드시 user_controlled 와 함께 사용해야 합니다. 필요한 소프트 키 디스플레이를 시스템에 지정하는 태그입니다. **구문:** <state type="<state>" /> 다음과 같은 문자열을 지정할 수 있습니다. – **notpressed** 소프트 키를 누르지 않았을 때 소프트 키를 표시합니다. – **pressed** 소프트 키를 눌렀을 때 소프트 키를 표시합니다. – **disabled** 소프트 키를 잠그고 회색으로 표시합니다. • **navigation** • **update_controls** • **function**

태그 이름	의미
SOFTKEY 계속	**구문:** 표준 소프트 키: <state type="<softkey state>" /> <softkey position = "<1>"> … … </softkey> 또는 스크립트 제어 소프트 키: <softkey position = "<1>" type="<user_defined>" > <state type="<softkey state>" /> … … </softkey> 또는 토글 소프트 키: <softkey position = "<1>" type="<toggle_softkey>" refvar="<variable name>" > … … </softkey> **예:** <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>

태그 이름	의미
SOFTKEY 계속	**예:** 또는 <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	"OK" 소프트 키의 반응을 정의하는 태그입니다.  소프트 키 블록 내에 다음과 같은 추가 작업을 정의할 수 있습니다. • **navigation** • **update_controls** • **function** **구문:** <SOFTKEY_OK> … … </SOFTKEY_OK>

태그 이름	의미

SOFTKEY_CANCEL	"취소" 소프트 키의 반응을 정의하는 태그입니다.
	 
	소프트 키 블록 내에 다음과 같은 추가 작업을 정의할 수 있습니다.
	• **navigation**
	• **update_controls**
	• **function**
	구문:
	`<SOFTKEY_CANCEL>`
	…
	…
	`</SOFTKEY_CANCEL>`

SOFTKEY_BACK	"뒤로" 소프트 키의 반응을 정의하는 태그입니다.
	 
	소프트 키 블록 내에 다음과 같은 추가 작업을 정의할 수 있습니다.
	• **navigation**
	• **update_controls**
	• **function**
	구문:
	`<SOFTKEY_BACK>`
	…
	…
	`</SOFTKEY_BACK>`

태그 이름	의미

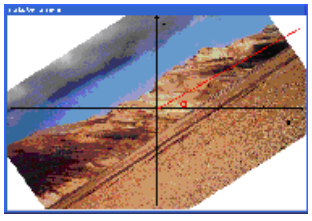
SOFTKEY_ACCEPT	"승인" 소프트 키의 반응을 정의하는 태그입니다.
	 
	
	소프트 키 블록 내에 다음과 같은 추가 작업을 정의할 수 있습니다.
	• **navigation**
	• **update_controls**
	• **function**
	구문:
	`<SOFTKEY_ACCEPT>`
	…
	…
	`</SOFTKEY_ACCEPT>`

TEXT	이 태그를 이용하여 텍스트를 지정된 위치에 표시합니다. 알람 번호가 사용될 경우 대화 상자는 그 번호에 저장되어 있는 텍스트를 표시합니다. **구문:** `<TEXT xpos = "<X 위치>" ypos = "<Y 위치>"> Text </TEXT>` **속성:** • **xpos** 좌측 상단의 X 위치 • **ypos** 좌측 상단의 Y 위치 • **color** 텍스트 색상 **자세한 정보**는 "색 코딩 (페이지 82)" 장을 참조하십시오. **값:** 디스플레이 해야 하는 텍스트

태그 이름	의미

IMG	태그를 이용하여 텍스트를 지정된 위치에 디스플레이합니다. BMP 및 PNG 이미지 형식이 지원됩니다.
	
	구문:
	`<IMG xpos = "<X 위치>" ypos = "<Y 위치>" name = "<이름>" />`
	`<IMG … xrot=" X축 각도 " yrot=" Y축 각도 " zrot=" Z축 각도 " />`
	
	속성:
	• **xpos**
	좌측 상단의 X 위치
	• **ypos**
	좌측 상단의 Y 위치
	• **name**
	전체 경로 이름. 이 경로 정보는 소문자로 표시됩니다.
	• **transparent**
	비트맵의 투명 색상
	자세한 정보는 "색 코딩 (페이지 82)" 장을 참조하십시오.

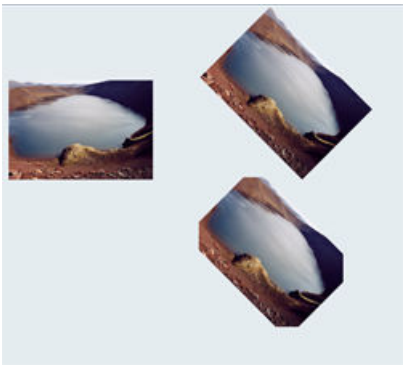
태그 이름	의미

IMG 계속	**예:**
	Z 축을 중심으로 34도 각도로 이미지를 회전합니다.
	`<img xpos = "5" ypos = "23" name = "f:/appl/pic3.bmp" height="355" width="550" zrot="34" color_bk="#ffffff" ></img>`
	
	참고:
	드라이브 지정은 시스템에 따라 다릅니다.
	 
	
	선택 사항:
	이미지를 원본 크기와 다른 크기로 표시하려면 **width** 및 **height** 속성을 사용해 치수를 정의할 수 있습니다.
	• **width**
	너비 (픽셀 단위)
	• **height**
	높이 (픽셀 단위)
	
	예:
	`<IMG xpos = "20" ypos = "40" name = "f:/appl/test.bmp" />`
	`<IMG xpos = "5" ypos = "23" name = "f:/appl/test.bmp" height = "355" width = "550"/>`

태그 이름	의미

IMG 계속	**속성:**
	AspectRatioMode
	이 속성을 이용하여 비비례적 확대가 수행되는 이미지의 가로 세로 비율 제어를 할 수 있습니다.
	
	값:
	• **Ignore**
	비트맵의 가로 세로 비율은 지정된 높이 및 너비에 맞게 조절됩니다.
	• **Keep** (기본값)
	가로 세로 비율이 유지되고 지정된 높이 및 너비 안에서 최대로 확장한 사각형으로 이미지 크기가 조절됩니다.
	• **KeepByExpanding**
	가로 세로 비율이 유지되고 지정된 높이 및 너비 밖으로 최대로 확장한 사각형으로 이미지 크기가 조절됩니다.
	
	예:
	`<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>`
	 
	• 위 - **KeepbyExpanding** 속성 지정
	• 왼쪽 아래 - **Ignore** 속성 지정
	• 오른쪽 아래 - **Keep** 속성 지정

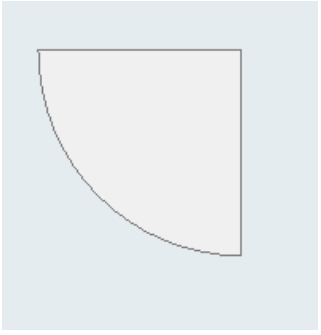
태그 이름	의미

IMG 계속	**속성:**
	ExpandingForRotation
	가로 세로 비율이 유지됩니다. 이미지는 회전 및 크기 조절된 이미지의 경계 상자에 해당하는 사각형으로 크기가 조절됩니다.
	
	예:
	`<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>`
	 
	• 왼쪽 - 이미지가 150x155 픽셀 크기로 조절됩니다.
	• 오른쪽 위 - 이미지가 150x155 픽셀 크기로 조절되고 **ExpandingForRotation** 속성으로 45도 회전합니다.
	• 오른쪽 아래 - 이미지가 150x155 픽셀 크기로 조절되고 45도 회전합니다.

태그 이름	의미
ARC	이 태그는 양식에서 부채꼴을 그리는 데 사용됩니다. **속성:** 양식 내 위치(픽셀): • **xpos1** - 활꼴 X 좌표의 시작점 • **ypos1** - 활꼴 Y 좌표의 시작점 • **xpos2** - 활꼴 X 좌표의 끝점 • **ypos2** - 활꼴 Y 좌표의 끝점 양식 내 위치(픽셀): • **ccx** - 활꼴 X 좌표의 중심 • **ccy** - 활꼴 Y 좌표의 중심 • **radius** - 원 반경, 값(픽셀) **참고:** **중심점 또는 반경** 두 파라미터가 모두 지정된 경우 원 중심이 사용됩니다.
ARC 계속	• **penwidth** 이 속성은 픽셀로 선 두께를 지정합니다. • **color** 선 색상 • **color_bk** 활꼴의 채우기 색상 **자세한 정보**는 "색 코딩 (페이지 82)" 장을 참조하십시오. • **style** 이 속성은 선 유형을 정의합니다. – **"SolidLine"** - 실선(기본값) – **"DashLine"** - 파선 – **"DotLine"** - 점선 – **"DashDotLine"** - 일점 쇄선 – **"DashDotDotLine"** - 이점 쇄선

태그 이름	의미
ARC 계속	• **type** – **cw** - 시계 방향 – **ccw** - 시계 반대 방향 • **arrow** 활꼴 끝에 화살표가 표시됩니다. – **"P1"** - 선의 시작점에 있는 화살표 – **"P2"** - 선의 시작점에 있는 화살표 – **"P1P2"** - 선의 시작점 및 끝점에 있는 화살표 – **"P1_FILLED"** - 선의 시작점에 있는 채워진 화살표 – **"P2_FILLED"** - 선의 시작점에 있는 채워진 화살표 – **"P1P2_FILLED"** - 선의 시작점 및 끝점에 있는 채워진 화살표 • **arrow_size** Arrow 속성이 사용되면 화살표 길이를 픽셀로 지정할 수 있습니다.

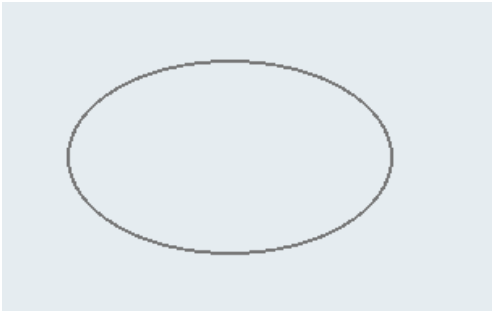
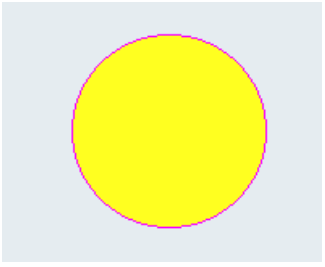
태그 이름	의미

ARC 계속	**예:**
	 
	`<arc xpos1="20" ypos1="200" xpos2="180" ypos2="200" ccx="100" ccy="200" type="ccw" PENWIDTH="1" color="#777777" arrow="p1p2_filled"/>`
	
	또는
	
	`<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"/>`

태그 이름	의미
BOX	태그는 지정된 위치에 지시된 색에 따라 직사각형을 그립니다. **구문:** `<BOX xpos = "<X 위치>" ypos = "<Y 위치>" width = "<X 확장>" height = "<Y 확장>" color = "<색상 코드>" />` **속성:** • **xpos** 좌측 상단의 X 위치 • **ypos** 좌측 상단의 Y 위치 • **width** X 방향으로 확장 (픽셀 단위) • **height** Y 방향으로 확장 (픽셀 단위) • **color** 색 코딩 **자세한 정보**는 "색 코딩 (페이지 82)" 장을 참조하십시오.

태그 이름	의미
ELLIPSE	이 태그는 양식에서 타원형을 그리는 데 사용됩니다. **속성:** 양식 내 위치(픽셀): • **xpos** - X 좌표 선의 시작점 • **ypos** - Y 좌표 선의 시작점 • **width** - 주변 직사각형의 너비(bounding box) • **height** - 주변 직사각형의 높이(bounding box) • **penwidth** 이 속성은 픽셀로 선 두께를 지정합니다. • **color** 선 색상 • **color_bk** 타원형의 채우기 색상 **자세한 정보**는 "색 코딩 (페이지 82)" 장을 참조하십시오. • **style** 이 속성은 선 유형을 정의합니다. – **"SolidLine"** - 실선(기본값) – **"DashLine"** - 파선 – **"DotLine"** - 점선 – **"DashDotLine"** - 일점 쇄선 – **"DashDotDotLine"** - 이점 쇄선

태그 이름	의미

ELLIPSE 계속	**예:**
	 
	`<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"/>`

태그 이름	의미

FUNCTION	**함수 호출**
	"**name**" 속성에 지정된 함수 본문을 실행하는 태그입니다.
	
	속성:
	• **name** = "함수 본문 이름"
	• **return** = "함수의 결과를 저장할 변수 이름"
	
	값:
	함수 본문으로 전송해야 하는 변수 목록 변수는 콤마 (,) 로 구분되어야 합니다. 최대 10개의 파라미터를 전송할 수 있습니다.
	상수 또는 텍스트 표현식을 호출 파라미터로 지정할 수도 있습니다. 식별자 _T는 텍스트 표현식을 식별하기 위한 수단이기 때문에 표현식 맨 앞에 위치해야 합니다.
	
	구문:
	`<FUNCTION name = "<function name>" />`
	호출 함수는 반환 값을 예측합니다.
	`<FUNCTION name = "<function name>" return = "<Variablenname>" />`
	파라미터 전송
	`<FUNCTION name = "<function name>"> var1, var2, var3 </FUNCTION>`
	`<FUNCTION name = "<function name>"> _T"Text", 1.0, 1 </FUNCTION>`
	
	예:
	"FUNCTION_BODY" 참조.

태그 이름	의미

FUNCTION_BODY	**함수 본문**
	이 태그는 하위 함수의 함수 본문을 포함합니다. 함수 본문은 DialogGui 태그 내에 프로그램되어야 합니다.
	
	속성:
	• **name** = "함수 본문 이름"
	• **parameter** = "파라미터 목록" (옵션)
	필요한 전송 파라미터를 목록으로 나열하는 속성입니다. 파라미터는 콤마 (,)로 구분되어야 합니다.
	함수 본문이 호출될 때 함수 호출 시 지정된 파라미터 값이 목록에 나열되어 있는 전송 파라미터로 복사됩니다.
	• **return** = "true" (옵션)
	이 속성을 true 로 설정하면 로컬 변수 $return 이 생성됩니다. 함수를 끝낼 때 호출 함수로 전달되는 함수의 반환 값을 이 변수에 복사해야 합니다.
	
	구문:
	
	파라미터가 없는 함수 본문
	`<FUNCTION_BODY name = "<function name>">`
	…
	…
	…
	`</ FUNCTION_BODY>`
	
	파라미터가 있는 함수 본문
	`<FUNCTION_BODY name = "<function_name>" parameter = "<p1, p2, p3>">`
	…
	`<LET name = "tmp></LET>`
	`<OP> tmp = p1 </OP>`
	…
	`</FUNCTION_BODY>`
	
	반환 값이 있는 함수 본문
	`<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>`

태그 이름	의미

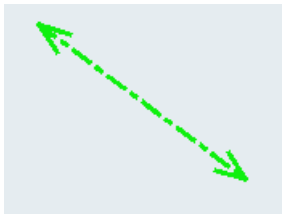
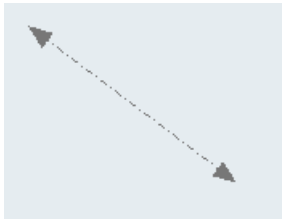
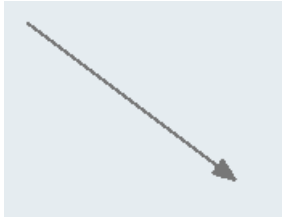
FUNCTION_BODY 계속	**예:**
	`<function_body name = "test" parameter = "c1,c2,c3" return = "true">`
	`<LET name = "tmp">0</LET>`
	`<OP> tmp = c1+c2+c3 </OP>`
	`<OP> $return = tmp </OP>`
	`</function_body>`
	…
	…
	…
	`<LET name = "my_var"> 4 </LET>`
	`<function name = "test" return = " my_var "> 2, 3,4</function>`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

LINE	이 태그는 양식에서 선을 그리는 데 사용됩니다. **속성:** 양식 내 위치(픽셀): • **xpos1** - X 좌표 선의 시작점 • **ypos1** - Y 좌표 선의 시작점 • **xpos2** - X 좌표 선의 끝점 • **ypos2** - Y 좌표 선의 끝점 • **penwidth** 이 속성은 픽셀로 선 두께를 지정합니다. • **color** 선 색상 **자세한 정보**는 "색 코딩 (페이지 82)" 장을 참조하십시오. • **style** 이 속성은 선 유형을 정의합니다. – **"SolidLine"** - 실선(기본값) – **"DashLine"** - 파선 – **"DotLine"** - 점선 – **"DashDotLine"** - 일점 쇄선 – **"DashDotDotLine"** - 이점 쇄선

태그 이름	의미

LINE 계속	• **arrow**
	선 끝에 화살표가 표시됩니다.
	– **"P1"** - 선의 시작점에 있는 화살표
	– **"P2"** - 선의 시작점에 있는 화살표
	– **"P1P2"** - 선의 시작점 및 끝점에 있는 화살표
	– **"P1_FILLED"** - 선의 시작점에 있는 채워진 화살표
	– **"P2_FILLED"** - 선의 시작점에 있는 채워진 화살표
	– **"P1P2_FILLED"** - 선의 시작점 및 끝점에 있는 채워진 화살표
	• **arrow_size**
	Arrow 속성이 사용되면 화살표 길이를 픽셀로 지정할 수 있습니다.
	
	구문:
	`<line xpos1="<X 좌표 시작점>" ypos1="<Y 좌표 시작점>" xpos2="<X 좌표 끝점>" ypos2="<Y 좌표 끝점>"`
	`PENWIDTH="<선 두께>" color="<선 색상>" style="<선 스타일>" arrow="<화살표 유형>" arrow_size="<화살표 크기>"/>`

태그 이름	의미

LINE 계속	**예:**
	`<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"/>`
	 

태그 이름	의미

REQUEST	태그를 이용하여 싸이클 읽기 서비스에 변수를 추가합니다(핫링크). 그 결과 컨트롤에 연결되지 않은 변수에 대한 액세스 시간이 감소됩니다.
	값이 변경되면 자동으로 기능이 호출되게 하려면 기능 이름을 추가 속성으로 지정해야 합니다.
	이 태그는 INIT 연산 내에서만 처리됩니다.
	
	속성:
	• **name**
	주소 식별자
	• **function**
	함수 이름
	
	구문:
	`<REQUEST name = "<NC-Variable>" />`
	또는
	`<REQUEST name = "<NC-Variable>" function="<function name>"/>`
	
	예:
	`<request name ="plc/mb10" />`
	
	또는
	
	`<function_body name="my_function" >`
	`<print text="value changed" />`
	`</function_body>`
	…
	…
	…
	`<request name ="plc/mb10" function="my_function"/>`

태그 이름	의미

RECALL	Recall 버튼을 통해 탐색이 발생하면 메뉴에서 이 태그를 사용할 수 있습니다. 태그가 프로그램되면 Recall 아이콘이 표시되고, 파서가 Recall 버튼을 처리할 수 있습니다.
	
	구문:
	`<recall>`
	…
	…
	`</recall>`

UPDATE_CONTROLS	연산자 컨트롤 및 기준 변수 사이의 비교를 실행하는 태그입니다.
	
	속성:
	• **type**
	이 속성은 데이터 비교의 방향을 정의합니다.
	= TRUE – 기준 변수에서 데이터를 읽어 조작 컨트롤에 복사합니다.
	= FALSE – 조작 컨트롤에서 데이터를 읽어 기준 변수에 복사합니다.
	
	구문:
	`<UPDATE_CONTROLS type = "<방향>"/>`
	
	예:
	`<SOFTKEY_OK>`
	`< UPDATE_CONTROLS type="false"/>`
	`</SOFTKEY_OK>`

3.8 사용자 메뉴 생성하기

3.8.1 프로세싱 싸이클 양식 작성

싸이클 지원 기능을 사용하면 양식 대화 상자를 이용해 싸이클 호출을 자동 생성 및 역컴파일할 수 있습니다.

이 기능을 관리할 수 있도록 다음의 태그가 제공됩니다.

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

FORM 태그에 싸이클 양식을 지정하려면 **type** 속성의 값을 **cycle**로 지정해야 합니다. 싸이클 양식을 지정하면 **NC_INSTRUCTION**을 처리할 수 있습니다.

예

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

NC_INSTRUCTION 태그에 생성할 싸이클 호출이 있습니다. 모든 싸이클 파라미터는 스페이스 리테이너를 사용해 보존해야 합니다.

예

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

CREATE_CYCLE 태그가 스페이스 리테이너 변수에 저장된 값을 준비하고 NC 명령을 생성합니다.

그 다음에 이 값은 지정된 변수로 복사됩니다.

태그 이름	설명

NC_INSTRUCTION	생성할 NC 명령을 정의하는 태그입니다.
	나열된 모든 싸이클 파라미터가 **FORM**의 문자열 변수로 자동 생성되고 **FORM**에 사용할 수 있습니다.
	
	사전조건: FORM의 **type** 속성의 값이 **CYCLE**로 설정되어 있어야 합니다.
	
	속성:
	• **refvar**
	이 태그에 기준 변수가 정된 경우, 모든 파라미터는 기준 변수 내 저장된 NC 블록의 값으로 사전 할당됩니다.
	구문:
	`<NC_INSTRUCTION> 자리 표시자가 있는 NC 명령 </ NC_INSTRUCTION>`
	
	예:
	`<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>`

태그 이름	설명

CREATE_CYCLE	NC 블록을 생성하는 태그입니다. 이 블록의 구문은 **NC_INSTRUCTION** 태그의 값에 의해 정의됩니다.
	NC 명령을 생성하기 전에 파서가 FORM의 CYCLE_CREATE_EVENT 태그를 호출합니다. 이 태그는 싸이클 파라미터 계산에 사용할 수 있습니다.
	구문:
	`<CREATE_CYCLE/>`
	옵션:
	기준 변수가 지정되면 명령이 생성된 호출을 이 변수로 복사합니다.
	`<create_cycle refvar="name" />`
	속성:
	• **refvar**
	이 태그에 기준 변수가 지정된 경우 이 변수에 NC 명령이 복사됩니다.
	예:
	`<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>`
	또는
	`<SOFTKEY_OK>`
	`<caption>OK</caption>`
	`<CREATE_CYCLE refvar= "cyc_string" />`
	`<close_form />`
	`<navigation>main_menu</navigation>`
	`</SOFTKEY_OK>`

3.8.2 대체 문자

시스템은 런타임에 대한 제어 속성 (속성 값) 정의 옵션을 제공합니다. 이 기능을 이용하기 위해, 해당 속성은 로컬 변수에 셋팅되어야 하고 변수 이름은 **character $** 및 속성 값으로서 태그에 전송되어야 합니다.

