



TM440
모션제어 - 기본 기능

I 버전 정보

버전	날짜	수정내역	번역	검수
1.0	2017.11.06	첫번째 버전 TM440TRE.00_V3090 (V1.0.3)	김동수	임은

Table 1: Versions

선행 및 필요 조건

교육 자료	TM410 – Working with Integrated Motion Control
소프트웨어	Automation Studio 3.0.90 Automation Runtime 3.08 ACP10_MC 라이브러리 2.280
하드웨어	None

II 목차

1	소개	1
1.1	학습목표	1
2	PLCopen standard	2
2.1	일반 정보	2
2.2	장점	2
3	PLCopen 모션 라이브러리	4
3.1	구조와 구성	4
3.2	평션 그룹	5
4	통합 제어 프로젝트	6
4.1	축 레퍼런스 사용하기	6
5	어플리케이션 기본	7
5.1	평션 블록 제어	7
5.2	드라이브 상태 개요	11
5.3	주기적인 축(Periodic axes) - 회전 축	13
6	프로그래밍 팁	17
6.1	제어 구조 사용하기	17
6.2	어플리케이션 프로그램에서 에러 핸들링	18
7	샘플 프로그램 사용하기	21
7.1	개요 - 샘플 적용	21
8	드라이브 파라미터 관리	23
8.1	초기화와 개별 파라미터 읽기	23
8.2	파라미터 전송과 설정 초기화	24
8.2.1	추가 정보	25
8.3	주기적으로 파라미터 읽고 쓰기	26
9	여러 개 단일 축 제어-소프트웨어 구성	27
10	요약	30

1 소개

어플리케이션 프로그램은 위치결정 작업에 기본이다.

이 곳에서 커맨드가 지정되고 드라이브로 전달되며 공정 신호가 평가되는 곳이다. 프로그램은 자동 시퀀스 공정에서 사용되는 드라이브를 제어한다.

이용 가능한 도구의 튼튼한 개요는 위치결정 작업 설정에 기초가 된다. 그러므로, 첫 번째 스텝은 드라이브 제어를 위해 사용되는 평션 블록의 전체 범위에 대한 명백한 개요를 얻는데 있다.

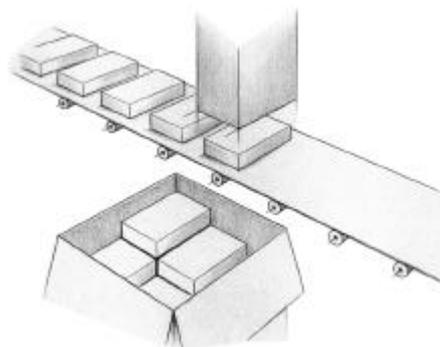


그림 1 Palletizing



이 교육 자료에서 우리는 어플리케이션 프로그램에서 통합 위치 결정 기능 사용법을 살펴볼 것이다.

다른 연습 문제를 통해 평션 블록의 행동을 이해하고 기능을 적용하는 중요한 기본 지식을 습득할 것이다.

PLCopen Motion Control standard 는 B&R 드라이브 솔루션의 컨셉을 운영하는데 중요한 역할을 한다.

그림 2 PLCopen Motion Control logo

1.1 학습목표

이 교육 자료에서 당신은 PLCopen 라이브러리를 어떻게 적용하고 활용하는지에 대한 개요를 배울 것이다.

아래 내용에 대해 배울 것이다.

- PLCopen 기능 사용 이점
- 당신의 프로젝트를 PLCopen functions 으로 개발
- 평션 블록 구조와 기능
- 드라이브 준비를 위한 기본 기능과 표준 동작 수행
- 전형적인 어플리케이션 프로그램 구조
- 샘플 프로그램 적용 프로세스

2 PLCopen standard

2.1 일반 정보

소프트웨어의 역할은 자동화 시스템에서 지속적으로 증가한다. 넓은 기능 범주 덕분에 어플리케이션의 복잡함은 점점 증가하고, 소프트웨어 개발과 유지보수는 점점 더 어렵게 만들고 있다. 게다가 시장에서 선택하는 솔루션은 매우 많다.

PLCopen 협회는 산업자동화 영역에서의 다른 영역, 요소, 도구를 표준화 하느라 바쁘다. 그 결과 다른 활동 영역들이 존재한다.