태그가 속성 값 또는 값으로 문자열을 받는 경우 변수 이름 앞에 **$$$** 문자를 붙여야 합니다.

예:

```
<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 struct_def.xml 파일 생성

작업 순서

일부 함수에서는 형식 정의에 **struct_def.xml** 파일을 사용합니다. 이 XML 파일이 없을 경우 다음 마크업을 사용해 파일 사본을 생성한 다음 함수 프로젝트에 통합하십시오.

참고

UTF8 형식으로 XML 파일을 생성하십시오.

스크립트

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 컴포넌트 주소 지정

NC 변수, PCL 블록 또는 드라이브 데이터의 주소 지정을 위해 해당 데이터에 대한 주소 이름이 생성되어야 합니다. 주소는 서브 경로 **구성요소 이름** 및 **변수 주소**로 구성됩니다. 사선 (/)은 구분 문자로서 사용되어야 합니다.

3.10.1 PLC 주소 지정

PLC 주소는 경로를 나타내는 **plc**로 시작합니다.

도표 3-3 아래 주소가 허용됩니다.

DBx.DB(f)	데이터 블록
I(f)x	입력
Q(f)x	출력
M(f)x	비트 메모리
V(f)x	변수

DBx.DBXx.b	데이터 블록
Ix.b	입력
Qx.b	출력
Mx.b	비트 메모리
Vx.b	변수

도표 3-4 데이터 형식 **f**:

B	바이트
W	워드
D	더블 워드

데이터 형식 이름은 비트 주소 지정에 적용될 수 없습니다.

주소 **x**:

S7-200 유효 주소 이름

비트 주소 지정:

b - 비트 번호

예:

```
<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 변수 주소 지정

NC 변수 주소는 경로를 나타내는 **nck**로 시작합니다.

이 섹션 다음에는 데이터 주소가 옵니다. 데이터 주소의 구조는 List Manual NC 변수 및 인터페이스 신호로부터 가져와야 합니다.

예:

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

3.10.3 채널 특정 주소 지정

주소 토큰에 채널 번호가 지정되어 있지 않으면 접근은 항상 작동 소프트웨어의 채널 1로 이뤄집니다.

데이터를 특정 채널로부터 읽는 것이 필요한 경우, 원하는 채널 숫자를 가진 식별자 **u** (장치)가 주소에 추가됩니다.

예제:

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

3.10.4 런타임 중 NC/PLC 주소 생성

런타임 중에 주소 식별자를 생성할 수 있습니다.

이 옵션에서는 문자열 변수의 내용이 연산 구문, nc.cap.read 및 nc.cap.write 기능에 주소로 사용됩니다.

주소를 이런 방식으로 지정하는 경우 다음 규칙을 준수해야 합니다.

- 변수 이름은 큰따옴표 안에 씁니다.
- 변수 이름 앞에 접두어로 ‚$' 문자 3개를 붙입니다.

구문:

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

예:

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

3.10.5 드라이브 컴포넌트 주소 지정

드라이브 콤포넌트의 주소는 경로를 나타내는 **drive**로 시작합니다.

그런 다음 드라이브 장치가 지정됩니다:

CU – 제어 유닛

DC – 드라이브 제어(모터 모듈)

CULNK – 확장 모듈(HUBs)

TM – 단말 모듈

LM – 라인 모듈

설정해야 하는 파라미터가 이 섹션에 추가됩니다.

예:

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

<!– CU에서 r0002 값 읽기 ->

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

<!– NX1에서 r0002 값 읽기 ->
<OP> r0002_content = "drive/cu/r0002[CU2]" </OP>

<!– CU에서 p107[0] 값 읽기 ->
<OP> p107_content = "drive/cu/p107[0]" </OP>
<PRINT text="%d"> p107_content </PRINT>

<!– CU에서 p107[0] 값 읽기 ->

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

<!– NX1에서 p107[0] 값 읽기 ->

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

드라이브 개체 주소 지정:

개별 개체 주소 지정을 위해 해당 개체가 반드시 파라미터 다음의 꺽쇠 괄호에 입력되어야 합니다.

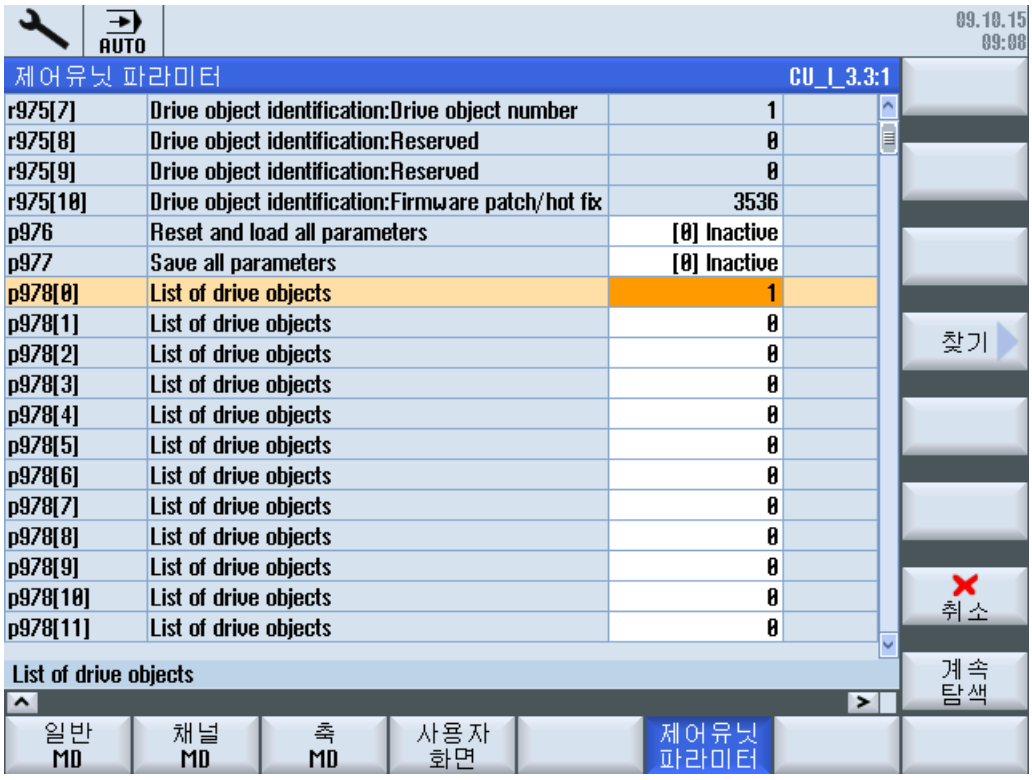


그림 3-7 컨트롤의 드라이브 파라미터 p0978 [...]

파라미터 번호[do<DO-index>]

예:

```
p0092[do1]
```

다른 방법으로, **$<변수 이름>** "대체 문자"를 이용하여 로컬 변수에서 드라이브 인덱스를 읽어올 수 있습니다.

```
z.B. DO$로컬 변수
```

예:

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

간접 주소 지정:

```
<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 예: 모터 모듈을 위한 DO 숫자를 결정합니다.

유형 11(서보)의 DO 숫자는 다음과 같이 결정됩니다.

모든 연결된 드라이브 오브젝트가 슬롯 번호와 함께 관련 CU의 p978 필드에 표시됩니다. 컴포넌트 유형 숫자는 p101 필드에 한꺼번에 목록화되어 있습니다.

별도 색인법은 다음 컴포넌트 유형 각각에 대해 사용됩니다.

CU – 제어 장치

DC – 드라이브 제어 (모터 모듈)

CULNK – 확장 모듈(HUBs)

TM – 단말 모듈

LM – 라인 모듈

주소 지정 색인은 연결된 개별 CU를 위한 p107 필드를 오름차순으로 쭉 훑어봄으로써 결정할 수 있고, 유형 색인은 원하는 유형이 발생할 때마다 하나씩 증가합니다. 기본값은 1입니다. NX 컴포넌트가 필드에서 발견된 경우, NX 컴포넌트 상의 집계는 현재 배열이 완전히 끝날 때까지 중단됩니다. NX 및 CU 컴포넌트는 발견되는 순서대로 실행됩니다.

색인 결정: NX를 갖는 CUI

CUI		NX1		주소 지정 색인	
				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	

이 토폴로지는 7개의 모터 모듈을 포함합니다. 1에서 4까지의 지수는 CU에 할당된 모터 모듈을 지정합니다. 5에서 7까지의 지수는 NX의 모듈을 지정합니다. 지수 1은 CU에 접근할 때 사용됩니다. NX는 지수에 의해 두 번 지정됩니다.

샘플 스크립트: xmldial.xml and 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 머신 데이터 및 셋팅 데이터 주소 지정

드라이브 및 셋팅 데이터는 문자 $ 과 그 다음에 오는 데이터 이름으로 확인됩니다.

머신 데이터:

$Mx_<이름[**index, AX<축_번호>**]>

셋팅 데이터:

$Sx_<이름[인덱스, AX<축_번호>]>

x:

N – 일반 머신 또는 셋팅 데이터

C – 채널 머신 또는 셋팅 데이터

A - 축 머신 또는 셋팅 데이터

인덱스:

필드의 경우 파라미터는 데이터의 인덱스를 표시합니다.

AX<축_번호>:

축별 데이터에 요구되는 축 (**<축_번호>**) 이 규정되어야 합니다.

다른 방법으로, **$<변수 이름>** "대체 문자"를 이용하여 로컬 변수에서 축 인덱스를 읽어올 수 있습니다.

예: AX$로컬변수

예:

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

축의 직접 주소 지정:

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

축의 간접 주소 지정:

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

3.10.8 채널별 머신 데이터

어떤 채널 숫자도 해당 주소 내에 정의되지 않았다면, 접근은 항상 작동 소프트웨어의 현재 설정된 채널로 이뤄집니다.

데이터를 특정 채널로부터 읽는 것이 필요한 경우, 꺽쇠괄호 내 원하는 채널 숫자를 가진 식별자 **u** (장치)가 주소에 추가됩니다. 배열 주소 지정에서는 채널 정의가 꺽쇠괄호 내 마지막 아규먼트입니다.

예제:

```
$MC_RESET_MODE_MASK
```

또는

```
$MC_AXCONF_GEOAX_ASSIGN_TAB[0]
```

```
$MC_RESET_MODE_MASK[u1]
```

또는

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

3.10.9 사용자 데이터 주소 지정

사용자 데이터의 주소를 지정할 때는 경로 부분 **gud**로 시작하고 그 뒤에 GUD가 전역인지 아니면 채널별인지 나타내는 마킹이 옵니다. 이 주소 부분 뒤에는 GUD 영역과 GUD 이름이 옵니다.

필드의 경우 이름 다음 꺽쇠괄호 ([]) 안에 필요한 필드 인덱스를 지정해야 합니다.

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

전역 사용자 데이터 주소 지정

주소를 지정할 때는 경로 섹션 gud로 시작하고 그 다음에 NCK 영역을 지정합니다. 이 주소 섹션 다음에는 GUD 영역을 지정합니다.

GUD 영역	지정
sgud	Siemens GUD
mgud	장비 제조업체 GUD
ugud	사용자 GUD

채널별 사용자 데이터의 주소 지정

주소를 지정할 때는 경로 섹션 gud로 시작하고 그 다음에 CHANNEL 영역을 지정합니다. 이 주소 섹션 다음에는 GUD 영역을 지정합니다.

그 다음에 GUD 이름을 입력하십시오. 배열의 주소를 지정해야 하는 경우 이름 뒤에 대괄호 ([]) 에 묶인 배열 서브스크립트가 옵니다.

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

다차원 배열 주소지정

다차원 배열의 주소를 지정할 때 줄 인덱스 뒤에는 열 인덱스가 와야 합니다. 인덱스는 마침표로 서로 구분하여 입력합니다.

다음은 채널별 GUD에 해당됩니다.

- 채널 번호는 줄/행 표기 전 또는 후에 **u** (유닛)로 표시한 다음 숫자를 입력합니다.
- 시퀀스는 콤마(,)로 구분해야 합니다.
- 채널 번호를 입력하지 않으면 첫 번째 채널에 액세스합니다.

예:

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

또는

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

3.11 사전 정의된 기능

스크립트 언어는 다양한 문자열 처리 함수와 표준 수학 함수를 제공합니다. 아래 목록의 함수 이름은 예비용으로 오버 로드될 수 없습니다.

함수 이름	의미

Ncfunc cap read	지정된 주소의 값을 로컬 변수로 복사하는 기능입니다. 읽기 작업에 에러가 없으면 반환 변수에 값 0이 포함됩니다.
	
	연산 명령과 달리 이 기능은 에러가 발생해도 스크립트 작업 처리를 인터럽트하지 않습니다.
	
	속성:
	• **return** - 실행 상태
	– 값 = 0 - 에러 없음
	– 값 = 1 - 변수를 읽을 수 없음
	• **rows** - 읽어야 할 배열의 추가 라인 개수(옵션)
	이 인덱스에서처럼, 기준 변수의 배열 인덱스가 정의되면 함수가 읽은 값을 대상 변수로 복사합니다.
	
	구문:
	`<function name="ncfunc.cap.read" return="error"> lokale variable, "address"</function>`
	
	예:
	`<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>`
	
	또는
	
	`<let name="cu_p978" dim="25" ></let>`
	`...`
	`...`
	`<function name="ncfunc.cap.read" return="error" rows="23">cu_p978, "drive/cu/p0978[0, 1]"`
	`</function>`

함수 이름	의미

Ncfunc cap write	값을 지정된 변수에 쓰는 기능입니다. 읽기 작업이 에러 없이 실행되면 반환 변수에 값 0이 반환됩니다.
	
	연산 명령과 달리 이 기능은 에러가 발생해도 스크립트 작업 처리를 인터럽트하지 않습니다.
	
	속성:
	• **return** - 실행 상태
	– 값 = 0 - 에러 없음
	– 값 = 1 - 변수를 읽을 수 없음
	• **rows** - 써야 할 배열의 추가 라인 개수(옵션)
	이 인덱스에서처럼, 기준 변수의 배열 인덱스가 정의되면 함수가 값을 대상 변수로 복사합니다.
	
	구문:
	`<function name="ncfunc.cap.read" return="error"> local variable or constant, "address"</function>`
	
	예:
	`<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>`
	
	또는
	
	`<let name="cu_p978" dim="25" ></let>`
	`...`
	`...`
	`<function name="ncfunc.cap.write" return="error" rows="23">cu_p978, "drive/cu/p0978[0, 1]"`
	`</function>`

함수 이름	의미

Ncfunc PI-Service	프로그램 호출 (PI) 서비스를 사용해 작업을 NCK로 전송할 수 있습니다.
	PI 서비스가 에러 없이 실행되면 반환 변수에 값 1이 반환됩니다.
	
	공구 목록 조작:
	_N_CREATO - 공구 생성
	_N_DELETO - 공구 삭제
	_N_CREACE - 공구 절삭날 생성
	_N_DELECE - 공구 절삭날 삭제
	
	워크 옵셋 활성화:
	_N_SETUFR - 실제 사용자 프레임 활성화
	_N_SETUDT - 실제 사용자 데이터 활성화
	
	블록 검색:
	_N_FINDBL - 블록 검색 활성화
	_N_FINDAB -- 블록 검색 취소
	
	머신 데이터 재구성:
	_N_CONFIG - 작업자가 차례로 입력한 활성화 레벨의 머신 데이터 NEW_CONF가 동시에 활성화됩니다.
	
	구문:
	`<function name="ncfunc.pi_service" return="return var"> pi name, var1, var2, var3, var4, var5`
	`</function>`
	
	속성:
	name - 기능 이름
	return- 실행 결과를 저장할 변수의 이름:
	• 값 == 1 – 작업이 성공적으로 실행되었습니다.
	• 값 == 0 – 작업이 실패했습니다.
	
	태그 값:

함수 이름	의미
	pi name - PI 서비스 이름 (문자열) var1 ~var5 - PI 전용 인수

함수 이름	의미

Ncfunc PI-Service 계속	**인수:**
	• _N_CREATO
	var1- 공구 번호
	• _N_DELETO
	var1- 공구 번호
	• _N_CREACE
	var1 - 공구 번호
	var2- 절삭날 번호
	• _N_DELECE
	var1- 공구 번호
	var2- 절삭날 번호
	• _N_SETUFR
	인수 없음
	• _N_SETUDT
	var1- 활성화할 사용자 데이터 영역
	– 1 - 공구 옵셋 데이터
	– 2 - 활성 베이직 프레임
	– 3 - 활성 조정 프레임
	• _N_FINDBL
	var1 - 탐색 모드
	– 2 - 형상 계산을 적용해 탐색
	– 4 - 블록 종점 탐색
	– 1 - 형상 계산 없이 블록 검색
	• _N_FINDAB
	인수 없음
	• _N_CONFIG
	인수 없음
	
	예:
	공구 번호 3인 공구 생성
	
	`<function name="ncfunc.pi_service">_T"_N_CREATO", 3</function>`
	
	공구 5의 절삭날 1 삭제
	
	`<function name="ncfunc.pi_service">_T"_N_DELECE", 5, 1`
	`</function>`

함수 이름	의미

ncfunc chan PI-Service	채널 관련 방식으로 PI 서비스를 실행하는 함수입니다. 채널 번호는 PI 서비스 이름 다음에 전달됩니다. 이 다음에는 모든 다른 호출 파라미터가 옵니다.
	
	파라미터:
	channel - 채널 번호
	
	구문:
	`<function name="ncfunc.chan_pi_service" return="error">`
	`_T"_N_SETUFR", channel, ...`
	`</function>`
	
	예:
	`<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	컨트롤에 정의된 부동 소수점 수 변환 규칙을 제공하는 기능입니다. 문자열 변수를 변수로 지정해야 합니다.
	
	디스플레이 머신 데이터 MD203 DISPLAY_RESOLUTION 및 MD204 DISPLAY_RESOLUTION_INCH를 참조하십시오.
	
	구문:
	`<function name="ncfunc.displayresolution" return="dislay_res" />`
	
	예:
	`<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"/>`

함수 이름	의미
Ncfunc Get drive by axis name	이 함수는 관련 축 이름을 지정하여 모터 모듈의 주소 할당 인덱스를 반환합니다. 할당을 확인할 수 없는 경우 함수가 값 -1을 반환합니다. 예를 들어, 축/드라이브 할당이 수행되지 않은 경우 발생할 수 있습니다. **파라미터:** **str** - 축 이름 **반환값:** 모터 모듈의 인덱스 - 지정된 인덱스가 기록된 변수 이름. **구문:** `<function name="NCFUNC.GETDRVBYAX_NAME" return="<index>"> <axis name></function>` **예:** `<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>`

함수 이름	의미

Ncfunc Get drive by axis index	이 함수는 관련 축 인덱스를 지정하여 모터 모듈의 주소 할당 인덱스를 반환합니다.
	할당을 확인할 수 없는 경우 함수가 값 -1을 반환합니다. 예를 들어, 축/드라이브 할당이 수행되지 않은 경우 발생할 수 있습니다.
	
	파라미터:
	index - 축 인덱스
	
	반환값:
	모터 모듈의 인덱스 - 지정된 인덱스가 기록된 변수 이름.
	
	구문:
	`<function name="NCFUNC.GETDRVBYAX_IDX" return="<index>">`
	`<axis index></function>`
	
	예:
	`<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>`

함수 이름	의미

Ncfunc Get drive by drive name	이 함수는 모터 모듈 이름을 지정하여 모터 모듈의 주소 할당 인덱스를 반환합니다.
	할당을 확인할 수 없는 경우 함수가 값 -1을 반환합니다. 예를 들어, 축/드라이브 할당이 수행되지 않은 경우 발생할 수 있습니다.
	
	파라미터:
	string - 모터 모듈 이름
	
	반환값:
	모터 모듈의 인덱스 - 지정된 인덱스가 기록된 변수 이름.
	
	구문:
	`<function name="NCFUNC.GETDRVBYDRV_NAME" return="<index>">`
	`<drive name></function>`
	
	예:
	`<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>`

함수 이름	의미
Ncfunc Get drive by bud address	이 함수는 모터 모듈의 버스 주소를 지정하여 모터 모듈의 주소 할당 인덱스를 반환합니다. 할당을 확인할 수 없는 경우 함수가 값 -1을 반환합니다. 예를 들어, 축/드라이브 할당이 수행되지 않은 경우 발생할 수 있습니다. **파라미터:** **bus** - 버스 번호 **slave** - 슬레이브 번호 **component** - 콤포넌트의 번호 **반환값:** 모터 모듈의 인덱스 - 지정된 인덱스가 기록된 변수 이름. **구문:** `<function name="NCFUNC.GETDRVBYBUS_ADDR" return="<index>"><bus>,<slave>,<component></function>` **예:** `<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>`

함수 이름	의미

Ncfunc Get MD limits	이 함수는 머신 데이터의 한계를 지정된 변수, 데이터 형식 **StructMDLimits**로 복사합니다. 구조 정의는 struct_def.xml 파일에 포함됩니다.
	자세한 정보는 "struct_def.xml 파일 생성 (페이지 157)" 장을 참조하십시오.
	
	파라미터:
	range - 머신 데이터 영역을 지정하는 문자열을 전송해야 합니다:
	• **displayMD** - 화면 표시 머신 데이터
	• **generalMD** - NC 전역 머신 데이터
	• **channelMD** - NC 채널별 머신 데이터
	• **axisMD** - NC 축별 머신 데이터
	• **generalSettingMD** - NC 전역 설정 데이터
	• **channelSettingMD** - NC 채널별 설정 데이터
	• **axisSettingMD** - NC 축별 설정 데이터
	• **channelGUD** - NC 채널별 사용자 데이터
	• **globalGUD** - NC 전역 사용자 데이터
	MD-number - 영역의 머신 데이터 번호
	variable name - 함수 호출 후 이 변수에는 한계 값이 포함됩니다.
	range/unit - 선택 사항(예: 채널 번호)
	
	구문:
	`<function name="NCFUNC.GETMD_LIMITS"><range>, <MD-number, <variabel name>, <range/unit>></function>`
	
	예:
	`<let name="limits" type="StructMDLimits" />`
	`<function name="NCFUNC.GETMD_LIMITS">_T"generalMD", 19220, _T"limits"</function>`
	`<op>`
	`upper_BAGS = limits.upperLimit;`
	`</op>`

함수 이름	의미

Ncfunc bico to int	BICO 형식으로 지정된 문자열을 정수 값으로 변환하는 함수입니다(SINAMICS 참조).
	
	구문:
	`<function name="ncfunc.bicotoint" return="integer variable">`
	`bico-string</function>`
	
	예:
	`<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	정수 값을 BICO 형식 문자열로 변환하는 함수입니다(SINAMICS 참조).
	
	구문:
	`<function name="ncfunc.inttobico"`
	`return="string variable">integer variable</function>`
	
	예:
	`<function name="ncfunc.inttobico"`
	`return="s_p0480_0">"drive/dc/p0480[0, DO2]"</function>`

함수 이름	의미

Ncfunc is bico str valid	BICO 형식으로 지정된 문자열을 위해 값 0을 반환하는 함수입니다(SINAMICS 참조).
	
	구문:
	`<function name="ncfunc.isbicostrvalid" return="integer variable">string varaible</function>`
	
	예:
	`…`
	`<let name="s_np0480_0" type="string"></let>`
	`…`
	`…`
	`<control name = "cp0480_0" xpos = "402" ypos = "76" hotlink="true" refvar="s_np0480_0" >`
	`<property item_data="4001" />`
	`</control>`
	`…`
	`…`
	`<function name="ncfunc.isbicostrvalid" return="valid">cp0480_0</function>`

Ncfunc password	NC에서 암호 레벨을 설정 또는 삭제하는 기능입니다.
	• 암호 설정:
	필요한 암호 레벨에 암호를 파라미터로 지정해야 합니다.
	• 암호 삭제:
	문자열이 비어 있으면 암호 레벨을 삭제합니다.
	
	구문:
	`<function name="ncfunc.password">암호 </function>`
	
	예:
	`<let name="암호"type="string"></let>`
	
	`<function name="ncfunc.password" > 암호 </function>`
	`<function name="ncfunc.password" > _T"CUSTOMER" </function>`
	
	암호 삭제:
	`<function name="ncfunc.password" > _T"" </function>`

함수 이름	의미

Control form color	이 함수는 대화 상자의 텍스트 또는 배경 색상을 문자열로 반환합니다.
	자세한 정보는 "색 코딩 (페이지 82)" 장을 참조하십시오.
	
	범위:
	• BACKGROUND – 배경 색상 값 요청
	• TEXT – 텍스트 (전경) 색상 값 요청
	
	구문:
	`<function name="control.formcolor" return="variable">_T"Area"</function>`
	
	예:
	`<let name="bk_color" type="string"></let>`
	
	`<function name="control.formcolor" return="bk_color">_T"BACKGROUND"</function>`

함수 이름	의미

Control local time	7개의 배열 요소가 있는 필드에 로컬 시간을 복사하는 기능입니다.
	변수의 이름이 호출 파라미터로 인식됩니다.
	각 배열 요소에는 다음 정보가 저장됩니다.
	• 인덱스 0 - 년
	• 인덱스 1 - 월
	• 인덱스 2 - 요일
	• 인덱스 3 - 날짜
	• 인덱스 4 - 시
	• 인덱스 5 - 분
	• 인덱스 6 - 초
	
	구문:
	`<function name ="control.localtime">_T"time_array"`
	`</function>`
	
	예:
	`<!-- 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	이 함수는 지정된 컨트롤이 존재하면 값 1을 반환합니다.
	
	구문:
	`<function name="CONTROL.EXIST"`
	`return="<return variable>"><control name></function>`
	
	예:
	`<let name="ret" /> <function name="CONTROL.EXIST"`
	`return="ret">_T"edit"</function>`

함수 이름	의미

Control is modified	이 함수는 지정된 편집 컨트롤의 내용이 변경되면 값 1을 반환합니다.
	
	구문:
	`<function name="CONTROL.ISMODIFIED"`
	`return="<return variable>"><control name></function>`
	
	예:
	`<let name="ret" /> <function name="CONTROL.ISMODIFIED"`
	`return="ret">_T"edit"</function>`

Control is visible	이 함수는 지정된 컨트롤이 보이면 값 1을 반환합니다.
	
	구문:
	`<function name="CONTROL.ISVISIBLE"`
	`return="<return variable>"><control name></function>`
	
	예:
	`<let name="ret" /><function name="CONTROL.ISVISIBLE"`
	`return="ret">_T"edit"</function>`

Control set modified	이 함수는 지정된 편집 컨트롤의 수정된 플래그를 변경합니다.
	
	파라미터:
	control name
	• 값 = 1 - 필드를 변경됨으로 표시
	• 값 = 0 - 필드를 변경되지 않음으로 표시
	
	구문:
	`<function name="CONTROL.SETMODIFIED"`
	`return="<return variable>"><control name>,`
	`<Flag></function>`
	
	예:
	`<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>`

함수 이름	의미

Control set working area	컨트롤을 삭제하거나 추가해 **스크롤 보기**의 내용을 변경하면 스크롤 보기의 표시 영역을 다시 계산해야 합니다.
	계산은 이 함수를 호출해서 수행합니다.
	
	값:
	스크롤 보기의 이름
	
	예:
	`...`
	`<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>`
	`...`

함수 이름	의미

문자열 비교	사전적 관점에서 두 문자열을 서로 비교합니다.
	문자열이 동일할 경우 값 0, 첫 번째 문자열이 두 번째 문자열보다 적을 경우 0보다 작은 값, 두 번째 문자열이 첫 번째 문자열보다 적을 경우 0보다 큰 값을 반환하는 기능입니다.
	파라미터:
	str1 - 문자열
	str2 - 비교 문자열
	
	구문:
	`<function name="string.cmp" return ="<int var>" > str1, str2 </function>`
	
	예:
	`<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>`
	결과:
	rval=0

함수 이름	의미

String to compare without making a distinction between uppercase/lowercase	대소문자 구분 없이 사전적 관점에서 두 문자열을 비교합니다.
	문자열이 동일할 경우 값 0, 첫 번째 문자열이 두 번째 문자열보다 적을 경우 0보다 작은 값, 두 번째 문자열이 첫 번째 문자열보다 적을 경우 0보다 큰 값을 반환하는 기능입니다.
	
	파라미터:
	str1 - 문자열
	str2 - 비교 문자열
	
	구문:
	`<function name="string.icmp" return ="<int var>" > str1, str2 </function>`
	
	예:
	`<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>`
	결과:
	rval=0

함수 이름	의미

문자열 왼쪽	문자열 1에서 첫 번째 nCount 문자를 추출하여 이것을 리턴 변수에 복사합니다.
	
	파라미터:
	str1 - 문자열
	nCount - 문자 개수
	
	구문:
	`<function name="string.left" return="<result string>"> str1,  nCount </function>`
	
	예:
	`<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>`
	결과:
	str2="A brown bear"

문자열 오른쪽	문자열 1에서 마지막 nCount 문자를 추출하여 이것을 리턴 변수에 복사합니다.
	
	파라미터:
	str1 - 문자열
	nCount - 문자 개수
	
	구문:
	`<function name="string.right" return="<result string>"> str1, nCount </function>`
	
	예:
	`<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>`
	결과:
	str2="brown dog."

함수 이름	의미

문자열 중앙	iFirst 인덱스를 시작으로 문자열 1에서 지정된 문자 개수를 추출하여 이것을 리턴 변수에 복사합니다.
	
	파라미터:
	str1 - 문자열
	iFirst - 시작 인덱스
	nCount - 문자 개수
	
	구문:
	`<function name="string.middle"`
	`return="<result string>"> str1, iFirst, nCount </function>`
	
	예:
	`<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>`
	결과:
	str2="brown"

문자열 길이	문자열의 문자 개수를 제공합니다.
	
	파라미터:
	str1 - 문자열
	
	구문:
	`<function name="string.length" return="<int var>"> str1 </function>`
	
	예:
	`<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>`
	결과:
	length =31

함수 이름	의미

문자열 바꾸기	찾은 모든 서브 문자열을 새 문자열로 교체합니다.
	
	파라미터:
	string - 문자열 변수
	find string - 교체할 문자열
	new string - 새 문자열
	
	구문:
	`<function name="string.replace"> string, find string, new string </function>`
	
	예:
	`<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>`
	결과:
	str1 = "A brown bear hunts a big salmon!"

문자열 제거	찾은 모든 서브 문자열을 제거합니다.
	
	파라미터:
	string - 문자열 변수
	remove string - 삭제할 하위 문자열
	
	구문:
	`<function name="string.remove"> string, remove string </function>`
	
	예:
	`<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>`
	결과:
	str1 = "A brown bear hunts"

함수 이름	의미

문자열 삽입	지정된 인덱스에 문자열을 삽입합니다.
	
	파라미터:
	string - 문자열 변수
	index - 인덱스(0부터 시작)
	insert string - 삽입할 문자열
	
	구문:
	`<function name="string.insert"> string, index, insert string`
	`</function>`
	
	예:
	`<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>`
	결과:
	str1 = "A brown bear hunts a brown dog"

문자열 삭제	지정된 시작 위치에서부터 정의된 수의 문자를 삭제합니다.
	
	파라미터:
	string - 문자열 변수
	start index - 시작 인덱스(0부터 시작)
	nCount - 제거할 문자 개수
	
	구문:
	`<function name="string.delete"> string, start index , nCount`
	`</function>`
	
	예:
	`<let name="str1" type="string">A brown bear hunts. </let>`
	
	`<function name="string.delete" > str1, 2, 5`
	`</function>`
	결과:
	str1 = "A bear hunts"

함수 이름	의미

문자열 찾기	전송된 문자열에서 서브 문자열과 처음으로 일치되는 문자열을 탐색하는 기능입니다.
	서브 문자열을 찾으면 최초 문자에 제로에서 시작하는 인덱스를 제공합니다. 찾지 못할 경우에는 -1을 제공합니다.
	
	파라미터:
	string - 문자열 변수
	findstring - 검색할 문자열
	startindex – 시작 인덱스(옵션)
	
	구문:
	`<function name="string.find" return="<int val>"> str1, find string </function>`
	
	예:
	`<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>`
	
	결과:
	Index = 2
	
	또는
	`<function name="string.find" return="index"> str1, _T"brown", 1 </function>`

함수 이름	의미

역방향 문자열 찾기	서브 스트링과 마지막으로 일치되는 문자열을 탐색하는 기능입니다.
	서브 문자열을 찾으면 최초 문자에 제로에서 시작하는 인덱스를 제공합니다. 찾지 못할 경우에는 -1을 제공합니다.
	
	파라미터:
	string - 문자열 변수
	find string - 검색할 문자열
	startindex – 시작 인덱스(옵션)
	
	구문:
	`<function name="string.reversefind"`
	`return="<int val>"> str1, find string </function>`
	
	예:
	`<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>`
	
	결과:
	Index = 21
	
	또는
	`<function name="string.reversefind"`
	`return="index"> str1, _T"brown" 10 </function>`
	
	결과:
	Index = 2

함수 이름	의미

String GetAt	이 함수는 지정 위치에서 문자를 읽습니다.
	파라미터:
	str - 문자열
	index - 판독되는 문자의 인덱스(0부터 시작)
	
	반환값:
	문자가 저장되는 문자열 변수의 이름을 지정해야 합니다.
	
	구문:
	`<function name="string.getat" return="<result string>"><string>, <index></function>`
	
	예:
	`<let name="resStr" type="string" />`
	`<function name="string.getat" return="resStr">_T"brown", 2</function>`

함수 이름	의미

String pixel height	문자열의 높이를 픽셀로 계산. 계산은 지정된 컨트롤 또는 양식의 현재 글꼴을 사용해 수행됩니다. 컨트롤의 이름은 문자열로 전송되어야 합니다. 빈 문자열이 지정되면 계산은 양식을 참조합니다.
	
	파라미터:
	control name
	문자열
	문자열을 직접 텍스트로 지정할 수 있거나 문자열 변수를 지정해야 합니다.
	
	반환:
	높이가 기록된 정수 변수의 이름이 예상됩니다.
	
	구문:
	`<function name="STRING.PIXELHEIGHT" return="<return variable>"><control name>, <varaible or text></function>`
	
	예:
	양식의 글꼴과 관련된 너비 계산
	`<let name="pixelsh" />`
	`...`
	`<function name="STRING.PIXELHEIGHT" return="pixelsh">_T"", cedit1</function>`
	
	컨트롤의 글꼴과 관련된 너비 계산
	`<function name="STRING. PIXELHEIGHT" return="pixelsh"> cedit1, cedit1</function>`

함수 이름	의미

String pixel width	문자열의 너비를 픽셀로 계산. 계산은 지정된 컨트롤 또는 양식의 현재 글꼴을 사용해 수행됩니다. 컨트롤의 이름은 문자열로 전송되어야 합니다. 빈 문자열이 지정되면 계산은 양식을 참조합니다.
	
	파라미터:
	control name
	문자열
	문자열을 직접 텍스트로 지정할 수 있거나 문자열 변수를 지정해야 합니다.
	
	반환:
	텍스트 너비가 기록된 정수 변수의 이름이 예상됩니다.
	
	구문:
	`<function name="STRING.PIXELWIDTH"`
	`return="<return variable>"><control name>,`
	`<varaible or text></function>`
	
	예:
	양식의 글꼴과 관련된 너비 계산
	`<let name="pixels" />`
	`...`
	`<function name="STRING. PIXELWIDTH"`
	`return="pixels">_T"", cedit1</function>`
	
	컨트롤의 글꼴과 관련된 너비 계산
	`<function name="STRING.PIXELWIDTH"`
	`return="pixels"> cedit1, cedit1</function>`

함수 이름	의미

String SetAt	이 함수는 지정 위치에 문자를 작성합니다. 인덱스는 최대 텍스트 길이보다 작아야 합니다.
	
	파라미터:
	str - 문자열
	char - 지정 위치에 작성할 문자
	index - 대상 변수의 문자열에 대한 인덱스(0부터 시작)
	
	구문:
	`<function name="string.setat" ><destination string>, <character string>, <index></function>`
	
	예:
	`<let name="str" type="string" >br_wn</string>`
	`<function name="string.setat">str, ">_T"o", 2 </function>`

함수 이름	의미

String Split	이 함수는 문자열을 여러 개의 하위 문자열로 나누어 지정된 문자열 배열에 복사합니다. 문자열 분리는 지정된 구분자에서 이뤄집니다. 구분자는 하위 문자열에 저장되지 않습니다.
	이 함수는 구분자의 개수가 지정된 배열 크기보다 크면 배열을 자동으로 확장합니다.
	
	파라미터:
	str - 문자열
	char - 구분자를 포함하는 변수의 이름
	number - 이 함수 실행 후 생성된 하위 문자열 개수를 포함하는 변수 이름
	
	반환값:
	함수 실행 후 하위 문자열을 포함하는 문자열 배열 이름을 지정해야 합니다.
	
	구문:
	`<function name=" string.split"`
	`return="<result string array>"><string>, <char>,`
	`<number></function>`
	
	예:
	`<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>`

함수 이름	의미

문자열 왼쪽 트림	문자열에서 시작 문자를 트림하는 기능입니다.
	
	파라미터:
	str1 - 문자열 변수
	
	구문:
	`<function name="string.trimleft" > str1 </function>`
	
	예:
	`<let name="str1" type="string">test trim left</let>`
	`<function name="string.trimleft" > str1 </function>`
	결과:
	str1 = "test trim left"

문자열 오른쪽 트림	문자열에서 마지막 문자를 트림하는 기능입니다.
	
	파라미터:
	str1 - 문자열 변수
	
	구문:
	`<function name="string.trimright" > str1 </function>`
	
	예:
	`<let name="str1" type="string"> test trim right </let>`
	`<function name="string.trimright" > str1`
	`</function>`
	결과:
	str1 = "test trim right"

함수 이름	의미

사인	각도로 전송된 값의 사인을 계산하는 기능입니다.
	
	파라미터:
	double - 각도
	
	구문:
	`<function name="sin" return="<double val>"> double </function>`
	
	예:
	`<let name= "sin_val" type="double"></let>`
	`<function name="sin" return="sin_val"> 20.0`
	`</function>`

코사인	각도로 전송된 값의 코사인을 계산하는 기능입니다.
	
	파라미터:
	double - 각도
	
	구문:
	`<function name="cos" return="<double val>"> double </function>`
	
	예:
	`<let name= "cos_val" type="double"></let>`