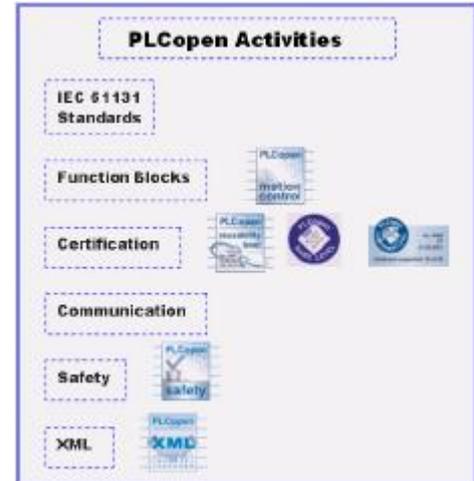


그림 3 PLCopen 활용 영역

PLCopen 협회와 활동에 관한 세부 정보는 인터넷 <http://www.plcopen.org/> 에서 찾을 수 있다.

다른 시스템 솔루션들이 모두 동일한 방식으로 동작하는 것을 보증하기 위해 가이드라인이 개발되었다.

협회에 가입되어 있는 모든 자동화 솔루션 공급자들은 PLCopen 을 사용하는 지정된 시스템을 위해 모두 동일한 소프트웨어 인터페이스를 제공한다. B&R 은 주요 멤버이다.

2.2 장점

제조회사와 독립적인 소프트웨어 개발

어플리케이션 프로그램은 하드웨어 독립적이다. 왜냐하면 표준화된 동일한 자동화 위치결정 기능 때문이다.

감소된 개발 시간

특수한 제공업체로부터 구체적인 시스템 솔루션을 아는 것은 더이상 필요하지 않다. 원칙적으로 이것은 기본 기능들을 올바른 방법으로 개발을 시작할 수 있다. 게다가 PLCopen 표준은 높은 유용성으로 구별된다.

간단한 프로그램 유지보수

PLCopen 덕분에, 우리는 어플리케이션 프로그램에서 깔끔하게 정렬된 구조를 제공하는 간단한 평선 블럭 사용 툴을 제공한다. 이렇게 하면 수정 사항 처리, 플랫폼 업데이트, 오류 찾는 것이 훨씬 쉬워진다.

B&R 의 완벽한 지원

위치결정 어플리케이션을 위한 PLCopen 표준은 B&R 드라이브 솔루션에서 이용가능하다. 표준화된 평선 블럭의 사용 덕분에 프로젝트 설정과 구성을 빠르고 쉽게 수행할 수 있다.

게다가 표준화에서는, 표준에 따라 준비된 B&R 의 특정 기능도 사용할 수 있다. 동일한 방식으로 B&R 드라이브 솔루션의 전체 기능을 사용할 수 있다.

이 평션 블럭들은 ACP10_MC 라이브러리에서 제공된다.



Programming W Libraries W Motion libraries W ACP10_MC

3 PLCopen 모션 라이브러리

PLCopen 모션 라이브러리는 Automation Studio 도움말에 목록이 있고 이 라이브러리 이름은 **ACP10_MC**이다. B&R 드라이브 솔루션을 제어하기 위한 표준화된 PLCopen 평선 블록을 포함한다.

표준화된 평선 블록 덕분에, 이 라이브러리는 B&R 특유의(B&R-specific) 확장을 포함한다.

3.1 구조와 구성

ACP10_MC 라이브러리는 어떤 시스템이든 사용될 수 있는 “절대 목표 위치로 위치결정”이나 “홈잉(Homing) 절차” 같은 기본 기능을 포함한다.

사용자에게 특정 분야의 표준 어플리케이션을 위한 완전히 확실적인 소프트웨어 인터페이스를 제공한다.

실제 B&R 드라이브 솔루션 기능 영역은 표준을 포함하는 것을 뛰어 넘는다.



그림 4 PLCopen Motion Control logo

예를 들면, 드라이브를 연결하기 위한 강력한 툴이 제공된다.(TM441 Motion Control: Multi-axis Functions 참조) 기본 위치결정 기능도 향상된 기능으로 사용가능하다.

ACP10_MC 라이브러리는 이 기능의 장점을 동일한 방식으로 유지하면서 사용자에게 제공할 수 있도록 B&R 특유의 기능을 포함하며 확장되었다.

확장된 기능은 표준 기능과 정확히 동일하게 작동된다.

PLCopen 표준 평선 블록이나 B&R 특유의 평선 블록은 이름으로 분류된다.

표준 평선 블록은 항상 이름 앞에 “MC_”라고 고정되어 시작한다.