	`<function name="cos" return="cos_val"> 20.0`
	`</function>`

탄젠트	각도로 전송된 값의 탄젠트를 계산하는 기능입니다.
	
	파라미터:
	double - 각도
	
	구문:
	`<function name="tan" return="<double val>"> double </function>`
	
	예:
	`<let name= "tan_val" type="double"></let>`
	`<function name="tan" return="tan_val"> 20.0`
	`</function>`

함수 이름	의미
ARCSIN	각도로 전송된 값의 아크사인을 계산하는 기능입니다. **파라미터:** **double** - -PI/2~+PI/2 범위의 x **구문:** <function name="arcsin" return="<double val>"> double </function> **예:** <let name= "arcsin_val" type="double"></let> <function name="arcsin" return="arcsin_val"> 20.0 </function>
ARCOS	각도로 전송된 값의 아크코사인을 계산하는 기능입니다. **파라미터:** **double -** -PI/2~+PI/2 범위의 x **구문:** <function name="arcos" return="<double val>"> double </function> **예:** <let name= "arccos_val" type="double"></let> <function name="arccos" return="arccos_val"> 20.0 </function>
ARCTAN	각도로 전송된 값의 아크탄젠트를 계산하는 기능입니다. **파라미터:** **double** - y/x의 아크탄젠트 **구문:** <function name="arctan" return="<double val>"> double </function> **예:** <let name= "arctan_val" type="double"></let> <function name="arctan" return="arctan_val"> 20.0 </function>
파일 처리	

함수 이름	의미

파일 읽기	지정된 파일의 내용을 문자열 변수로 읽어 들이는 함수입니다.
	읽어 들일 문자 수를 두 번째 파라미터로 지정할 수 있습니다.
	
	속성:
	name - 파일 이름은 소문자로 작성해야 합니다.
	다른 디렉토리의 파일들은 appl 또는 dvm 디렉토리를 시작 지점으로 사용하는 해당 경로를 통해 액세스됩니다.
	return - 로컬 변수의 이름
	
	파라미터:
	progname - 파일 이름
	number of characters - 읽어 들일 문자 수(바이트)(옵션)
	
	구문:
	`<function name="doc.readfromfile"`
	`return="<string var>"> progname, number of characters`
	`</function>`
	
	예:
	`<let name = "my_var" type="string" ></let>`
	
	NC 파일 시스템
	`<function name="doc.readfromfile"`
	`return="my_var"> _T"n:\mpf\test.mpf"`
	`</function>`
	
	콤팩트 플래시 카드
	`<function name="doc.readfromfile"`
	`return="my_var"> _T"f:\appl\test.mpf"`
	`</function>`
	
	또는
	
	`<function name="doc.readfromfile"`
	`return="my_var"> _T".\test.mpf"`
	`</function>`

함수 이름	의미

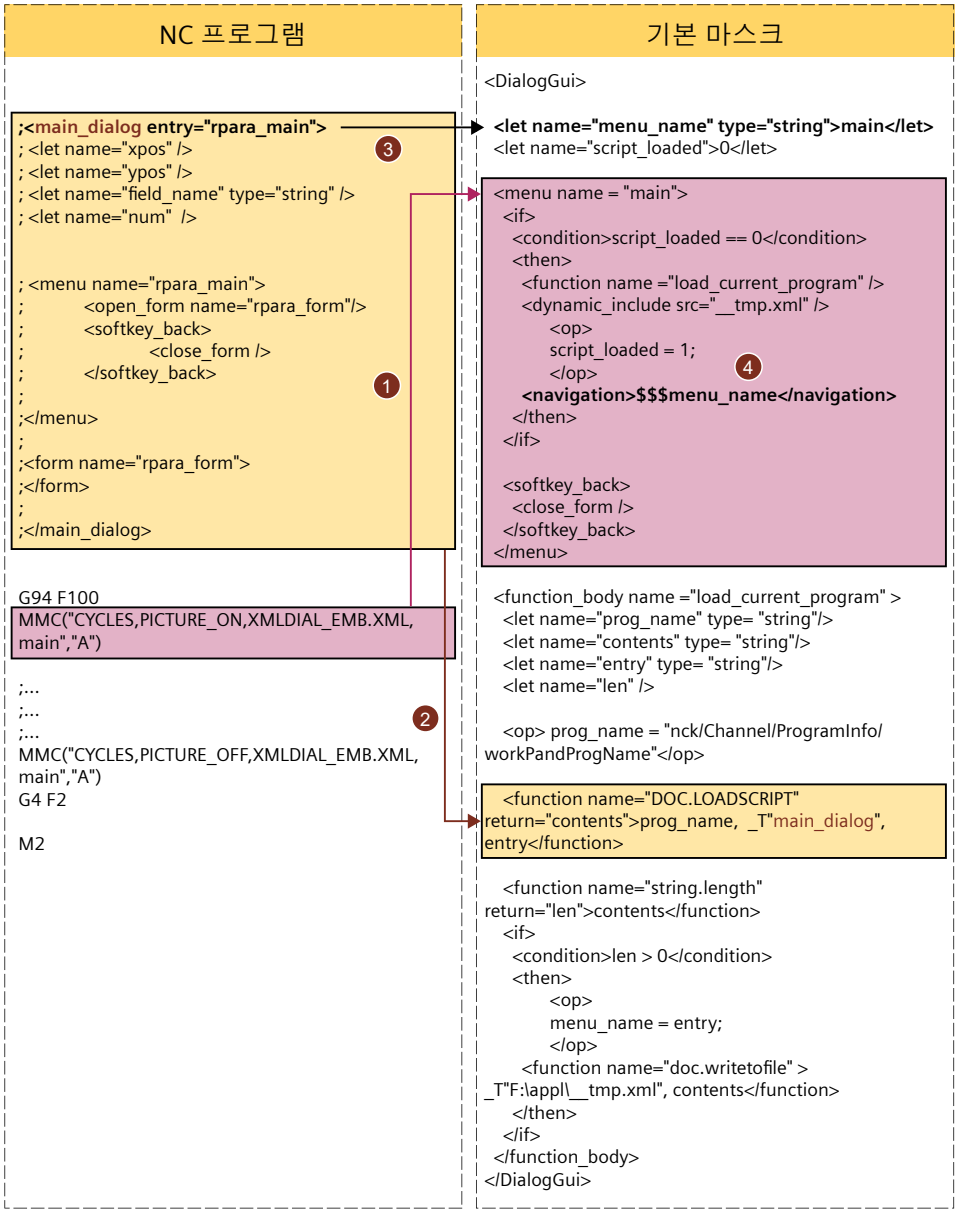
파일에 쓰기	지정된 파일에 문자열 변수의 목차를 쓰는 기능입니다.
	파일 이름은 소문자로 작성해야 합니다.
	다른 디렉토리의 파일들은 appl 또는 dvm 디렉토리를 시작 지점으로 사용하는 해당 경로를 통해 액세스됩니다.
	
	파라미터:
	progname - 파일 이름
	str1 - 문자열
	
	구문:
	`<function name="doc.writetofile" > progname, str1 </function>`
	
	예:
	`<let name = "my_var" type="string" > file content </let>`
	
	NC 파일 시스템
	`<function name="doc.writetofile">_T"n:\mpf\test.mpf", my_var </function>`
	
	콤팩트 플래시 카드
	`<function name="doc.writetofile">_T"f:\appl\test.mpf", my_var </function>`
	
	또는
	
	`<function name="doc.writetofile">_T".\test.mpf", my_var </function>`

함수 이름	의미

파일 삭제	디렉토리에서 지정된 파일을 제거하는 기능입니다.
	파일 이름은 소문자로 작성해야 합니다.
	다른 디렉토리의 파일들은 appl 또는 dvm 디렉토리를 시작 지점으로 사용하는 해당 경로를 통해 액세스됩니다.
	
	파라미터:
	progname - 파일 이름
	
	구문:
	`<function name="doc.remove" > progname </function>`
	
	예:
	NC 파일 시스템
	`<function name="doc.remove">_T"n:\mpf\test.mpf"`
	`</function>`
	
	콤팩트 플래시 카드
	`<function name="doc.remove">_T"f:\appl\test.mpf"`
	`</function>`
	
	또는
	
	`<function name="doc.remove">_T".\test.mpf"`
	`</function>`

함수 이름	의미

Extracting script parts	가공 프로그램에 포함된 대화 상자 설명을 지정된 로컬 변수로 복사하는 함수입니다.
	
	지정할 호출 파라미터는 프로그램 이름, 대화 상자 이름 및 메인 메뉴 이름을 저장할 변수입니다. 대화 상자 설명의 이름이 가공 프로그램에 있으면 반환 변수에 이 설명이 포함됩니다. 변수의 내용이 파일에 저장되면 스크립트가 간접 호출로 실행될 수 있습니다.
	
	시스템은 활성 가공 프로그램으로부터 대화 상자 설명을 추출하고 대화 상자를 활성화하는 스크립트를 제공합니다. 가공 프로그램에 연결된 화면을 활성화하기 위해 MMC 명령에서 이 스크립트를 호출할 수 있습니다.
	
	구문:
	`<function name="doc.loadscript"`
	`return="<name of script variable>">progname, _T"dialog part name",`
	`main menu </function>`
	
	속성:
	return - 추출된 스크립트가 저장되는 변수
	
	파라미터:
	progname - 프로그램의 전체 경로 (경로 이름은 DOS 표기법으로 함수로 전달될 수 있습니다.)
	main menu - 발견된 메뉴 이름이 이 변수로 복사됩니다.
	dialog part name - 대화 상자 설명이 포함된 태그 이름
	
	예:
	`<function name="doc.loadscript" return="contents">prog_name,`
	`_T"main_dialog", entry</function>`

함수 이름	의미
Extracting script parts 계속	예 다이어그램: 

함수 이름	의미

존재	파일이 존재하면 기능은 값 1을 리턴합니다.
	파일 이름은 소문자로 작성해야 합니다.
	다른 디렉토리의 파일들은 appl 또는 dvm 디렉토리를 시작 지점으로 사용하는 해당 경로를 통해 액세스됩니다.
	
	파라미터:
	progname - 파일 이름
	
	구문:
	`<function name="doc.exist" return="<int_var>" > progname </function>`
	
	예:
	`<let name ="exist">0</let>`
	
	NC 파일 시스템
	`<function name="doc.exist" return="exist">_T"n:\mpf\test.mpf" </function>`
	
	콤팩트 플래시 카드
	`<function name="doc.exist" return="exist">_T"f:\appl\test.mpf" </function>`
	
	또는
	
	`<function name="doc.exist" return="exist">_T".\test.mpf" </function>`

함수 이름	의미

NC 프로그램 선택	실행하도록 지정된 프로그램을 선택하는 기능입니다. 프로그램은 NC 파일 시스템에 저장해야 합니다.
	
	파라미터:
	progname - 파일 이름
	
	구문:
	`<function name="ncfunc.select"> progname </function>`
	
	예:
	NC 파일 시스템
	`<function name="ncfunc.select"> _T"n:\mpf\test.mpf" </function>`

개별 비트 설정	지정된 변수의 개별 비트를 조작할 때 사용하는 기능입니다.
	비트를 설정 또는 리셋할 수 있습니다.
	
	구문:
	`<function name="ncfunc.bitset" refvar="address" value="set/reset" > bit0, bit1, … bit9 </function>`
	
	속성:
	refvar - 비트 조합을 써야 하는 변수의 이름을 지정합니다.
	value – 비트 값 (값 범위: 0 및 1)
	
	값:
	0으로 시작하는 비트 번호를 기능 값으로 전송해야 합니다.
	호출당 최대 10개의 비트를 수정할 수 있습니다.
	
	예:
	`<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>`

함수 이름	의미

컨트롤 삭제	지정된 이미지 컨트롤을 삭제하는 기능입니다.
	
	구문:
	`<function name="control.delete"> control name </function>`
	
	속성:
	name - 기능 이름
	
	값:
	control name - 컨트롤 이름
	
	예:
	`<function name="control.delete"> _T"my_editfield"`
	`</function>`

함수 이름	의미

항목 추가	목록 끝에 새 요소를 삽입하는 기능입니다.
	참고:
	이 기능은 컨트롤 유형이 "listbox" 및 "graphicbox"인 경우에만 사용할 수 있습니다.
	
	구문:
	`<function name="control.additem"> control name, item </function>`
	
	속성:
	name - 기능 이름
	
	값:
	control name - 컨트롤 이름
	item - 삽입할 표현식
	itemdata - 정수 값 (사용자가 정의)
	
	예:
	`<let name ="itemdata">1</let>`
	…
	…
	…
	`<op> item_string = _T"text1" </op>`
	`<function name="control.additem">_T"listbox1", item_string, itemdata </function>`

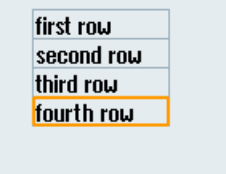
함수 이름	의미

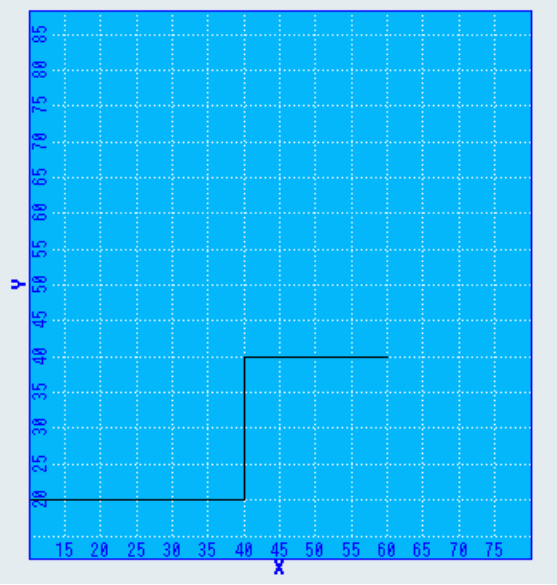
항목 삽입	지정된 위치에 새 요소를 삽입하는 기능입니다.
	참고:
	이 기능은 컨트롤 유형이 "listbox"인 경우에만 사용할 수 있습니다.
	
	구문:
	`<function name="control.insertitem"> control name, index, item, itemdata </function>`
	
	속성:
	name - 기능 이름
	
	값:
	control name - 컨트롤 이름
	index - 위치 (0부터 시작)
	item - 삽입할 표현식
	itemdata - 정수 값 (사용자가 정의)
	
	예:
	`<let name ="itemdata">1</let>`
	…
	…
	…
	`<op> item_string = _T"text2" </op>`
	`<function name="control.insertitem">_T"listbox1", 1, item_string, itemdata </function>`

함수 이름	의미

항목 삭제	지정된 위치에 있는 요소를 삭제하는 기능입니다.
	참고:
	이 기능은 컨트롤 유형이 "listbox"인 경우에만 사용할 수 있습니다.
	
	구문:
	`<function name="control.deleteitem"> control name, index </function>`
	
	속성:
	name - 기능 이름
	
	값:
	control name - 컨트롤 이름
	index - 인덱스 (0부터 시작)
	
	예:
	`<function name="control.deleteitem">_T"listbox1", 1</function>`

함수 이름	의미

항목 로드	컨트롤에 표현식 목록을 삽입하는 기능입니다.
	이 기능은 컨트롤 유형이 "listbox" 및 "graphicbox"인 경우에만 사용할 수 있습니다.
	
	구문:
	`<function name="control.loaditem"> control name, list </function>`
	
	속성:
	name - 기능 이름
	
	값:
	control name - 컨트롤 이름
	list- 문자열 변수
	변수에 포함된 문자열이 아래에 설명된 구조를 따라야 합니다.
	
	목록 구조:
	목록에는 여러 표현식이 포함되어 있습니다. \\n을 사용해 표현식을 서로 구분해야 합니다.
	
	예:
	예에서는 내용이 control.loaditem 함수를 사용해 목록 상자에 할당되었습니다.
	컨트롤 문자 \\n은 라인에 속한 표현식을 구분합니다.
	`<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>`
	 

함수 이름	의미
Load Item 계속	**예:** 예에서는 내용이 control.loaditem 함수를 사용해 그래픽 상자에 할당되었습니다. 컨트롤 문자 \\n은 그래픽 요소를 구분합니다. `<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>` 

함수 이름	의미

비우기	지정된 목록 상자 또는 그래픽 상자 컨트롤의 내용을 삭제하는 기능입니다.
	
	구문:
	`<function name="control.empty"> control name,`
	`</function>`
	
	속성:
	name - 기능 이름
	
	값:
	control name - 컨트롤 이름
	
	예:
	`<function name="control.empty">_T"listbox1"`
	`</function>`

포커스 가져오기	입력 포커스가 있는 컨트롤의 이름을 제공하는 기능입니다.
	
	구문:
	`<function name="control.getfocus" return="focus_name" />`
	
	속성:
	name - 기능 이름
	return – 컨트롤 이름을 복사할 문자열 변수를 지정해야 합니다.
	
	예:
	`<let name>="focus_field" type="string"></let>`
	`<function name="control.getfocus" return="focus_field"/>`

함수 이름	의미

포커스 설정	지정된 컨트롤에 입력 포커스를 설정하는 기능입니다.
	기능에 텍스트 표현식을 사용해 컨트롤 이름을 전송해야 합니다.
	
	구문:
	`<function name="control.setfocus"> control name`
	`</function>`
	
	속성:
	name - 기능 이름
	
	값:
	control name- 컨트롤 이름
	
	예:
	`<function name="control.setfocus" ">_T"listbox1"`
	`</function>`

커서 선택 가져오기	목록 상자에 커서 인덱스를 제공하는 기능입니다.
	기능에 텍스트 표현식을 사용해 컨트롤 이름을 전송해야 합니다.
	
	구문:
	`<function name="control.getcurssel" return="var">control name`
	`</function>`
	
	예:
	`<let name>="index"></let>`
	`<function name="control.getcurssel" return="index">_T"listbox1"`
	`</function>`

함수 이름	의미

커서 선택 설정	목록 상자에서 적절한 라인에 커서를 설정하는 기능입니다.
	기능에 텍스트 표현식을 사용해 컨트롤 이름을 전송해야 합니다.
	
	구문:
	`<function name="control.setcurssel" > control name, index </function>`
	
	예:
	`<let name>="index">2</let>`
	`<function name="control.setcurssel" ">_T"listbox1",index </function>`

항목 가져오기	목록 상자에서 선택한 라인의 내용을 지정된 변수에 복사하는 기능입니다.
	문자열 변수를 기준 변수로 지정해야 합니다.
	기능에 텍스트 표현식을 사용해 컨트롤 이름을 전송해야 합니다.
	
	구문:
	`<function name="control.getitem" return="var"> control name, index </function>`
	
	예:
	`<let name>="index">2</let>`
	`<let name>="item" type="string"></let>`
	
	`<function name="control.getitem" return="item" ">_T"listbox1",index </function>`

항목 데이터 가져오기	목록 상자에서 사용자가 지정한 요소의 값을 지정된 변수에 복사하는 기능입니다.
	편집 컨트롤의 경우 사용자가 지정한 값 (item_data) 을 지정된 변수에 복사합니다.
	정수 변수를 기준 변수로 지정해야 합니다.
	기능에 텍스트 표현식을 사용해 컨트롤 이름을 전송해야 합니다.
	
	구문:
	`<function name="control.getitemdata" return="var"> control name, index </function>`
	
	예:
	`<let name>="index">2</let>`
	`<let name>="itemdata"></let>`
	
	`<function name="control.getitemdata" return="itemdata" ">_T"listbox1",index</function>`

함수 이름	의미
이미지 상자 설정	이 함수는 **이미지 상자** 및 **그래픽 상자** 컨트롤의 표시 영역을 제어하는 데 사용됩니다. 지정할 호출 파라미터는 컨트롤 이름, 컨트롤 명령 및 관련 값입니다. **구문:** `<function name="control.imageboxset">control name, command, command parameter</function>` **호출 파라미터:** • **x_offs** 이 함수는 이미지 일부를 지정된 X 위치로 이동합니다. 지정할 추가 파라미터는 왼쪽 모서리의 좌표입니다. • **y_offs** 이 함수는 이미지 일부를 지정된 Y 위치로 이동합니다. 지정할 추가 파라미터는 위쪽 모서리의 좌표입니다. • **xy_offs** 이 함수는 이미지 일부를 지정된 X/Y 위치로 이동합니다. 지정할 추가 파라미터는 왼쪽 및 위쪽 모서리의 좌표입니다. • **followCursor** 이미지 일부가 설정된 커서 위치를 따릅니다. • **zoomplus** 이미지가 0.12배 확대됩니다. • **zoomminus** 이미지가 0.12배 축소됩니다. • **autozoom** 이미지가 표시 영역에 자동으로 맞춰집니다. • **Update** 컨트롤을 다시 그립니다. • **SetBkColor** 지정된 배경 색상을 설정하는 명령입니다. 지정할 추가 파라미터는 색상 코드입니다. • **SetCursorRect** 직사각형으로 지정된 부분은 표시 영역으로 이동됩니다. • **SetAnimationState** (이미지 상자 컨트롤에만 적용) – **start** - 애니메이션을 시작하는 명령입니다. – **stop** - 애니메이션을 중지하는 명령입니다.

함수 이름	의미
Image Box Set 계속	**예:** ... ... <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>
이미지 상자 가져오기	이 함수는 컨트롤 속성 이미지 상자를 쿼리하는 데 사용됩니다. 지정할 호출 파라미터는 컨트롤 명령 및 관련 값입니다. **구문:** <function name="control.imageboxget">control name, command, command parameter</function> **호출 파라미터:** **GetViewSize** 표시 영역의 크기를 반환합니다. 지정할 추가 파라미터는 구조체 유형 SIZE의 변수입니다. **예:** <let name="view_size" type="size" /> <function name="control.imageboxget">_T"topoview", _T"**GetViewSize**", view_size</function>

함수 이름	의미

비트맵 치수	비트맵 치수를 구조체 유형 SIZE의 변수로 다시 복사하는 함수입니다. 유형을 정의하려면 프로젝트에 struct_def.xml 파일이 포함되어야 합니다.
	자세한 정보는 "struct_def.xml 파일 생성 (페이지 157)" 장을 참조하십시오.
	
	구문:
	`<function name="bitmap.dim" >name, variable type`
	`</function>`
	
	파라미터:
	name - 파일 경로
	variable type - 유형 SIZE 변수의 변수 이름
	
	예:
	`<let name="bmp_size" type="size" />`
	
	`<function name="bitmap.dim" >_T"test.bmp", bmp_size`
	`</function>`

화면 해상도	시스템의 절대 화면 해상도를 구조체 유형 SIZE의 변수로 다시 복사하는 함수입니다. 유형을 정의하려면 프로젝트에 struct_def.xml 파일이 포함되어야 합니다. **자세한 정보**는 "struct_def.xml 파일 생성 (페이지 157)" 장을 참조하십시오. **구문:** `<function name="hmi.screen_resolution" >varialble type </function>`
HMI 해상도 가져오기	SINUMERIK Operate에 의해 사용되는 화면 해상도를 구조체 유형 SIZE의 변수로 다시 복사하는 함수입니다. 유형을 정의하려면 프로젝트에 struct_def.xml 파일이 포함되어야 합니다. **자세한 정보**는 "struct_def.xml 파일 생성 (페이지 157)" 장을 참조하십시오. **구문:** `<function name="hmi.get_hmi_resolution" >variable type </function>`

함수 이름	의미
캡션 높이 가져오기	제목 표시줄 높이를 픽셀 값으로 반환하는 함수입니다. **구문:** `<function name="hmi.get_caption_height" return="<return var>" />` **속성:** **return** - 정수 변수

Get Font Families	이 함수는 설치된 글꼴의 목록을 제공합니다. 글꼴 이름은 LF 문자로 구분되어 표시됩니다. 문자열 변수의 이름이 반환 값으로 전달됩니다.
	
	구문:
	`<function name="HMI.GET_FONT_FAMILIES"`
	`return="< String-Variable>" />`
	
	예:
	`<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	지정된 숫자의 절대값을 반환하는 기능입니다.
	
	구문:
	`<function name="abs" return="var"> value`
	`</function>`

SDEG	지정된 값을 각도로 변환하는 기능입니다.
	
	구문:
	`<function name="sdeg" return="var"> value`
	`</function>`

함수 이름	의미

SRAD	지정된 값을 라디안으로 변환하는 기능입니다.
	
	구문:
	`<function name="srad" return="var"> value`
	`</function>`

SQRT	지정된 값의 제곱근을 계산하는 기능입니다.
	
	구문:
	`<function name="sqrt" return="var"> value`
	`</function>`

ROUND	전송된 숫자를 지정된 소수 자리까지 반올림하는 기능입니다. 소수 자리가 지정되지 않으면 첫 번째 소수 자리에서 숫자를 반올림합니다.
	
	구문:
	`<function name="round" return="var"> value, nDecimalPlaces`
	`</function>`

FLOOR	전송된 값보다 작거나 같은 가장 큰 정수 값을 제공하는 기능입니다.
	
	구문:
	`<function name="floor" return="var"> value`
	`</function>`

CEIL	전송된 값보다 크거나 같은 가장 작은 정수 값을 제공하는 기능입니다.
	
	구문:
	`<function name="ceil" return="var"> value`
	`</function>`

LOG	지정된 값의 로그를 계산하는 기능입니다.
	
	구문:
	`<function name="log" return="var"> value`
	`</function>`

LOG10	지정된 값의 상용 로그 (10을 밑으로 하는 로그) 를 계산하는 기능입니다.
	
	구문:
	`<function name="log10" return="var"> value`
	`</function>`

함수 이름	의미
POW	"a^b" 값을 계산하는 기능입니다. **구문:** `<function name="pow" return="var"> a, b </function>`
MIN	전송된 값과 비교하여 두 값 중 더 작은 값을 반환하는 기능입니다. **구문:** `<function name="min" return="var"> value1, value2 </function>`
MAX	전송된 값과 비교하여 두 값 중 더 큰 값을 반환하는 기능입니다. **구문:** `<function name="max" return="var"> value1, value2 </function>`
RANDOM	의사 난수 (pseudo random number) 를 반환하는 기능입니다. **구문:** `<function name="random" return="var" </function>`

함수 이름	의미
MMC	**함수**: SINUMERIK Operate에서 가공 프로그램으로부터 사용자 정의 대화 상자 양식을 표시하기 위해 MMC 명령을 사용할 수 있습니다. 대화 상자 양식의 모양은 순수한 텍스트 구성으로 정의됩니다(제조사 디렉토리의 XML 파일). 시스템 소프트웨어는 변경되지 않습니다. 운영 기반 시스템의 변경 때문에 파라미터는 다음과 같습니다. XML → CYCLES 또는 POPUPDLG XML_ON → PICTURE_ON XML_OFF → PICTURE_OFF **구문:** MMC("조작 영역, 명령, XML 스크립트 파일, 메뉴 이름, 예비, 예비, 디스플레이 시간 또는 확인 변수, 예비", "확인 모드") **설명:** • MMC SINUMERIK Operate에서 가공 프로그램으로부터 대화형으로 대화 상자 양식 호출. • CYCLES 또는 POPUPDLG 가공 프로그램에서 시작할 사용자 대화 상자의 식별자. • PICTURE_ON 또는 PICTURE_OFF 명령: 대화 상자 선택 또는 선택 해제: • XML 파일 XML 스크립트 파일의 이름 • 메뉴 대화 상자 표시를 관리하는 메뉴 태그의 이름. • 예비용 SINUMERIK Operate용으로 예비 • 예비용 SINUMERIK Operate용으로 예비 • 시간 확인 모드 "N"을 위한 대화 상자의 표시 시간. • 예비용 SINUMERIK Operate용으로 예비 • 확인 모드 "S" 동기. "OK" 소프트 키를 이용한 확인 "N" 비동기. 설정 시간이 지나면 대화 상자가 닫힘

함수 이름	의미

MMC 계속	**동기 호출의 예:**
	운영 기반 시스템의 변경 때문에 파라미터는 다음과 같습니다.
	XML → CYCLES 또는 POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	NC 명령
	`MMC("POPUPDLG or CYCLES",PICTURE_ON,mmc_cmd.xml,cmd1,,,,,","S")`
	파일: 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>`
	
	비동기 호출의 예 (확인 불필요):
	운영 기반 시스템의 변경 때문에 파라미터는 다음과 같습니다.
	XML → CYCLES 또는 POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	NC 명령
	`MMC("POPUPDLG or CYCLES",PICTURE_ON,mmc_cmd.xml,cmd1,,,10,,","N")`
	
	파일: 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>`

함수 이름	의미
	</paint> </form>

함수 이름	의미

MMC 계속	**가공 프로그램으로부터 스크립트 일부를 추출하는 예:**
	운영 기반 시스템의 변경 때문에 파라미터는 다음과 같습니다.
	XML → CYCLES 또는 POPUPDLG
	XML_ON → PICTURE_ON
	XML_OFF → PICTURE_OFF
	NC 명령
	`MMC("POPUPDLG, PICTURE_ON, XMLDIAL_EMB.XML,main","S")`
	`파일: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>`

함수 이름	의미
	</if> </function_body> </DialogGui>

함수 이름	의미
MMC 계속	프로그래밍 예 ;<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>

함수 이름	의미
	; <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	이 함수는 문자열의 내용이 숫자로 평가될 수 있는지 여부를 확인합니다. 공백과 탭은 무시됩니다. 부호와 소수점은 계속 유효한 문자로 간주됩니다. 조건이 충족되면 함수는 값 1을 반환합니다. **파라미터:** **string** - 문자열 **구문:** <function name="isnumeric" return="result"><string> </function>
ROOT	이 함수는 숫자에서 n번째 근을 추출합니다. **파라미터:** **value** - 숫자 **n** - 근의 지수 **구문:** <function name="ROOT" return="result"><value>, <n></function>

3.12 멀티터치 조작

3.12.1 멀티터치 기능

개요

멀티터치 디스플레이를 적용할 때 디스플레이의 확장된 기능성에 맞춰 작동을 조절해야 합니다. 예를 들어 소프트 키의 고정 위치가 없어지고 버튼의 자유로운 위치 지정으로 대체되거나 보완됩니다. 버튼 설계 옵션의 다양성으로 인해, 외관이 전적으로 운영 소프트웨어의 설계 사양을 기준으로 하는 프로그래밍에 단지 하나의 컨트롤만을 사용하는 것은 더 이상 합리적이지 않습니다. 예를 들어 2가지 상태만 아는 토글 컨트롤은 아이콘으로 각 상태를 알려주며 탭이나 플릭 동작에 반응하는 스위치로 대체할 수 있습니다.

디스플레이 그래픽에 이동 동작에 반응하는 기능을 제공할 수 있습니다. **imagebox** 컨트롤은 이 목적을 위해 확장되었습니다.

참고

paint 태그에 출력되는 그래픽은 동작에 반응하지 않습니다.

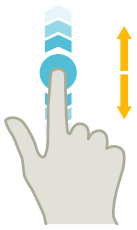
파서에서 제공하는 컨트롤 바깥 쪽에서 손가락 동작을 스크립트에서 처리할 수 있으므로 대화 상자에 새로운 탐색 전략을 제공하는 등의 옵션을 제공할 수 있습니다. 손가락 동작은 대화 상자 내용의 확대 및 축소에 사용될 수 있습니다.

Easy XML 파서는 다음 손가락 동작을 지원합니다.

손가락 동작

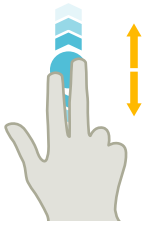
탭

- 창 선택
- 오브젝트 선택(예: NC 세트)
- 입력 필드 활성화
  - 값 입력 또는 덮어쓰기
  - 값을 변경하려면 다시 탭합니다.



한 손가락으로 세로로 빠르게 터치

- 목록에서 스크롤(예: 프로그램, 도구, 제로 포인트)
- 파일에서 스크롤(예: NC 프로그램)



두 손가락으로 세로로 빠르게 터치

- 목록에서 페이지 스크롤(예: ZO)
- 파일에서 페이지 스크롤(예: NC 프로그램)

세 손가락으로 세로로 빠르게 터치

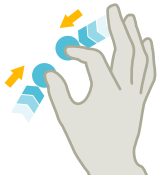
- 목록의 처음이나 끝으로 스크롤
- 파일의 처음이나 끝으로 스크롤

한 손가락으로 가로로 빠르게 터치

- 많은 열이 있는 목록에서 스크롤

스프레드

- 그래픽 콘텐츠 확대(예: 시뮬레이션, 금형 가공 보기)



손가락 좁히기/넓히기

- 그래픽 콘텐츠 축소(예: 시뮬레이션, 금형 가공 보기)

한 손가락으로 누르고 이동

- 그래픽 콘텐츠 이동(예: 시뮬레이션, 금형 가공 보기)
- 목록 콘텐츠 이동

gesture_event 태그와 **$gestureinfo** 변수는 손가락 동작 처리에 사용됩니다. 디코딩된 손가락 동작에 대한 프로그램 동작을 가능하게 합니다.

3.12.2 손가락 동작 프로그래밍

gesture_event tag

참고

이 태그는 화면 조작반이 멀티터치 동작을 지원하는 경우에만 실행됩니다.

운영 소프트웨어가 손가락 동작을 인식하면 파서가 **$gestureinfo** 변수로 동작 정보를 제공하고 **gesture_event** 태그를 실행합니다. 이 변수는 **GestureInfoStruct** 데이터 유형을 가지고 있고 다음 속성을 포함합니다.

속성	값	의미
type	3	이동 동작
	4	축소 동작
	5	탭 동작
flag	1	배율 계수 변경됨
	2	회전 각도 변경됨
state	1	시작됨
	2	업데이트됨
	3	완료
item_data		활성 컨트롤 식별
	-1	컨트롤에 동작이 지정되지 않음
	!= -1	동작은 생성되는 동안 이 값이 지정된 컨트롤로 실행됩니다.

속성	값	의미
지점		동작 위치
	point.x	동작의 X 위치
	point.y	동작의 Y 위치
delta		동작의 시작과 현재 이벤트 간 차이
	<배율 차이>	배율 계수 변경 시
	<각도 차이>	회전 각도 변경 시

속성은 **struct_def.xml** 파일에 사전 정의됩니다.

자세한 정보는 "struct_def.xml 파일 생성 (페이지 157)" 장을 참조하십시오.

3.12.3 그래픽의 동작 컨트롤

Imagebox 컨트롤 변수

다음 3가지 속성은 **Imagebox** 컨트롤 변수 사용 시 그래픽의 동작 컨트롤에 사용할 수 있습니다.

속성	의미/동작
rotationangle	이 속성 값은 그래픽이 표시될 각도를 나타냅니다.
setrotationmode	속성 값이 **true** 면 손가락 좁히기/넓히기 동작을 처리할 수 있습니다.
setzoommode	속성 값이 **true** 면 디스플레이 영역에서 그래픽을 이동하거나 확대 및 축소할 수 있습니다. 이 속성의 기본 값은 **false** 입니다.

구문

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

비트맵 회전 예

비트맵을 시계 방향으로 30.5도 회전:

```
<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>
```

비트맵 크기 조절 예

비트맵 크기 조절 허용:

```
<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 컨트롤 함수

속성은 **control.imageboxset** 함수를 이용해서 설정할 수도 있습니다.

다음 명령을 사용할 수 있습니다.

명령	의미/동작
SetRotationAngle	그래픽이 **lparam1** 에 표시된 각도로 회전합니다.
SetRotationMode	**lparam1** 값은 회전과 관련한 손가락 좁히기/넓히기 동작 파악을 정의합니다. 값이 1이면 동작이 컨트롤에 의해 처리됩니다.
SetZoomMode	**lparam1** 값이 1이면 그래픽 이동 이나 확대 축소를 위한 손가락 좁히기/넓히기 동작을 파악합니다.

3.12.4 동작 처리

태그 메시지

배율 계수가 1.0보다 크면 디스플레이 영역의 이미지가 이동합니다. 계수가 1.0이면 손가락 동작을 이미지 등의 목록 탐색에 사용할 수 있습니다. 이 경우 파서는 **메시지** 태그에서 평가될 수 있는 양식으로 메시지를 전송합니다.

$message_par1 메시지 파라미터는 컨트롤의 Item_Data 값을 포함합니다.
$message_par2 메시지 파라미터는 동작의 이동 방향을 알립니다.

$message_par2 는 다음 값을 허용할 수 있습니다.

- -1 왼쪽 롤링
- 1 오른쪽 롤링
- -2 위로 롤링
- 2 아래로 롤링

예

```
…
…
<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 왼쪽 롤링
1 오른쪽 롤링
-2 위로 롤링
2 아래로 롤링
```

```
 -->
<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 사용자 지정 버튼 구성

버튼은 동작 버튼이나 선택 버튼으로서 양식에 통합될 수 있습니다.

3.13.1 푸시버튼

태그 속성

푸시버튼은 버튼이나 스위치(래칭 기능이 있는 버튼)로 사용할 수 있는 컨트롤 요소입니다. 이 버튼은 "softkey look & feel" 스타일과 사용자 지정 스타일에서의 속성 지정을 통해 설계할 수 있습니다. 각 속성은 **property** 태그를 이용해 지정해야 합니다.

color_fg, **color_bk**, **color_fg_pressed** 및 **color_bk_pressed** 속성을 이용해 전경 또는 배경색을 변경할 수 있습니다.

누른/잠긴 상태나 **해제한** 상태는 1이나 0의 값으로 표현됩니다. 버튼의 상태 변화를 파악하기 위해 핸들러 기능이 표시되어야 합니다. 핸들러 기능은 컨트롤 태그의 **기능** 속성으로 표시됩니다.

스위치의 상태는 컨트롤 변수나 지정된 기준 변수를 판독하여 파악할 수 있습니다.

터치 조작에서는 손가락 동작이 "누른" 상태가 되었다가 "놓음"이 되었을 때만 "해제"됩니다.

구문

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

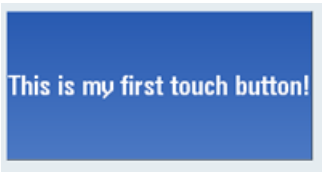
또는 핸들러 기능이 있는 경우

```
<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>
```

또는

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

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

비활성화됨



활성화됨, 래칭됨

예

```
<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 푸시버튼 기능

3.13.2.1 푸시버튼용 하위 태그

태그 캡션

프로그래머는 **caption** 태그를 이용하여 "안 누른" 상태일 때 버튼 텍스트가 표시되도록 지정합니다. 텍스트는 기본 설정으로 중앙에 정렬됩니다. 텍스트 정렬은 **alignment** 속성을 이용해 변경할 수 있습니다. 다음 속성 값을 지정할 수 있습니다.

속성 값	정렬
left	왼쪽 정렬
right	오른쪽 정렬
top	위로 정렬
bottom	아래로 정렬
center	가운데 정렬

"누른" 상태에 다른 텍스트를 표시하려면 **pressed** 속성과 **true** 값을 이용해 두 번째 캡션 지침을 프로그래밍해야 합니다.

구문

중앙 정렬 표시

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

왼쪽 정렬 표시

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

누른 상태 표시

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

</control>
```

3.13.2.2 푸시버튼 속성

property 태그를 이용해 컨트롤에 추가 속성을 지정합니다.

버튼 속성 결정

checkable 속성을 통해 버튼을 스위치로 사용할 수 있습니다. 그러려면 이 속성의 값이 **true** 로 설정되어야 합니다.

구문

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

스위치 비활성화

disabled 속성은 버튼의 조작 가능 여부를 제어합니다. 값이 **true** 이면 컨트롤 동작이 처리되지 않습니다.

지정된 기준 변수에 의한 상태 변화로 버튼이 업데이트됩니다.

구문

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

예

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

아이콘 지정

사용자 지정 아이콘 또는 색상을 지정하여 사전 지정된 디자인을 덮어쓸 수 있습니다. "안 누른", "누른" 및 "비활성화" 상태 별로 버튼에 아이콘을 지정할 수 있습니다. "누른" 또는 "비활성화" 상태에 대한 지정이 없으면 컨트롤은 "안 누른" 상태에 대한 아이콘을 표시합니다.

기타 속성은 각 아이콘의 정렬, 크기 조절 및 투명성을 제어합니다.

속성	의미/동작
Picture	전경 아이콘 "안 누른" 상태의 아이콘 이름
PicturePressed	전경 아이콘 "누른" 상태의 아이콘 이름
PictureDisabled	전경 아이콘 "비활성화됨" 상태의 아이콘 이름
BackgroundPicture	배경 아이콘 "안 누른" 상태의 아이콘 이름
BackgroundPicturePressed	배경 아이콘 "누른" 상태의 아이콘 이름
BackgroundPictureDisabled	배경 아이콘 "비활성화됨" 상태의 아이콘 이름

Alignment 속성

아이콘 지정 목록의 **정렬** 을 참조하는 값을 가진 태그에서 추가적인 속성을 지정할 수 있습니다.

속성 값	정렬
left	왼쪽 정렬
right	오른쪽 정렬
top	위로 정렬
bottom	아래로 정렬
center	가운데 정렬
stretch	배경 아이콘: **stretch** 값이 사용되면 파서는 버튼의 사각형 영역에 맞게 아이콘의 크기를 조절합니다. **transparent** 속성을 이용해 투명 색상을 선언함으로써 버튼에 어떤 윤곽선이라도 생성할 수 있습니다.

3.13.2.3 푸시버튼의 컨트롤 변수

버튼 속성은 아래의 속성 변수에 새 값을 할당하여 변경할 수 있습니다. 컨트롤 이름을 지정하고 그 다음에 컨트롤 변수 이름을 지정하여 연산 명령으로 값이 할당됩니다. 두 이름은 마침표로 구분해야 합니다.

구문

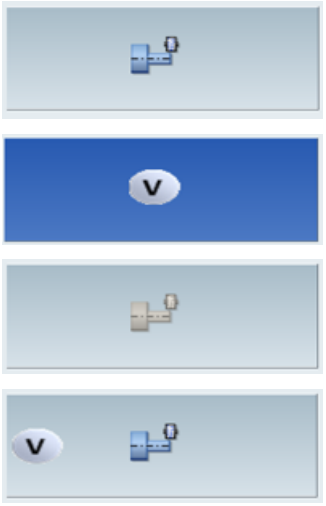
<컨트롤 이름>.<컨트롤 변수>

컨트롤 변수	의미/동작
backgroundpicture	변수 유형: String 배경 아이콘 이름
BackgroundPictureDisabled	변수 유형: String 원하는 상태의 배경 아이콘 이름
BackgroundPicturePressed	변수 유형: String 누른 상태의 배경 아이콘 이름
그림	변수 유형: String 전경 아이콘 이름
PictureDisabled	변수 유형: String 원하는 상태의 전경 아이콘 이름
PicturePressed	변수 유형: String 누른 상태의 전경 아이콘 이름
picture_textalignedtopicture	변수 유형: Bool 값: 1 = true 0 = false 버튼 텍스트가 아이콘에 정렬됩니다.
picturedisabled_textalignedtopicture	변수 유형: Bool 값: 1 = true 0 = false 버튼 텍스트가 "비활성화 상태" 아이콘에 정렬됩니다.

컨트롤 변수	의미/동작
picturepressed_textalignedtopicture	변수 유형: Bool 값: 1 = true 0 = false 버튼 텍스트가 "누른 상태" 아이콘에 정렬됩니다.
backgroundpicture_keepaspectratio	변수 유형: Bool 값: 1 = true 0 = false 배경 아이콘의 가로 세로 비율이 유지되거나 무시됩니다.
backgroundpicturedisabled_keepaspectratio	변수 유형: Bool 값: 1 = true 0 = false "비활성 상태" 배경 아이콘의 가로 세로 비율이 유지되거나 무시됩니다.
backgroundpicturepressed_keepaspectratio	변수 유형: Bool 값: 1 = true 0 = false "누른 상태" 배경 아이콘의 가로 세로 비율이 유지되거나 무시됩니다.
picture_keepaspectratio	변수 유형: Bool 값: 1 = true 0 = false 아이콘의 가로 세로 비율이 유지되거나 무시됩니다.

컨트롤 변수	의미/동작
picturedisabled_keepaspectratio	변수 유형: Bool 값: 1 = true 0 = false "비활성 상태" 아이콘의 가로 세로 비율이 유지되거나 무시됩니다.
picturepressed_keepaspectratio	변수 유형: Bool 값: 1 = true 0 = false "누른 상태" 아이콘의 가로 세로 비율이 유지되거나 무시됩니다.
backgroundpicture_scaled	변수 유형: Bool 값: 1 = true 0 = false 배경 아이콘이 버튼의 크기에 맞게 조절됩니다.
backgroundpicturedisabled_scaled	변수 유형: Bool 값: 1 = true 0 = false "비활성화 상태" 배경 아이콘이 버튼의 크기에 맞게 조절됩니다.
backgroundpicturepressed_scaled	변수 유형: Bool 값: 1 = true 0 = false "누른 상태" 배경 아이콘이 버튼의 크기에 맞게 조절됩니다.

컨트롤 변수	의미/동작
picture_scaled	변수 유형: Bool 값: 1 = true 0 = false 아이콘이 버튼의 크기에 맞게 조절됩니다.
picturedisabled_scaled	변수 유형: Bool 값: 1 = true 0 = false "비활성화 상태" 아이콘이 버튼의 크기에 맞게 조절됩니다.
picturepressed_scaled	변수 유형: Bool "누른 상태" 아이콘이 버튼의 크기에 맞게 조절됩니다.
backpicture_alignment	변수 유형: String alignment 속성 참조.
backgroundpicturedisabled_alignment	
backgroundpicturepressed_alignment	
picture_alignment	
picturedisabled_alignment	
picturepressed_alignment	
caption	변수 유형: String "안 누른 상태" 버튼 텍스트
captionpressed	변수 유형: String "누른 상태" 버튼 텍스트
caption_alignment	변수 유형: String alignment 속성 참조.
captionpressed_alignment	
captiondisabled_alignment	
disabled	변수 유형: Bool 값: 1 = true - 푸시버튼 비활성화됨 0 = false - 푸시버튼 작동 가능



그림

PicturePressed

PictureDisabled

BackgroundPicture + 그림

TextAlignedToPicture 속성

속성 값이 **true** 이면, 텍스트가 아이콘에 대해 정렬됩니다.

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

Scaled 속성

속성 값이 **true** 이면 버튼 높이 또는 너비의 50%가 그래픽에 사용될 수 있도록 버튼의 크기에 맞게 이미지 크기가 조절됩니다.

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

Stretch 속성(배경 아이콘에만 적용 가능)

속성 값이 **true** 이면 버튼 크기에 맞게 이미지 크기가 조절됩니다.

```
<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 스위치 on/off

스위치 컨트롤은 아이콘을 통해 두 가지 태 중 하나에 대한 신호를 보내는 그래픽 요소입니다. 이 컨트롤은 터치나 마우스 동작을 통해 작동합니다.

적용된 상태가 기준 변수에 저장되거나 상태 변경이 컨트롤에 지정된 함수에서 판단될 수 있습니다. 함수는 컨트롤 태그의 **function** 속성을 이용해 지정됩니다.

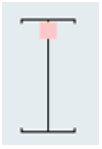
alignment 속성은 버튼의 수직 또는 수평 정렬을 활성화합니다.

이 컨트롤에는 표준 디자인이 없습니다. 컨트롤에 할당된 아이콘이 없으면 경계 상자와 스위치 위치(점으로 표시)만 표시됩니다.

구문

```
<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>
```

Alignment 속성



수직 스위치: 값 **vr**

수평 스위치: 값 **hr**

3.13.4 스위치 기능

3.13.4.1 스위치 속성

태그 속성

다음 스위치 위치는 **property** 태그로 컨트롤에 지정될 수 있습니다.

속성	의미/동작
left_position	스위치가 왼쪽으로 이동하면 이 속성 값이 컨트롤 변수에 지정됩니다.
right_position	스위치가 오른쪽으로 이동하면 이 속성 값이 컨트롤 변수에 지정됩니다.
upper_position	스위치가 위로 이동하면 이 속성 값이 컨트롤 변수에 지정됩니다.
lower_position	스위치가 아래로 이동하면 이 속성 값이 컨트롤 변수에 지정됩니다.
disabled	이 속성은 버튼의 조작 가능 여부를 제어합니다. 값이 **true** 이면 제어 동작을 평가하지 않습니다. 지정된 기준 변수에 의한 상태 변화는 스위치 상태 업데이트를 유발합니다.

마지막 스위치 디자인은 추가 속성 지정을 통해 결정되어야 합니다. 이러한 속성은 **left_position**, **right_position**, **upper_position** 또는 **lower_position** 속성과 함께 정의되어야 합니다.

transparent 속성을 이용해 한 색상을 투명색으로 선언함으로써 스위치에 어떤 윤곽선이라도 생성할 수 있습니다.

속성	의미/동작
그림	"안 누른" 상태의 아이콘 전경 이름 아이콘
pictureDisabled	"비활성화" 상태의 아이콘 전경 이름 아이콘
투명	이 속성 값은 투명 색의 색상 값을 포함합니다. 이 색상 값은 스위치에 지정된 모든 아이콘에 사용됩니다.
caption	이 속성 값은 스위치 위치와 연관된 텍스트를 포함합니다. 이 텍스트는 요소에 표시되고 스위치 크기의 제약을 받습니다.

3.13.4.2 스위치의 컨트롤 변수

스위치 속성은 아래의 컨트롤 변수에 새 값을 할당하여 변경할 수 있습니다. 컨트롤 이름을 지정하고 그 다음에 컨트롤 변수 이름을 지정하여 연산 명령으로 값이 할당됩니다. 두 이름은 마침표로 구분해야 합니다.

구문

<컨트롤 이름>.<컨트롤 변수>

컨트롤 변수	의미/동작
left_position	변수 유형: Int 스위치가 왼쪽으로 이동하면 이 속성 값이 컨트롤 변수에 지정됩니다.
right_position	변수 유형: Int 스위치가 오른쪽으로 이동하면 이 속성 값이 컨트롤 변수에 지정됩니다.
lower_position	변수 유형: Int 스위치가 아래로 이동하면 이 속성 값이 컨트롤 변수에 지정됩니다.
upper_position	변수 유형: Int 스위치가 위로 이동하면 이 속성 값이 컨트롤 변수에 지정됩니다.
Picture_left_position	변수 유형: String 왼쪽 위치 아이콘 이름
Picture_right_position	변수 유형: String 오른쪽 위치 아이콘 이름
Picture_upper_position	변수 유형: String 위쪽 위치 아이콘 이름
Picture_lower_position	변수 유형: String 아래쪽 위치 아이콘 이름
Picturedisabled_left_position	변수 유형: String "비활성화 상태" 왼쪽 위치 아이콘 이름
Picturedisabled_right_position	변수 유형: "비활성화 상태" 오른쪽 위치 아이콘 문자열 이름

컨트롤 변수	의미/동작
Picturedisabled_upper_position	변수 유형: String "비활성화 상태" 위쪽 위치 아이콘 이름
Picturedisabled_lower_position	변수 유형: String "비활성화 상태" 아래쪽 위치 아이콘 이름
Caption_left_position	변수 유형: String 왼쪽 위치 텍스트
Caption_right_position	변수 유형: String 오른쪽 위치 텍스트
Caption_upper_position	변수 유형: String 위쪽 위치 텍스트
Caption_lower_position	변수 유형: String 아래쪽 위치 텍스트
disabled	변수 유형: Bool 값: 1 = true - 스위치 비활성화됨 0 = false - 스위치 작동 가능

수평 표시

```
<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>
```

수직 표시

```
<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>
```

또는 핸들러 함수 지정

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

예

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>
```

컨트롤 변수를 이용한 속성 지정

```
<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 라디오 버튼

라디오 버튼은 여러 옵션 중 하나를 선택할 수 있게 합니다. 옵션 별로 버튼을 생성해야 합니다. 같은 양식, 그룹 상자 또는 스크롤 영역에 속한 모든 라디오 버튼은 하나의 버튼만 선택 표시될 수 있도록 상호 작용합니다. 버튼 상태는 컨트롤 변수에 전송됩니다.

구문

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

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

3.13.6 체크상자

체크상자는 여러 옵션을 선택할 수 있도록 합니다. 옵션 별로 버튼을 생성해야 합니다. 체크박스 상태는 컨트롤 변수에 전송됩니다.

구문

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

3.13.7 그룹 상자

그룹 상자는 컨트롤 그룹을 프레임 및 타이틀과 함께 최적으로 묶습니다. 그룹 내에서만 상호작용하도록 만들어진 연관 컨트롤을 논리적으로 그룹화합니다. 포함된 컨트롤의 좌표는 제목줄 아래의 상단 왼쪽 코너를 참조합니다.

그룹에 속한 모든 컨트롤은 컨트롤 태그에 서브 컨트롤으로서 포함됩니다.

포함된 컨트롤 이름과 **Item_Data** 값 지정 시, 양식에 속한 모든 컨트롤과 관련하여 고유한 이름과 값이어야 한다는 사실을 주의하십시오.

그룹 상자의 제목은 **caption** 태그로 지정합니다.

구문

```
<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 스크롤 영역

스크롤 영역은 지정된 영역 내의 컨트롤 표시에 사용됩니다. 포함된 컨트롤의 좌표는 스크롤 영역의 상단 왼쪽 코너를 참조합니다. 컨트롤이 가시 영역 밖에 생성되는 경우, 가시 영역 이동이 활성화됩니다. 이 영역에 속한 모든 컨트롤은 컨트롤 태그에 서브 컨트롤으로서 포함되어 있어야 합니다.

포함된 컨트롤 이름과 **Item_Data** 값 지정 시, 양식에 속한 모든 컨트롤과 관련하여 고유한 이름과 값이어야 한다는 사실을 주의하십시오.

구문

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

터치 조작

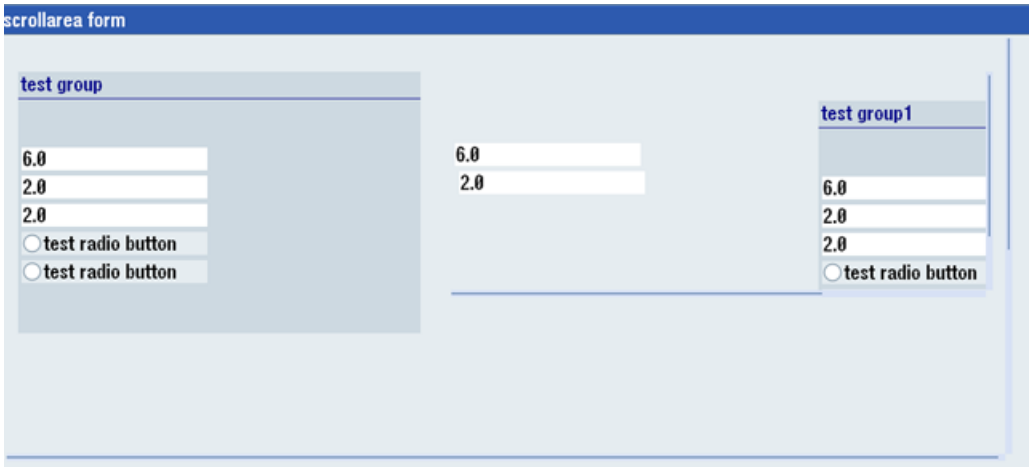
현재 완전하게 보이지 않는 컨트롤에 포커스가 있는 경우 컨트롤이 완전하게 표시될 때까지 가시 영역이 이동합니다.

탭 동작

포함된 컨트롤에 동작을 지정할 수 없는 경우 이 동작이 스크립트에 전달됩니다.

예

중첩된 스크롤 영역




```
<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 사이드스크린 애플리케이션

3.14.1 사이드스크린의 Easy XML

Easy XML 스크립트 언어는 사이드스크린 영역의 대화 상자 생성에도 사용할 수 있습니다. 사이드스크린의 **페이지** 와 **위젯** 구성요소가 대화 상자 표시에 사용 가능합니다. **slsidescreen.ini** 파일을 통해 연결할 수 있습니다. 사이드스크린 구성요소는 운영 소프트웨어가 시작될 때 자동으로 실행되어 시스템이 종료될 때만 종료됩니다. 이것은 사이드스크린 구성요소가 처음으로 활성화될 때 파서가 지정된 스크립트를 한 번 불러온다는 것을 의미합니다.

사이드스크린 영역의 크기는 **slsidescreen_<화면 해상도>.ini** 파일의 레이아웃 파라미터에 의해 결정됩니다. 독립적 구성을 활성화하기 위해 Easy XML 스크립트의 가용 너비는 276픽셀입니다. Easy XML 대화 상자의 가용 높이는 사용되는 사이드스크린 구성요소에 따라 다릅니다. 페이지의 경우 768픽셀이 가능합니다. 위젯의 경우 사이드스크린 관리에 의해 높이가 결정되고 **GETHMIResolution** 함수로 조회할 수 있습니다.

스크립트 파서는 메뉴가 형상을 활성화하거나 다른 형상으로 분기할 것을 기대합니다. 메인 메뉴의 이름은 **main** 이어야 합니다. 사이드스크린은 소프트 키 태그를 지원하지 않기 때문에 형상에 대한 작업자 제어를 통해 다른 형상으로 이동해야 합니다.

사이드스크린 페이지나 위젯의 수는 파서의 제약을 받지 않습니다.

3.14.2 사이드스크린 대화 상자 통합

페이지나 위젯은 **slsidescreen.ini** 파일을 이용해 통합됩니다.

페이지는 **[Sidescreen]** 섹션에 입력됩니다. 각 페이지는 키워드 **PAGE** 다음에 순차적 페이지 번호 및 원하는 속성을 지정해야 합니다. **name** 속성은 페이지의 오브젝트 이름을 정의합니다. **implementation** 속성은 실행 라이브러리와 클래스 이름을 지정합니다. 두 속성 모두 마침표로 구분해야 합니다. **SlSideScreenPage** 유형 페이지는 사이드스크린 요소를 포함할 수 있습니다. 이것은 "**ELEMENT**" 입력을 통해 이루어집니다. **ELEMENTxxx** 요소 라인 생성 규칙은 페이지 생성 규칙과 일치합니다. Easy XML으로 구성된 대화 상자만 페이지에 포함해야 하는 경우, **implementation** 속성에 대한 값으로서 **slagmforms.SlEESideScreenPage** 가 지정되어야 합니다.

위젯은 **[Element_<name>]** 섹션의 사이드스크린 요소에 추가될 수 있습니다.

WIDGETxxx 위젯 라인 생성 규칙은 페이지 생성 규칙과 일치합니다.

Easy XML 사이드스크린 위젯을 활성화하려면 **implementation** 속성 값으로 **slagmforms.SlEESideScreenWidget** 을 지정해야 합니다.

구문

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

…
…

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

위젯 식별자는 메인 모듈 이름을 만들 때 기본적으로 사용됩니다.

예

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

메인 모듈 이름: sidescreen_proginfluence.xml

추가 속성

이름이 **PAGE_<pagename>**, **ELEMENT_<elmentname>** 또는 **WIDGET_<widgetname>** 인 섹션을 **slsidescreen.ini** 파일에 입력하여 페이지나 위젯에 추가 속성을 지정할 수 있습니다.

페이지, 요소 또는 위젯에 다음 속성을 지정할 수 있습니다.

속성	의미/동작
TextId	텍스트의 언어 종속적 식별자
TextFile	텍스트 파일 이름
TextContext	텍스트 컨텍스트 EASY_XML
Icon	아이콘 이름

속성	의미/동작
maskPath	이 속성은 사용할 메인 모듈 이름을 지정합니다.
maskName	이 속성은 사용할 메인 메뉴 이름을 지정합니다.
focusable	이 속성은 입력 포커스를 요소에 설정할 수 있는지 여부를 결정합니다.

예

```
[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 언어 및 텍스트 관리

언어 종속적 텍스트를 관리하려면 각 구성요소를 텍스트 파일에 지정해야 합니다. 텍스트 파일 이름은 구성요소의 이름, 언어 버전, ".ts" 확장자로 만들어집니다.

EASY_XML 이 텍스트 컨텍스트로 사용되어야 합니다.

구문

<name>_<language>.ts

예

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

텍스트 파일 이름: sidescreen_proginfluence_**eng.ts**

지정된 텍스트 파일이 없으면 파서가 **oem_xml_screens_xxx.ts** 기본 텍스트 파일에서 텍스트 식별자를 검색합니다.

3.14.4 사이드스크린 구성요소

3.14.4.1 사이드스크린 요소

페이지가 디폴트 실행에 연결되어 있으면 요소가 이 페이지에 지정될 수 있습니다. **[Page_<page_name>]** 섹션이 이를 위해 생성되고 이 페이지와 관련된 모든 요소가 목록으로 표시됩니다.

예

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

[Page_<page_name>]

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

각 요소는 속성과 관련 위젯이 표시될 **[Element_<element name>]** 섹션을 필요로 합니다.

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

3.14.4.2 사이드스크린 위젯

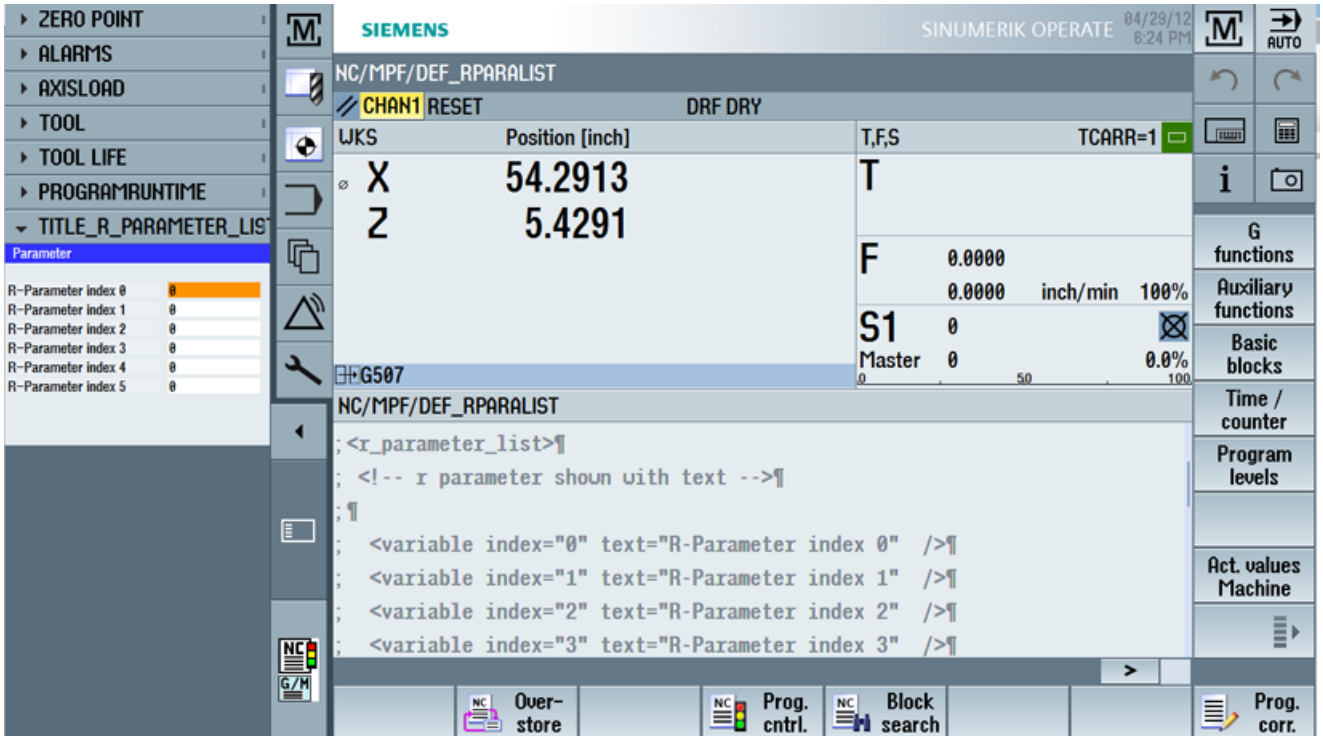
구성된 형상이 표시될 수 있는 영역이 사이드스크린 관리에 의해 위젯에 지정됩니다. 기본적으로 위젯은 입력 포커스를 얻지 못하기 때문에 필드를 수정할 수 없습니다. 이러한 동작은 **focusable** 속성을 지정하여 변경할 수 있습니다. 위젯이 열리면 파서가 메인 메뉴와 관련된 양식을 엽니다. 사용자는 양식 내에서 탐색 명령을 통해 다른 메뉴로 분기할 수 있습니다.

예

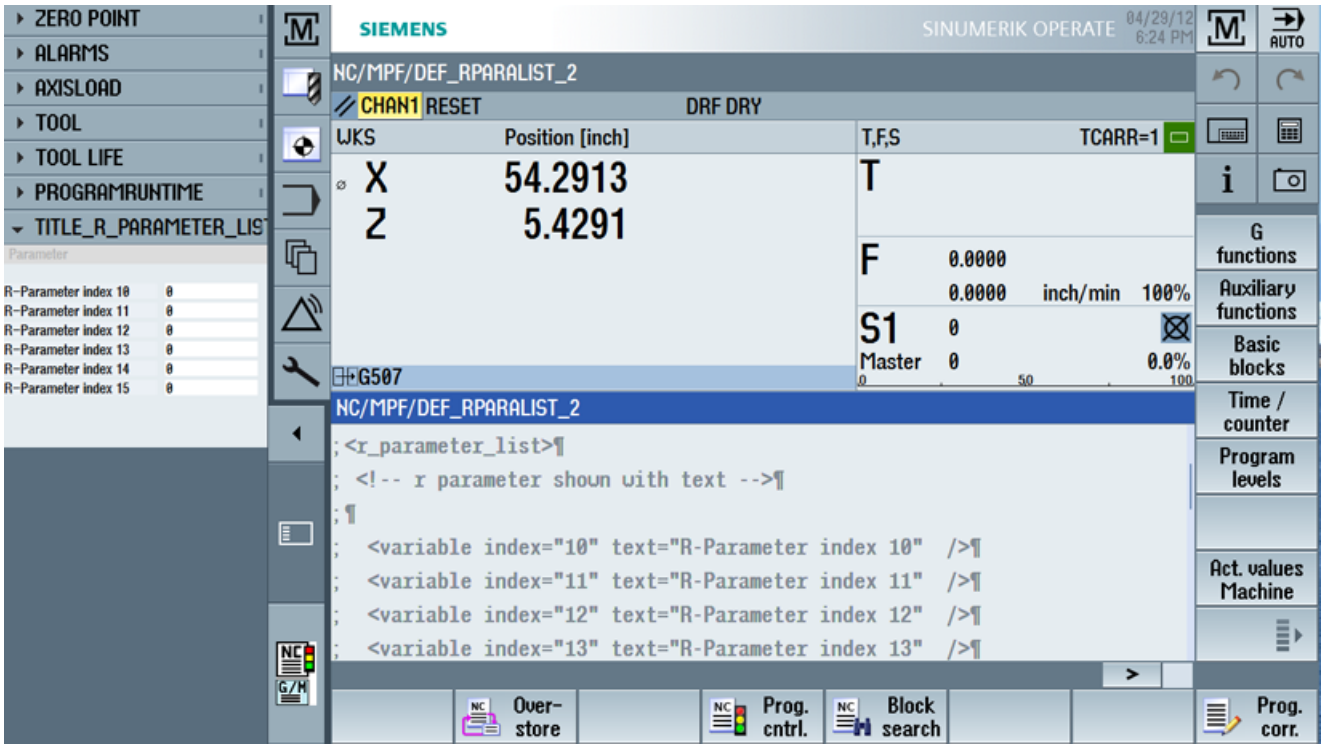
Easy XML 스크립트는 활성 가공 프로그램에서 위젯의 데이터 내용에 대한 설명을 검색합니다. 이 특정 예에서 R 파라미터가 있는 목록 설명이 표시되어야 합니다. 각 파라미터에 설명 텍스트를 지정할 수 있습니다.

툴박스 예

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



다른 공구 프로그램이 선택되어 있으면 위젯이 자동으로 재작업됩니다.


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
NC/MPF/DEF_RPARALIST_2
CHAN1 RESET
DRF DRY
WKS
Position [inch]
X 54.2913
Z 5.4291
T,F,S
TCARR=1
T
F 0.0000
0.0000 inch/min 100%
S1 0
Master 0 0.0%
G507
NC/MPF/DEF_RPARALIST_2
;<r_parameter_list>¶
; <!-- r parameter shown with text -->¶
;¶
; <variable index="10" text="R-Parameter index 10" />¶
; <variable index="11" text="R-Parameter index 11" />¶
; <variable index="12" text="R-Parameter index 12" />¶
; <variable index="13" text="R-Parameter index 13" />¶
AUTO
G functions
Auxiliary functions
Basic blocks
Time / counter
Program levels
Act. values Machine
Over-store
Prog. cntrl.
Block search
Prog. corr.

3.14.4.3 사이드스크린 페이지

slagmforms.SIEESideScreenPage 실행을 통해 시작하는 페이지는 양식의 전체 사이드스크린 영역을 사용할 수 있도록 합니다. 제목줄이 없으므로 제목을 Caption 명령문으로 구성할 수 없습니다. 기본적으로 페이지는 입력 포커스를 얻지 못하기 때문에 필드를 수정할 수 없습니다. 이러한 동작은 **focusable** 속성을 지정하여 변경할 수 있습니다. 페이지가 열리면 파서가 메인 메뉴와 관련된 양식을 엽니다. 사용자는 양식 내에서 탐색 명령을 통해 다른 메뉴로 분기할 수 있습니다.

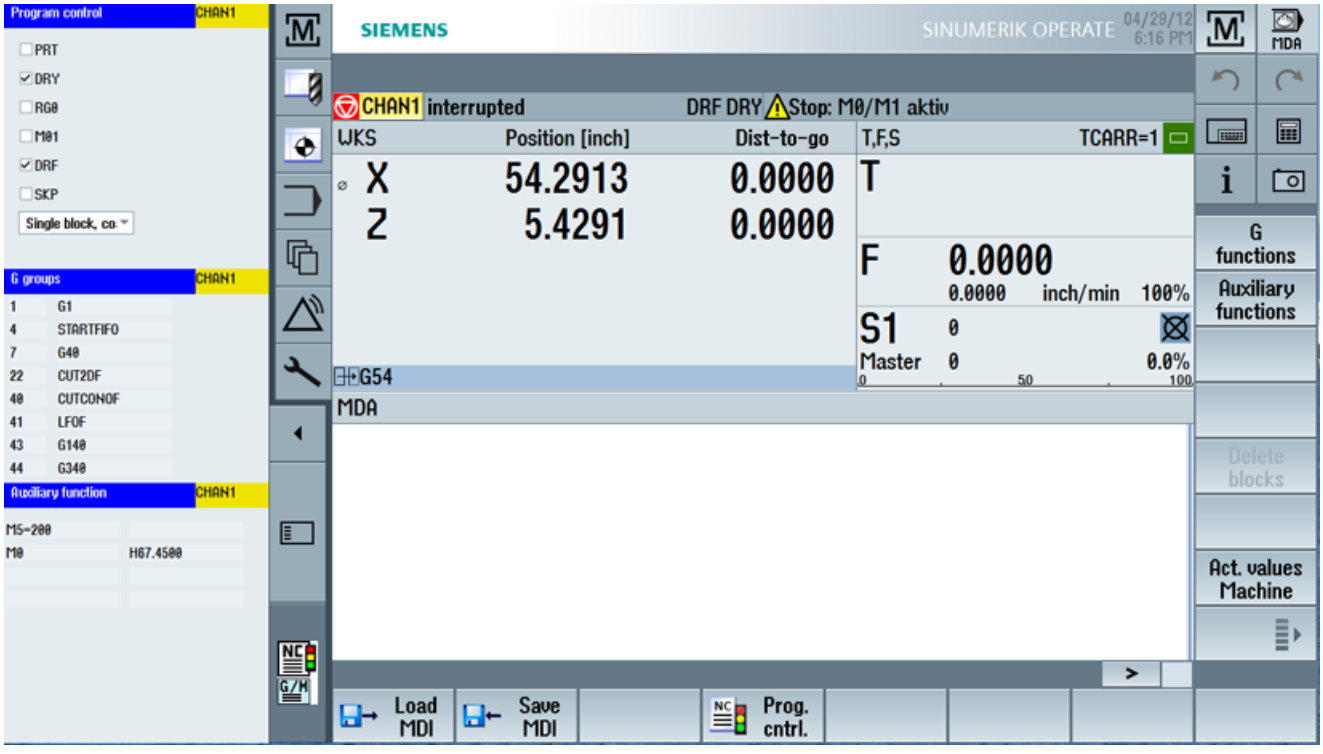
예

사이드스크린 페이지가 Easy XML 양식만 보여주는 페이지를 표시됩니다.

이 예에서는 추가적인 정보가 사이드스크린 영역에 표시됩니다.

툴박스 예

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
```

텍스트 파일: sidescreen_proginfluence_deu.ts

3.15 PLC 하드키를 통한 대화상자 선택

적용 분야

PLC는 운영 소프트웨어에서 다음과 같은 기능을 시작할 수 있습니다.