(예: MC_MoveAdditive, MC_ReadAxisError)

B&R 특유의(B&R-specific) 평선 블록과 확장자는 “MC_BR_”로 시작하는 이름을 가진다.

(예: MC_BR_BrakeOperation, MC_BR_AutConrol)

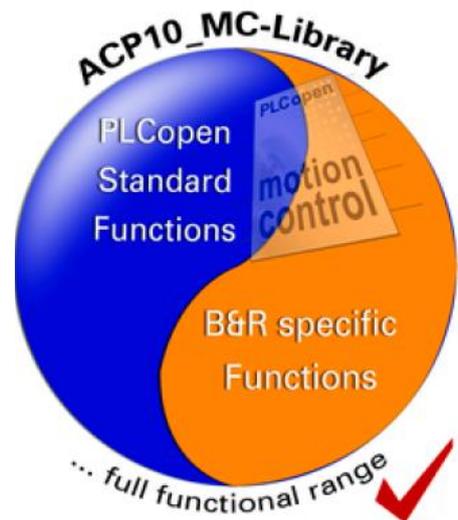


그림 5 ACP10_MC 라이브러리

3.2 평선 그룹

ACP10_MC 라이브러리는 상대적으로 많은 평선 블록을 포함하며, 그들의 사용과 기능에 따라서 대략 몇개의 그룹으로 나뉜다,

기본 기능을 위한:

- 드라이브 준비
- 추가적이고 절대적인 같은 표준 움직임
- 드라이브 상태 결정
- 설정 및 실제 값 읽기
- 드라이브 에러 인식과 결정
- 디지털 I/O 신호 제어 와 쿼리(Query)
- 위치 측정
- PLCopen 축 파라미터 관리
- 파라미터 IDs 관리

다 축 제어 기능을 위한:

- 전기(electronic) 기어박스 생성
- 캠 프로파일을 이용한 드라이브 연결
- 캠 프로파일 automat 설정 및 제어

기술적인 기능을 위한:

- 토크 제어
- 주기적인 설정 값 입력
- 프린트 마크 제어(Print mark control)
- 캠 프로파일 automat



Programming W Libraries W Motion libraries W ACP10_MC W Function blocks

Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Overview
of the supported function blocks

4 통합 제어 프로젝트

Automation Studio 에서 드라이브 구성을 위한 과정은 이전 교육 자료 (TM410)에 있다. 특수 명령어는 “NC Test” 진단도구를 사용하여 사전에 드라이브로 전달되었다.

아래 예는 축 동작 제어를 위한 어플리케이션 프로그램이 어떤 구조로 되어 있는지 보여준다.

첫 번째로 목표 축(axis object) 접근을 위한 참조 축(axis reference)이 필요할 것이다. 참조 축은 사전에 모션 위자드(Motion Wizard)를 수행하며 생성되었고 맵핑 테이블(mapping table)에 추가되었다.

?
 Motion W Configuration W Motion control W Creating an axis

4.1 축 레퍼런스 사용하기

위자드는 NC mapping table 에 동일한 이름을 가진 ACP10AXIS_Typ 타입의 global PV 를 생성한다. 그리고 연결은 파워링크(POWERLINK)을 통해 실제하드웨어와 소프트웨어 객체가 연결된다.

NC Object Name	Nc Obj.	Channel	Simulation	NC INIT Parameter	ACOPOS Parameter
gAxis01	ncAXIS	1	Standard	gaxis01i	gaxis01a
IF1.ST1 VAxis1	ncV_A	1			

그림 6 NC mapping, mapping table

참조 축의 주소(address)는 “Axis” 파라미터를 사용하는 모든 PLCopen 평선 블록들에 상속된다. 레더 다이어그램에서 어드레스 연결은 어드레스 평선 사용으로 대체될 수 있다.

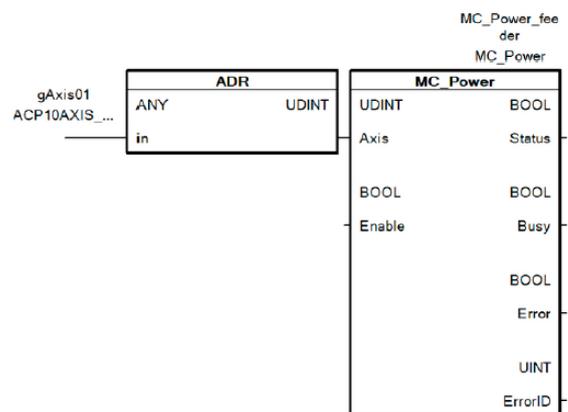


그림 7 축 레퍼런스가 MC_Power 평선 블록으로 전달된다.