- 작업 영역 선택
- 작업 영역 내의 특정 시나리오 선택
- 소프트 키에 구성된 기능 실행

하드키

PLC 키를 포함한 모든 키를 하드키라고 부릅니다. 최대 254개의 하드키를 정의할 수 있습니다. 하드 키는 다음과 같이 분류됩니다.

키 번호	용도
키 1~키 9	화면 조작반 전면의 키
키 10~키 49	예비
키 50~키 254 키 50~키 81	PLC 키: OEM용 예비 키
키 255	제어 정보에 의해 사전 지정

하드키 1~9는 다음과 같이 사전 지정되어 있습니다.

키 명칭		동작 / 효과
HK1	기계좌표	"기계 좌표" 영역 또는 마지막 대화상자를 선택합니다.
HK2	프로그램	"프로그램" 영역, 마지막 대화상자 또는 마지막 프로그램을 선택합니다.
HK3	옵셋	"파라미터" 영역 또는 마지막 대화상자를 선택합니다.
HK4	프로그램 관리자	"프로그램" 작업 영역, "프로그램 관리자" 루트 화면을 선택합니다. 기능 없음
HK5	알람	"진단" 영역, "알람 목록" 대화상자를 선택합니다.
HK6	CUSTOM	"사용자 정의" 작업 영역을 선택합니다.
HK7	메뉴 선택	"메인 메뉴"를 선택합니다.

키 명칭		동작 / 효과
HK8	기능 메뉴	"기능" 작업 영역을 선택합니다.
HK9	사용자 메뉴	"사용자" 작업 영역을 선택합니다.

구성

각 키는 구성 파일 systemconfiguration.ini의 [keyconfiguration] 영역에 구성합니다. 각 라인은 하드키 이벤트를 정의합니다. 하드키 이벤트는 특정 하드키를 n번째 활성화했을 때의 반응을 말합니다. 예를 들어 특정 하드키를 두 번째 눌렀을 때와 세 번째 눌렀을 때 반응이 다를 수 있습니다.

구성 파일 systemconfiguration.ini의 항목은 사용자 설정으로 덮어쓸 수 있습니다. 이 용도로 [System user 디렉토리]/cfg 및 [System oem 디렉토리]/cfg 디렉토리를 사용할 수 있습니다.

하드키 이벤트 구성을 위한 라인은 다음과 같은 구조를 갖습니다.

```
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: 하드키 번호, 값 범위: 1 – 254

n: 이벤트 번호 - 하드키의 활성화 횟수와 동일, 값 범위: 0 – 9

요구사항

PLC 사용자 프로그램이 다음 조건을 충족해야 합니다.

하나의 하드키만 처리해야 합니다. 따라서 운영 소프트웨어가 이전 요청을 승인한 경우에만 새 요청을 설정할 수 있습니다. PLC 사용자 프로그램이 MCP 키에서 하드키를 가져오는 경우 키를 아주 빨리 눌렀을 때 키 입력이 취소되지 않도록 충분한 버퍼 스토리지를 제공해야 합니다.

PLC 인터페이스

하드키 선택을 위한 영역은 PLC 인터페이스에 제공됩니다. 이 영역은 DB19.DBB10에 있습니다. 이 DB에서 PLC는 50~254 사이의 값을 키에 직접 지정할 수 있습니다.

운영 소프트웨어는 두 단계로 요청을 승인합니다. 2단계 승인이 필요한 이유는 같은 키 코드를 두 번 연속해서 누른 경우 운영 소프트웨어가 두 이벤트를 분리해서 정확히 인식할 수 있

어야 하기 때문입니다. 첫 번째 단계에서 제어 정보 255가 바이트 DB19.DBB10에 쓰여집니다. 사전 정의된 이 가상 키 스트로크가 활성화되면 HMI가 모든 PLC 키 시퀀스를 구분해서 인식할 수 있게 됩니다. 제어 정보는 PLC 사용자 프로그램에 영향을 미치지 않으므로 변경해서는 안됩니다. 두 번째 단계에서는 DB19.DBB10를 삭제하여 PLC에 대한 실제 승인이 이루어집니다. 이 시점부터 PLC 사용자 프로그램은 새 하드키를 지정할 수 있습니다. 이와 동시에 현재 하드키 요청이 운영 소프트웨어에서 처리됩니다.

예

구성 파일:

```
; OP 하드키(KEY1-KEY9) 구성 및
; PLC 하드키 (KEY50~KEY254)
[keyconfiguration]
; 장비 키(하드키 블록)
KEY1.0 = area:=AreaMachine, dialog:=SlMachine
; 프로그램 키(하드키 블록)
KEY2.0 = area:=AreaProgramEdit
; 오프셋 키(하드키 블록)
KEY3.0 = area:=AreaParameter
; 프로그램 매니저 키(하드키 블록)
KEY4.0 = area:=AreaProgramManager
; 알람키(하드키 블록)
KEY5.0 = area:=AreaDiagnosis, dialog:=SlDgDialog,
cmdline:="-slGfwHmiScreen SlDgAeAlarmsScreen"
; 사용자 지정(하드키 블록)
KEY6.0 = area:=Custom
; 장비 키(옵션, Shift + F10)
KEY7.0 = area:=AreaMachine, dialog:=SlMachine,
cmdline:="-MKey"
; 메뉴 사용자(하드키 블록, Ctrl + F10)
KEY10.0 = area:=MenuUser
; 메뉴 기능(하드키 블록, Ctrl + Shift + F10)
KEY11.0 = area:=MenuFunction
KEY50.0 = area:=CustomXML, dialog:=SlEECustomDialog
```

영역 및 대화상자 식별자는 [System Siemens 디렉토리]/cfg 디렉토리의 systemconfiguration.ini에서 확인할 수 있습니다.

SlEECustomDialog 만 사용한다면, 파서는 **xmldial.xml** 파일이 애플리케이션 디렉토리에 있고 메뉴 태그의 이름이 **main**일 것으로 기대합니다. 다른 파일 이름이나 메인 메뉴 이름은 **cmdline** 키를 추가하여 알릴 수 있습니다. **-mainModule** 파라미터는 불러올 XML 설명을 정의합니다. 파서는 메인 메뉴가 이 스크립트에 있을 것으로 기대합니다. **-entry** 파라미터는 메인 메뉴의 이름을 정의합니다. 이 파라미터가 없으면 파서는 **main** 이라는 이름을 이용해 기능을 실행합니다. 값이 **slagmdialog.hmi** 인 **-conf** 파라미터가 항상 포함되어야 합니다.

예:

```
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"
```

사용자 인터페이스

systemconfiguration.ini 파일에 있는 다음과 같은 입력 항목을 통해 상기 설명한 기능을 위해 EASY XML을 사용할 수 있습니다.

```
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 예약된 소프트 키에 사용자 대화상자 지정

소프트 키는 기존 기능의 확장을 위해 모든 표준 작업 영역에서 예약됩니다. 해당 파일은 사용자 프로젝트와 함께 OEM 소프트 키의 활성화 및 링크를 위해 시스템의 다음 디렉토리에 저장됩니다.

/siemens/sinumerik/hmi/template/cfg

이 파일들은 작업 영역별 XML 설명을 포함하는데, 이 설명은 운영 소프트웨어 실행 시 작업 영역의 메뉴 트리에서 해석 및 통합됩니다.

별도 애플리케이션의 통합을 위해 해당 작업 영역 관련 파일이 템플릿 디렉토리에서 다음 디렉토리로 복사되어 수정됩니다.

/oem/sinumerik/hmi/template/cfg

3.16.1 언어 중립적 소프트 키 텍스트 정의

TEXTFILE 태그는 OEM 영역에 지정될 텍스트 파일의 파일 이름을 포함합니다. 이 파일 이름은 언어와 파일 확장자 없이 지정됩니다.

구문

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

예

oem_xml_screens_deu.ts

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

표시될 소프트 키 텍스트는 **property** 태그에서 **softkey** 태그로 관리됩니다. OEM 값은 원하는 텍스트 ID로 대체됩니다. **translationContext** 속성은 텍스트가 지정되는 텍스트 파일의 영역을 참조합니다. easyXML의 경우, EASY_XML 컨텍스트가 지정됩니다.

구문

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

예

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

3.16.2 Easy XML 영역 지정

탐색 대상을 **softkey** 태그에도 지정해야 합니다. 그러므로 탐색 대상 **CustomXML** 을 **area** 태그의 **name** 속성에 입력해야 합니다. **dialog** 와 **screen** 속성은 이 파라미터들이 자동으로 정의되므로 삭제됩니다.

구문

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

예

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

탐색 대상 **area** 가 사용되면 파서는 xmldial.xml 파일을 시작 모듈로, **main** 메뉴를 메뉴 관리의 시작 지점으로 기대합니다. 여러 애플리케이션을 동시에 제공하려면 **switchToDialog** 함수를 사용합니다. 이 경우, NAVIGATION 태그는 FUNCTION 태그로 대체해야 합니다. **CustomXML** 영역, **SlEECustomDialog** 대화 상자, 시작 모듈의 이름, 메인 메뉴의 이름이 호출 인수로서 이 함수에 전달됩니다.

switchToArea 태그로 easyXML 영역에서 표준 사용자 영역으로 돌아갑니다.

구문

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

프로젝트 예

sldgarea_oem.xml

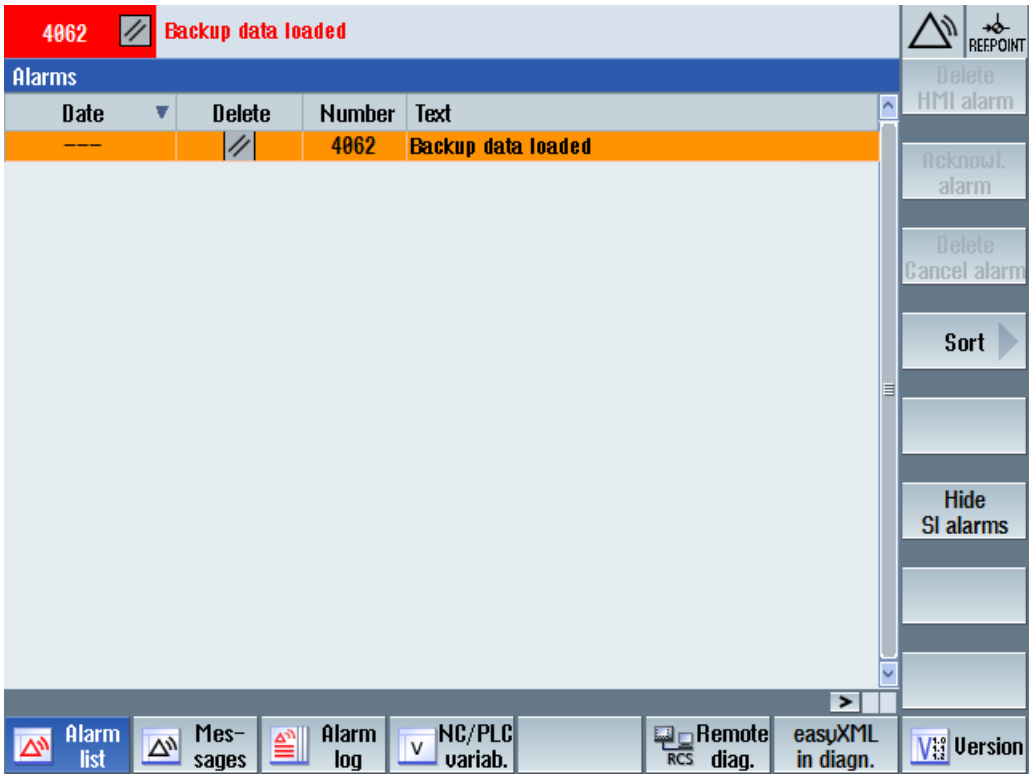
```
<!DOCTYPE DIALOG>
<DIALOG>
<TEXTFILE>oem_xml_screens</TEXTFILE>
<!-- ========================================================================
-->
<!-- 영역 진단의 첫 번째 수평 메인 메뉴에서 OEM 소프트 키 번호 7 -->
<!-- ========================================================================
-->
<MENU name="DgGlobalHu">
 <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>
```

또는

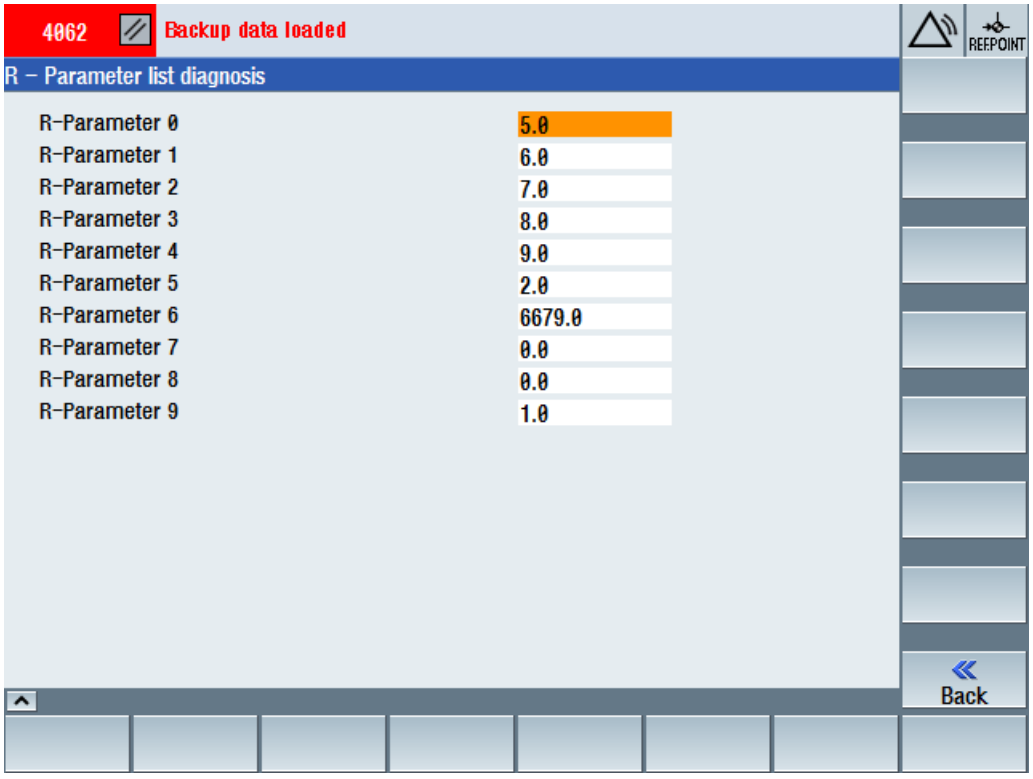
```
<!DOCTYPE DIALOG>
<DIALOG
<!-- ========================================================================
-->
<!-- 영역 진단의 첫 번째 수평 메인 메뉴에서 OEM 소프트 키 번호 7 -->
<!-- ========================================================================
-->
<TEXTFILE>oem_xml_screens</TEXTFILE>
<MENU name="DgGlobalHu">
 <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>
<!-- 이 양식은 R 파라미터 목록을 보여줍니다.-->
<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 태그 설명

태그 소프트 키

softkey 태그 밖에서 POSITION 속성을 이용하여 **state** 속성으로 소프트 키 상태를 설정할 수 있습니다. 상태가 설정될 소프트 키 번호는 속성 값으로 지정됩니다.

예

```
<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 태그

참고

예에서 요소의 사전 할당을 제거해야 합니다.

형식 정의 내에서 "**element**" 태그로 변수가 선언됩니다. 태그 속성은 **let** 명령의 속성과 일치합니다. 요소는 변수 생성 시 사전 할당할 수 있습니다.

예

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 태그

필드 요소의 개수는 **dim** 속성으로 지정됩니다. 2차원 필드의 경우 두 차원을 구분하기 위해 첫 번째 차원을 지정한 후 쉼표를 넣고 두 번째 차원을 지정합니다. 필드 요소는 변수 이름 다음 꺽쇠 괄호 ([]) 안에 지정된 필드 인덱스를 통해 액세스합니다. 배열의 크기는 직접 또는 변수 내용을 통해 지정할 수 있습니다.

예

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

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

또는

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

또는

```
<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 언어 중립적 텍스트 생성

대화 상자 형식에서 사용되는 모든 텍스트는 언어 중립적으로 관리할 수 있습니다. 즉, 런타임 동안 현재 선택된 언어로 된 텍스트로 대체되는 대화 상자에서 텍스트 식별자가 사용됩니다. 이 텍스트는 모든 사용 가능 언어에 대해 텍스트 식별자와 함께 UTF8 형식으로 된 텍스트 파일에 저장됩니다.

oem_xml_screens_<언어 코드>.ts 파일은 기본적으로 텍스트 파일로서 Easy XML 함수에 지정됩니다. EASY_XML 식별자는 텍스트 컨텍스트로서 지정됩니다.

스크립트에서 텍스트 식별자 앞에 이중 달러 기호를 접두어로 붙여 테스트 식별자를 표시합니다.

구조

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>
```

예

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

3.18 프로젝트별 텍스트 파일

3.18.1 프로젝트별 텍스트 파일 생성

별도의 프로젝트를 관리하기 위해 프로젝트, 각 양식, 각 메뉴에 별도의 텍스트 파일을 사용할 수 있습니다. TEXTFILE 속성은 해당 태그에서 알림을 제공합니다.

텍스트 파일의 이름은 언어 식별자와 파일 확장자 없이 속성 값으로서 전달됩니다. 양식이나 메뉴가 닫힐 때까지 계속해서 텍스트에 액세스할 수 있습니다. 텍스트 파일에 **DialogGui** 태그를 불러오면 프로젝트를 나갈 때까지 내용에 액세스할 수 있습니다. 텍스트 컨텍스트를 한 번 이상 사용할 경우 마지막으로 로드된 파일에서 텍스트 식별자 검색이 수행됩니다. 텍스트 파일당 하나의 텍스트 컨텍스트를 사용할 수 있습니다.

예

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>
```

활성화

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

또는

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

또는

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

```
</menu>
```

대화 상자 예

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>
```

대화 상자 스크립트

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 텍스트 컨텍스트 지정

텍스트 그룹화는 텍스트 파일 내에서 이름이 독립적인 영역의 이름을 통해 가능합니다. 영역 식별자는 **context** 태그 내에서 **name** 태그를 이용해 정의됩니다.

구문

```
<context>
```

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

예

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

특정 컨텍스트가 사용 중이라면, TEXTCONTEXT 속성을 통해 프로젝트, 양식 또는 메뉴에 대해 텍스트 파일 이름과 함께 해당 컨텍스트 이름을 지정해야 합니다. 새 컨텍스트가 설정될 때까지 컨텍스트 내에 저장된 텍스트에 계속 액세스할 수 있습니다. 그러므로 프로젝트에 설정된 컨텍스트는 양식이나 메뉴에 설정된 컨텍스트로 대체됩니다.

프로젝트 예제

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 Textfilelist 지정

프로젝트에 여러 텍스트 파일을 사용하려는 경우 **textfilelist** 속성을 사용해 목록에 있는 필수 텍스트 파일의 이름을 파서에 전달할 수 있습니다.

텍스트 파일 이름의 표기는 한 줄씩 인식되고 줄 바꿈으로 끝나야 합니다. 언어 확장자와 파일 확장자가 없는 파일의 이름은 텍스트 파일 이름으로 예상됩니다.

예

`form2_screens_deu.ts` 파일은 `form2_screens`으로 목록에 나타납니다.

textfilelist.ini

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

활성화

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

3.19 세션 복원

Easy XML 프로젝트가 종료되면 모든 로컬 변수가 활성화되고 스크립트 명령이 언로드됩니다. 제어 시스템이 꺼진 후에도 변수의 데이터를 유지하려는 경우 변수에 **permanent** 속성을 지정해야 합니다. 제어 시스템이 꺼질 때까지 유지될 데이터에는 **poweroff** 속성을 지정해야 합니다.

Easy XML 애플리케이션을 다시 선택하면 메인 메뉴에서 애플리케이션을 다시 선택한 후 활성화할 메뉴나 양식을 제어하는 것이 가능합니다. 이와 관련해 필요한 모든 데이터를 제공하는 것은 프로그래머의 책임입니다.

DialogGUI 태그에서 **restoresession** 속성이 설정된 경우 파서가 마지막으로 열린 메뉴와 마지막으로 열린 양식을 다시 선택합니다 tag, the parser organizes reselection of the menu last opened and the last opened form. 필요한 데이터를 제공하는 것은 프로그래머의 책임입니다.

스타트업 대화 상자 생성

4

4.1 기능 개요

목적

"Easy Extend" 기능은 는 옵션 장비를 스타트업, 활성화, 비활성화 또는 테스트할 수 있는 간편한 수단을 제공합니다. 제어 시스템이 사용 가능한 장비 및 디바이스 상태를 목록으로 표시합니다. 제어 시스템은 최대 64개 디바이스를 관리할 수 있습니다.

디바이스를 활성화 또는 비활성화할 때는 소프트 키를 사용합니다.

"Easy Extend" 기능은 "파라미터" 영역에서 제공됩니다.