?
 Programming W Libraries W Motion libraries W ACP10_MC W Concept W Implementation

5 어플리케이션 기본

이 장은 ACP10_MC 평션 블록 사용법에 대한 몇가지 기본 정보를 제공한다. 평션 블록 사용법 뿐만 아니라 모니터링 절차 사용 방법도 살펴 볼 것이다.

5.1 평션 블록 제어

모든 평션 블록은 동일한 동작과 상태 파라미터를 사용하여 접근한다.

모든 평션 블록의 표준화된 동작은 프로그램 사용에 있어서 깔끔한 개요를 보장한다.

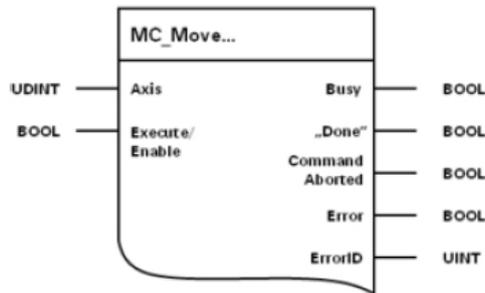


그림 8 표준 평션 파라미터

파라미터	설명
Axis	평션 블록 사용을 위한 축을 결정하는 참조 축을 구체화한다.
Enable	“enable” 입력 제어를 가진 평션 블록들은 입력 레벨에 따라 업무를 제어한다. 평션블록 입력 값이 TRUE 이면 값을 넘겨 받는다. 입력이 FALSE 이면, 액션이 완료되고 모든 결과가 리셋된다.
Execute	“Execute” 입력을 가진 평션 블록이 상승 엣지(positive edge)가 e” 입력에서 일어날 때, 입력 값을 읽고 그들의 업무를 실행시킨다.(예를 들어서, Execute 에 상승 엣지가 발생할때 새로운 속도가 적용된다.) 입력 받고 기능을 시작하면, 입력 리셋에 의해 일찍 끝날 수 없다. 다른 블록을 불러서(call) 인터럽트 걸거나 멈추게 한다.(예를 들어서 MC_Halt, MC_Stop, MC_Move, etc.)
Busy	각 액션이 성공적으로 시작되고 현재 실행 중인 것을 가리킨다. 출력이 살아 있는 한, 평션 블록은 계속 불러야 하나 상태 정보, 기능 등은 부정적인 영향을 받는다.
“Done“	각 평션 블록은 액션의 완성을 가리키는 상태 출력을 포함한다. 이 출력은 평션 블록에 따라 다르게 라벨링 되어 있으나, 항상 동일한 목적을 제공한다.
CommandAborted	다른 평션 블록 요구에 의해 중단된 커맨드 신호이다.
Error	평션 블록 요구 동안 에러가 발생한 신호이다.
ErrorID	에러가 발생하면, 해당 에러 넘버가 반환된다. 프로그램에서 발생한 에러 정보를 Automation Studio 도움말에서 볼 수 있다.

표 1 표준 파라미터와 상태 정보 설명



Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Error numbers

성공적인 명령 수행 과정

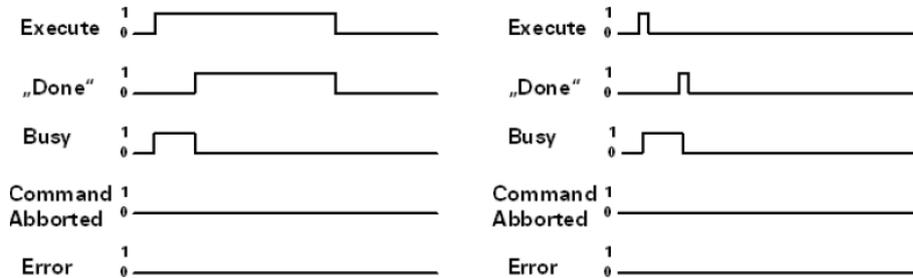


그림 9 성공적인 명령 수행 과정

과정은 상승 엣지(positive edge)가 Execute 입력에서 일어날 때 시작된다. 프로세스 활성화 명령은 Busy 출력을 통해 확인 가능하다. Done(또는 InVelocity, InGear 등)은 액션이 성공적으로 완료된 것을 나타낸다.

에러를 포함한 명령 수행 과정

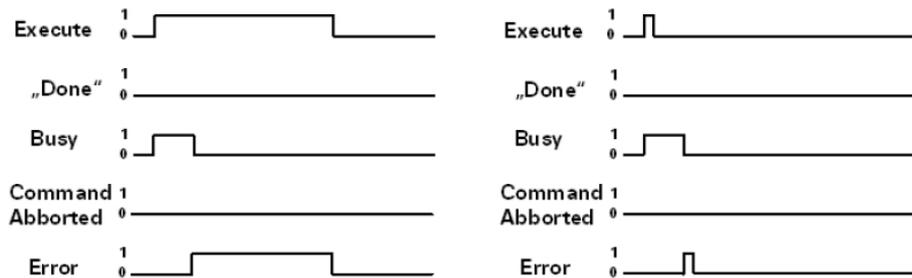


그림 10 에러를 포함한 명령 수행 과정

이 경우, 액션이 활성화된 후 에러가 발생한다. “Error” 출력이 활성화 되면 해당 에러번호는 “ErrorID”를 이용하여 나타난다.

인터럽트 받는 명령 수행 과정

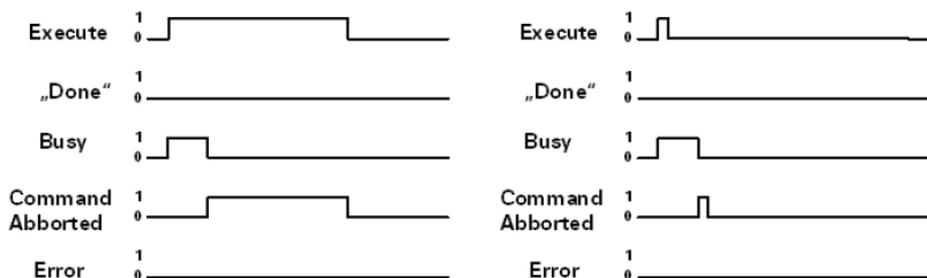


그림 11 인터럽트 받는 명령 수행 과정

이 경우, 커맨드가 다른 커맨드의 방해 받는 진행 과정이다. 사전에 실행된 펄스 블록(Busy=1)이 다른 펄스 블록 실행에 영향을 받는 경우이다.

예를 들면, 절대 동작(MC_MoveAbsolute)은 관련있는 동작에 의해 방해 받을 수 있다.
(MC_MoveAdditive)

상태 데이터를 읽기 위한 평선 블럭과 (위치, 파라미터 등) 움직임 제어를 위한 커맨드는 서로 영향을 미치지 않는다.

요약

상태 정보 Done, 명령어 Aborted, Error 와 ErrorID 는 Execute 입력이 리셋되기 전까지 동일하게 남아 있다.

만약 Execute 입력 신호가 도착하기 전에 이미 실행 중이면, 상태 결과와 에러 인디케이터는 1 사이클의 기간동안 설정된다:

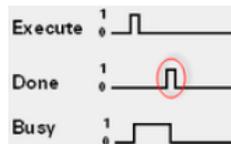


그림 12 Signal duration is equal to one cycle

에러가 발생해서 평선 블럭 출력을 리셋시키고 싶으면, Execute/Enable 입력을 FALSE 로 설정해야 하며, 에러 원인을 수정 후(에러는 반드시 인지해야한다 -acknowledge)에는 반드시 평선을 재시작해야 한다.

Enable 입력을 가진 평선 블럭은 입력이 설정되어 있는 한 활성화되어 있다. 반면 액션이 끝나고 평선 블럭 출력에 있는 상태 값은 리셋된다.



PLCopen 기능을 필요로 하는 평선 블럭은 프로그램 / cyclic 영역에서 불러야 한다(call). 하이 레벨 프로그래밍 언어에서, 프로그램 끝에서 평선을 부르는 것(call)을 추천한다.

이미지 기반 프로그래밍 언어에서, 네트워크가 프로그램 흐름을 뛰어넘게 된다면 특별한 케어가 반드시 수반되어야 한다.

올바른 접근 방법은 예제 프로그램에서 만나 볼 수 있다.(“샘플 프로그램 사용하기 “참조)

예제: 기본 ACP10_MC 평선 적용

ACP10_MC 라이브러리 평선 블럭은 어플리케이션 프로그램에서 빠르고 쉽게 사용될 수 있다. 드라이브는 절대 동작(absolute movement)으로 실행될 수 있어야 한다. 이를 위해 래더 다이어그램 언어를 사용하라.