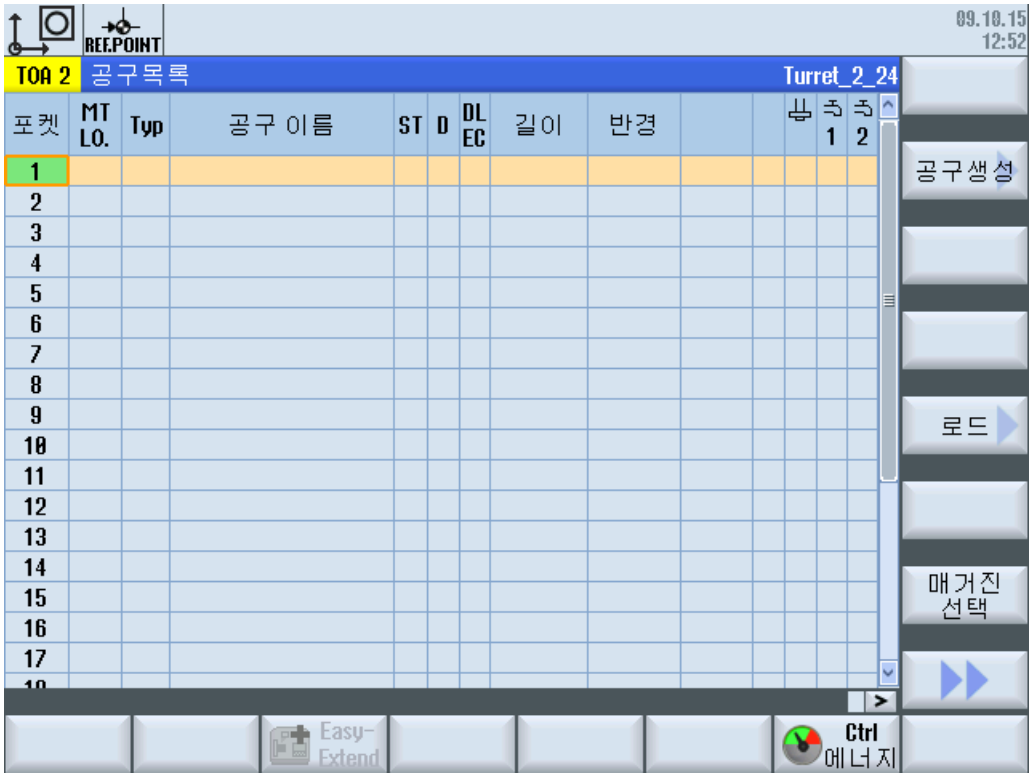


그림 4-1 메인 화면 "파라미터"

설정

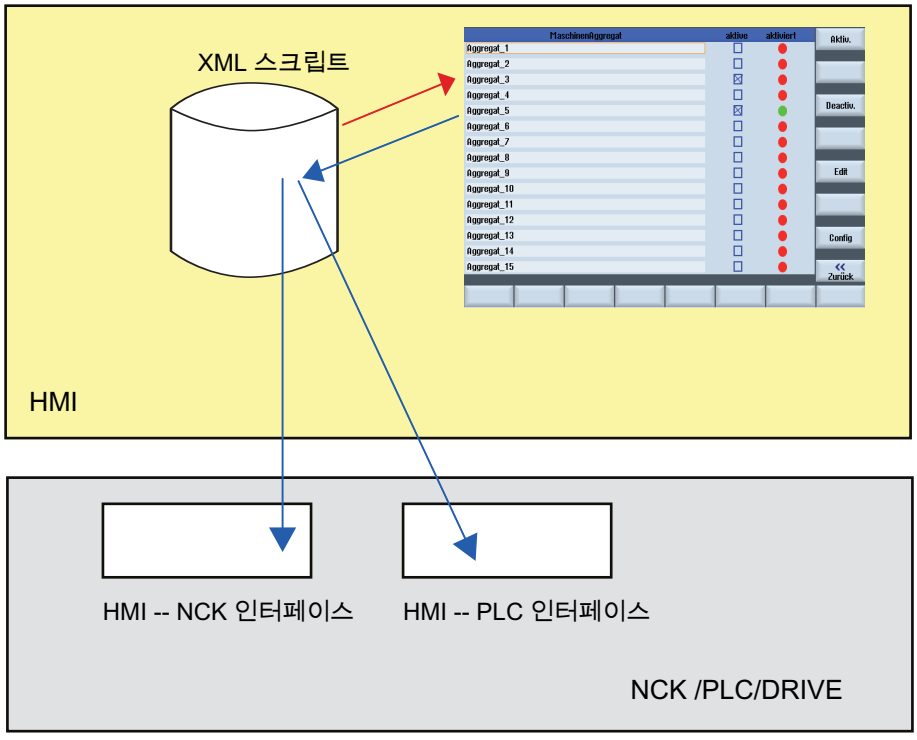


그림 4-2 "Easy Extend" 이용 방법

"Easy Extend" 기능을 이용하려면 장비 제조업체가 다음 함수를 설정해야 합니다.

- **PLC ↔ HMI 인터페이스**
  옵션 디바이스는 사용자 인터페이스와 PLC 사이의 인터페이스를 통해 관리됩니다.
- **스크립트 처리**
  장비 제조업체는 디바이스를 스타트업, 활성화, 비활성화 또는 테스트하는 순서를 명령문 스크립트에 저장합니다.
- **파라미터 대화 상자(옵션)**
  파라미터 대화 상자는 스크립트 파일에 저장된 디바이스 정보를 표시합니다.

파일 저장

"Easy Extend"에 속한 파일은 시스템 CompactFlash 카드의 "dvm"(장비 제조업체) 디렉토리에 저장됩니다.

파일	이름	대상 디렉토리
텍스트 파일	oem_aggregate_xxx.ts	\oem\sinumerik\hmi\lng
스크립트 파일	agm.xml	\oem\sinumerik\hmi\dvm
백업 파일	제한 없음	\oem\sinumerik\hmi\dvm\archives
PLC 사용자 프로그램	제한 없음	PLC

4.2 PLC 사용자 프로그램에서 설정

설정 로드

생성된 설정이 스크립트 및 텍스트 파일과 함께 제어 시스템의 제조업체 디렉토리로 전송됩니다. 이때 관련 PLC 사용자 프로그램이 로드되어야 합니다.

장비 프로그래밍

작업자 콤포넌트와 PLC 간의 통신은 PLC 사용자 프로그램에서 데이터 블록 DB9905를 통해 이루어집니다. 이 데이터 블록에는 디바이스 관리를 위해 128 워드가 준비되어 있습니다. 데이터 블록 지정자를 위한 설명은 "PLC_INTERFACE (페이지 314)" 장에 나와 있습니다.

디바이스 1부터 PLC 워드가 지정됩니다.

데이터 블록	디바이스 지정
DB9905.DBB0	디바이스 1
DB9905.DBB4	디바이스 2
...	...
DB9905.DBB192	Device 49
DB9905.DBB196	디바이스 50

각 디바이스에는 다음 의미를 가진 4 바이트가 사용됩니다.

바이트	비트	설명	
0	0	== 1	디바이스가 스타트업되었음 (HMI 확인)
	1	== 1	디바이스가 활성화될 예정 (HMI 요청)
	2	== 1	디바이스가 비활성화될 예정 (HMI 요청)
	3-7	예비	
1	0-7	예비	
2	0	== 1	디바이스가 활성 상태 (PLC 확인)
	1	== 1	디바이스에 에러가 있음
	2-7	예비	
3	0-7	디바이스의 고유 이름	

파일 저장

Easy Extend 파일은 시스템 콤팩트 플래시 카드의 "oem" (제조업체) 및 "oem_i" (개인) 디렉토리에 저장됩니다.

참고

파일 이름에는 소문자만 사용됩니다.

파일	이름	대상 디렉토리
텍스트 파일	oem_aggregate_xxx .ts	/oem/sinumerik/hmi/lng/ /oem_i/sinumerik/hmi/lng/
스크립트 파일	agm.xml	/oem/sinumerik/hmi/dvm /oem_i/sinumerik/hmi/dvm
백업 파일	제한 없음	/oem/sinumerik/hmi/dvm/archives /oem_i/sinumerik/hmi/dvm/archives
PLC 사용자 프로그램	제한 없음	PLC

일반 순서

장비 제조업체는 다음 단계를 실행하여 필요한 데이터를 사용할 수 있게 해야 합니다.

1. PLC 상에서 활성화 시 디바이스를 활성화하는 PLC 사용자 프로그램 생성
2. "표준 기계" 스타트업 이후 시리즈 스타트업 백업 파일에 데이터 백업
3. 디바이스를 설치하고 스타트업을 한 후 차등 시리즈 스타트업 백업 파일에 있는 데이터 읽어오기

참고

기계 설정 변경

드라이브 머신 데이터를 수정해야만 하는 경우 먼저 제어 시스템에서 수정 작업을 해야 합니다. 모든 디바이스 및 기기에 대해 이 절차를 반복해야 합니다.

축 추가

기계 축을 추가하여 기계를 확장하는 경우 지정된 순서대로 드라이브 오브젝트 (DO) 를 설치해야 합니다. 시리즈 스타트업 백업 파일에는 다양한 장비 제조업체 기본 기계가 포함되어 있기 때문에 이 순서가 변경되면 백업 파일을 적용할 수 없습니다.

"제어 콤포넌트"에는 다음 설정을 선택할 것을 권장합니다.

- NC 데이터
- PLC 데이터
- 드라이브 데이터
  - ACX 형식 (이진)

4.3 사용자 인터페이스의 디스플레이

사용자 인터페이스의 대화 상자

"Easy Extend" 기능에서 다음 대화 상자를 사용할 수 있습니다.

- 제어 시스템이 사용 가능한 장비가 표시된 **설정 가능한 대화 상자**를 제공합니다.
- 아직 첫 번째 스타트업을 하지 않은 경우 제어 시스템은 **스타트업 대화 상자**를 엽니다.

스타트업 절차 (XML 명령: "START_UP") 를 프로그램했지만 디바이스가 아직 스타트업되지 않은 경우 컨트롤이 스타트업 절차를 시작합니다.

이때 제어 시스템은 스크립트 파일에 저장된 시리즈 스타트업 백업 파일을 읽어오기 전에 전체 데이터 백업을 먼저 실시합니다.

스타트업 중 에러가 발생하면 스타트업 엔지니어는 스타트업 절차를 취소하거나 기계 설정에서 에러 원인을 직접 수정할 수 있습니다.

- "취소" 함수를 사용하면 스타트업을 조기에 중단할 수 있습니다. 이 경우 제어 시스템은 이전에 저장했던 스타트업 파일을 다시 복사합니다.

스타트업이 성공적으로 완료된 후 기계를 꺼야 한다면 XML 명령문 "POWER_OFF"를 사용하여 제어 시스템에 해당 메시지가 표시되도록 프로그래밍할 수 있습니다.

4.4 언어별 텍스트 생성

텍스트 파일의 구조

언어별 텍스트가 포함된 XML 파일은 UTF8 형식으로 생성해야 합니다.

예

oem_aggregate_eng.ts

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

oem_aggregate_deu.ts

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

4.5 사용자의 파워 유닛 설정 예제

드라이브 오브젝트 활성화

축을 옵션으로 판매하기 위해서 장비 제조업체는 활성화할 드라이브 오브젝트를 미리 스타트업한 후 다시 비활성화해 둡니다.

이 축을 활성화하려면 다음 절차를 수행하십시오.

- p0105를 통해 드라이브 오브젝트를 활성화하십시오.
- 채널 머신 데이터에서 두 번째 축을 인에이블하십시오.
- p0971를 통해 드라이브 머신 데이터를 백업하십시오.
- 데이터가 쓰여질 때까지 기다리십시오.
- NCK와 드라이브를 다시 시작하십시오.

프로그래밍:

```
<DEVICE>
  <list_id>1</list_id>
  <name> "Activate the drive"</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>
```

PLC로 제어되는 디바이스 활성화

출력 바이트 10을 통해 디바이스에 주소를 지정하고 입력 바이트 9를 통해 데이터 세트가 준비되었다는 신호를 PLC에 보냅니다.

활성화를 위해 지정된 코드에 출력 바이트가 설정됩니다. 그런 다음 장비의 데이터 세트가 준비될 때까지 WHILE 루프가 대기합니다.

프로그래밍:

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

4.6 스크립트 언어

참고

사용자 대화 상자 생성하기 (페이지 25) 함수에 설명된 모든 스크립트 요소는 "Easy Extend" 기능을 위한 기초를 구성합니다. 추가 디바이스 관리가 필요하면 스크립트 요소를 추가로 정의할 수 있습니다.

스크립트의 프로그램 부분

스크립트는 다음과 같은 영역으로 나뉩니다.

- "Easy Extend" 식별자
- 디바이스에 실행할 수 있는 작업을 정의하는 프레임
- 디바이스 식별자
- 디바이스 스타트업을 위한 식별자
- 디바이스 활성화를 위한 식별자
- 디바이스 비활성화를 위한 식별자
- 디바이스 테스트를 위한 식별자
- 파라미터 대화 상자 식별자

개별 태그는 다음 단원에 설명되어 있습니다.

설명

식별자 <태그>	의미
AGM	"스타트업 마법사"의 식별자
DEVICE	디바이스 설명을 위한 식별자. **속성:** • **option_bit** 옵션 관리를 위해 디바이스에 고정 비트 번호를 지정.
NAME	대화창에 표시될 디바이스의 이름을 지정하는 식별자. 텍스트 참조를 사용하는 경우 대화창은 이 식별자에 저장된 텍스트를 표시.

식별자 <태그>	의미
START_UP	디바이스를 스타트업하기 위해 필요한 작업 순서를 설명하는 식별자
SET_ACTIVE	디바이스를 활성화하기 위해 필요한 작업 순서를 설명하는 식별자 **속성:** • **timeout** 초 단위로 타임아웃을 지정할 수 있는 속성입니다. 이 시간이 경과한 후에도 스크립트가 완료되지 않으면 시스템이 처리를 중단합니다.
SET_INACTIVE	디바이스를 셧다운하기 위해 필요한 작업 순서를 설명하는 식별자 **속성:** • **timeout** 초 단위로 타임아웃을 지정할 수 있는 속성입니다. 이 시간이 경과한 후에도 스크립트가 완료되지 않으면 시스템이 처리를 중단합니다.
TEST	디바이스의 작동 성능을 테스트하기 위한 명령문이 포함된 식별자 **속성:** • **timeout** 초 단위로 타임아웃을 지정할 수 있는 속성입니다. 이 시간이 경과한 후에도 스크립트가 완료되지 않으면 시스템이 처리를 중단합니다.
UID	PLC ↔ HMI 인터페이스에서 디바이스를 식별하기 위한 고유 번호 식별자
VERSION	버전 식별자

부정적 기능 실행 결과의 승인

자동 제공되는 변수 "$actionresult"를 사용하여 시스템은 XML 구문분석기에 부정적인 실행 결과를 통보할 수 있습니다. 이 변수를 0으로 설정하면 구문분석기가 기능 처리를 중단합니다.

예제

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>                                   "스타트업 마법사"의 식별자
<DEVICE>
  <NAME> 디바이스 1 </NAME>             디바이스 식별자
  <START_UP>                            디바이스 스타트업을 위한 식별자
  …
  </START_UP>
  <SET_ACTIVE>                          디바이스 활성화를 위한 식별자
  …
  </SET_ACTIVE>
  <SET_INACTIVE>                        디바이스 비활성화를 위한 식별자
  …
  </SET_INACTIVE>
  <TEST>                                디바이스 테스트를 위한 식별자
  …
  </TEST>
</DEVICE>
…
</AGM>
```

4.6.1 CONTROL_RESET

설명

이 식별자는 하나 이상의 제어 콤포넌트를 다시 시작하도록 설정할 수 있습니다. 스크립트는 제어 시스템이 싸이클 작업을 재개한 후에만 계속 실행됩니다.

프로그래밍

식별자:	**CONTROL_RESET**	
구문:	<CONTROL_RESET resetnc="TRUE" />	
속성:	resetnc="true"	NC 콤포넌트가 재시작됩니다.
	resetdrive="true"	드라이브 콤포넌트가 재시작됩니다.

4.6.2 FILE

설명

식별자를 사용하여 표준 백업 파일을 읽거나 생성할 수 있습니다.

- 백업 파일 읽어오기:
  백업 파일을 읽어오려면 백업 파일의 이름을 명시해야 합니다.
- 백업 파일 생성:
  create= "true" 속성이 지정되면 지정된 이름으로 표준 백업 파일 (*.arc) 을 생성하고 이 파일을 .../dvm/archives 디렉토리에 저장합니다.

프로그래밍

식별자:	**FILE**	
구문:	<file name ="<archive name>" />	
	<file name ="<archive name>" create="true" class="<data classes>" group="<area>" />	
속성:	name	파일 이름 식별자.

create	지정된 이름으로 …/dvm/archives/ 디렉토리에 스타트업 백업 파일을 생성합니다.
group	백업 파일에 저장할 데이터 그룹을 지정합니다. 여러 개의 데이터 그룹을 저장하는 경우 각 그룹 사이에 공백을 넣어 구분해야 합니다. 백업 파일에 다음과 같은 데이터 그룹을 포함시킬 수 있습니다. • NC • PLC • HMI • DRIVES

예

```
<!-- Create data class archive -->
<file name="user.arc" create="true"
group="nc plc hmi" />

<!–백업 파일을 컨트롤로 리드인->
<file name="user.arc" />
```

4.6.3 OPTION_MD

설명

옵션 머신 데이터를 다시 정의할 수 있도록 하는 식별자입니다. 배송 당시 시스템은 MD14510 $MN_USER_DATA_INT[0]~$MN_USER_DATA_INT[3]을 사용하도록 설정되어 있습니다.

이 옵션들을 PLC 사용자 프로그램이 관리하는 경우 데이터 블록 또는 GUD에 적절한 데이터 워드를 제공해야 합니다.

데이터는 비트 단위로 구성됩니다. 나열된 디바이스에 비트 0부터 고정 비트를 지정합니다. 즉, 디바이스 1에 비트 0, 디바이스 2에 비트 1, 이런 방식으로 지정합니다. 16개 이상의 디바이스를 관리하는 경우 영역 인덱스를 통해 디바이스 그룹의 주소 식별자 1~3을 지정합니다.

참고

값 범위 변환

MD14510 $MN_USER_DATA_INT[i]의 값 범위는 -32768~+32767입니다. 머신 데이터 대화 상자를 통해 디바이스를 비트별로 활성화하려면 비트 조합을 십진수 형식으로 변환해야 합니다.

프로그래밍

식별자:	**OPTION_MD**	
구문:	영역 0:	
	<option_md name = "데이터의 주소 식별자" />	
	또는	
	<option_md name = "데이터의 주소 식별자" index= "0"/>	
	영역 1~3:	
	<option_md name = "데이터의 주소 식별자" index= "영역 인덱스"/>	
속성:	name	주소 식별자 (예: $MN_USER_DATA_INT[0])
	index	영역 인덱스 식별자
		0 (디폴트 설정): 디바이스 1~16
		1: 디바이스 17~32
		2: 디바이스 33~48
		3: 디바이스 49~64

4.6.4 PLC_INTERFACE

설명

PLC ↔ HMI 인터페이스를 다시 정의할 수 있도록 하는 식별자입니다. 시스템은 주소 지정이 가능한 128 워드를 필요로 합니다.

기본값: DB9905

프로그래밍

식별자:	**PLC_INTERFACE**	
구문:	<plc_interface name = "데이터의 주소 식별자" />	
속성:	name	주소 식별자 (예: "plc/mb170")

예:`plc/mb170`

4.6.5 POWER_OFF

설명

작업자에게 기계의 전원을 끄도록 요구하는 메시지의 식별자입니다. 메시지 텍스트는 시스템에 영구 저장됩니다.

프로그래밍

식별자:	**POWER_OFF**
구문:	<power_off />
속성:	--

4.6.6 WAITING

설명

NC 또는 드라이브를 리셋한 후 다시 시작될 때까지 대기시키는 식별자입니다.

프로그래밍

식별자:	**WAITING**	
구문:	<WAITING WAITINGFORNC ="TRUE" />	
속성:	waitingfornc="true"	NC가 재시작할 때까지 대기합니다.
	waitingfordrive="true"	드라이브가 재시작할 때까지 대기합니다.

4.6.7 대화창을 위한 XML 식별자

파라미터 설정 대화 상자

런타임 중 추가 파라미터를 설정 또는 출력할 수 있도록 각 디바이스에 대해 대화 상자를 설정할 수 있습니다. "추가 파라미터" 소프트 키를 누르면 대화 상자가 나타납니다.

사용자 대화 상자 생성하기 (페이지 25) 장에 설명된 모든 스크립트 요소는 대화 상자 생성에 사용할 수 있습니다.

예제

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE AGM>
<AGM>
<DEVICE>
  <NAME> Device 1 </NAME>
  <START_UP>
  …
  </START_UP>
  <SET_ACTIVE>
  …
  </SET_ACTIVE>
…
  <FORM name="<대화 상자 이름>">          사용자 대화 상자 식별자
    <INIT>
      <CONTROL name = "edit1" .../>     입력 필드 식별자
    </INIT>
    <PAINT>                             텍스트 또는 이미지 표시를 위한 식별자
      <TEXT>hello world !</TEXT>
    </PAINT>
```

```
  </FORM>

</DEVICE>
…
</AGM>
```

4.6.8 SOFTKEY_OK, SOFTKEY_CANCEL

설명

SOFTKEY_OK 식별자는 "OK" 소프트 키를 사용하여 대화창을 닫을 때 표준 동작을 덮어씁니다. SOFTKEY_CANCEL 식별자는 "CANCEL" 소프트 키를 사용하여 대화창을 닫을 때 표준 동작을 덮어씁니다.

이 식별자 내에서 다음과 같은 기능이 수행됩니다.

- 데이터 조작
- 조건부 처리
- 루프 처리

프로그래밍

식별자:	**SOFTKEY_OK**
구문:	<SOFTKEY_OK>
	…
	</SOFTKEY_OK>
식별자:	**SOFTKEY_CANCEL**
구문:	<SOFTKEY_CANCEL>
	…
	</SOFTKEY_CANCEL>

인덱스