- 1) 모션 위자드를 사용하여 축 추가
- 2) MC_Power 평선 블럭 추가
- 3) MC_Home 평선 블럭 추가
- 4) MC_MoveAbsolute 평선 블럭 추가
- 5) 모든 평선 블럭을 참조 축 연결
- 6) 프로그램 전송
- 7) CPU 재시작
- 8) 드라이브 켜기
- 9) 드라이브 휴밍
- 10) 절대 동작(absolute movement) 수행

예제: 현재 위치 읽기

MC_ReadActualPosition 평션 블록을 가지고 있는 드라이브의 현재 위치를 읽어라.



- Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Drive preparation
- Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Basic movements
- Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Determination of Axis State



참조 축은 반드시 모든 평션 블록에 연결되어야 한다. 항상 어드레스 기능을 불러올 필요 없이, 어드레스는 내부 변수로 저장될 수 있고 이를 평션 블록에 연결한다. 이 방법은 어플리케이션 프로그램에서 참조 축을 대체 하기 쉽다.

Init program

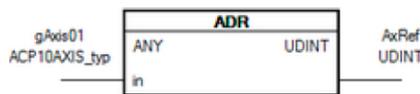


그림 13 참조 축에 주소 값 할당

Cyclic program

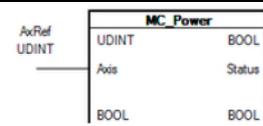


그림 14 참조 축에 연결

표 2 주소 값을 사용하여 참조 축 연결



MC_Power 평션 블록은 제어기와 전력 부품을 켜는데 요구된다. ACOPOSmulti 시스템에서 파워 서플라이 모듈도 스위치 켜는 것이 요구된다.



평션 블록 인스턴스 초기 값은 반드시 비어 있어야 합니다. 따라서 “permanent” 또는 “retain” 속성을 가진 변수는 사용할 수 없습니다.

5.2 드라이브 상태 개요

지정된 상태들은 드라이브를 작동시키기 위해 결정된다. 상태는 복잡한 동작 절차의 간단한 개요를 제공하고 에러 상황들을 쉽게 벗어나게 해준다.

상태	설명
Disabled	드라이브 제어가 꺼진다.
Standstill	드라이브는 현재 동작이 실행되지 않고 위치결정을 위한 명령 대기 중이다.
Homing	드라이브는 홈잉 절차를 수행 중이다.
Errorstop	드라이브는 에러 발생 후 정지 상태이다.
Stopping	드라이브는 움직임을 멈추고 있다.
Discrete motion	드라이브는 지정 위치로 동작 중이다. 그 동작은 지정된 목표를 가진다.
Continuous motion	드라이브는 타겟 위치 없이 움직이고 있다. 그 동작은 지정된 목표가 없다.
Synchronized motion	드라이브는 다른 드라이브와 동기되었다.

표 3 드라이브 상태 개요

이 상태들 사이의 전환은 특정 명령을 호출하면서 초기화된다.

아래는 시퀀스 예시이다.

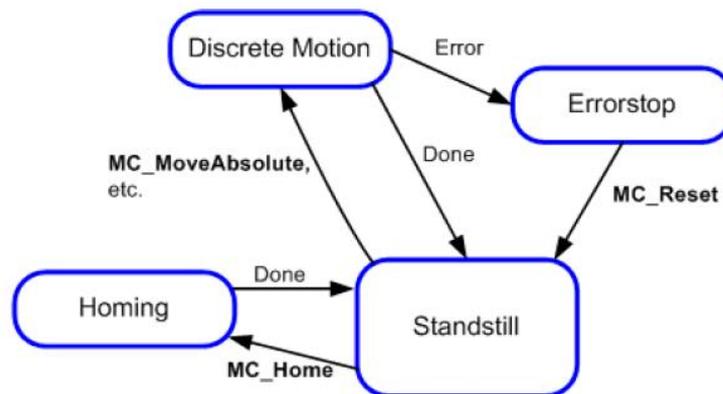


그림 15 상태 진행 단계

축이 대기 상태(Standstill)에 있다고 가정해보자. 홈잉(Homing) 절차를 성공적으로 수행하자마자 MC_MoveAbsolute 명령은 동작을 시작하는데 사용될 수 있다.

타겟 위치에 도달한 후, 드라이브는 “Initial state”(초기 상태)로 반환한다. 만약 드라이브 에러가 위치결정 액션 중에 발생한다면 축은 에러 상태로 들어간다(Errorstop). 드라이브 에러가 고쳐진 후 MC_Reset 기능을 사용하여 인지될 수 있다.

완전한 모든 상태 개요:

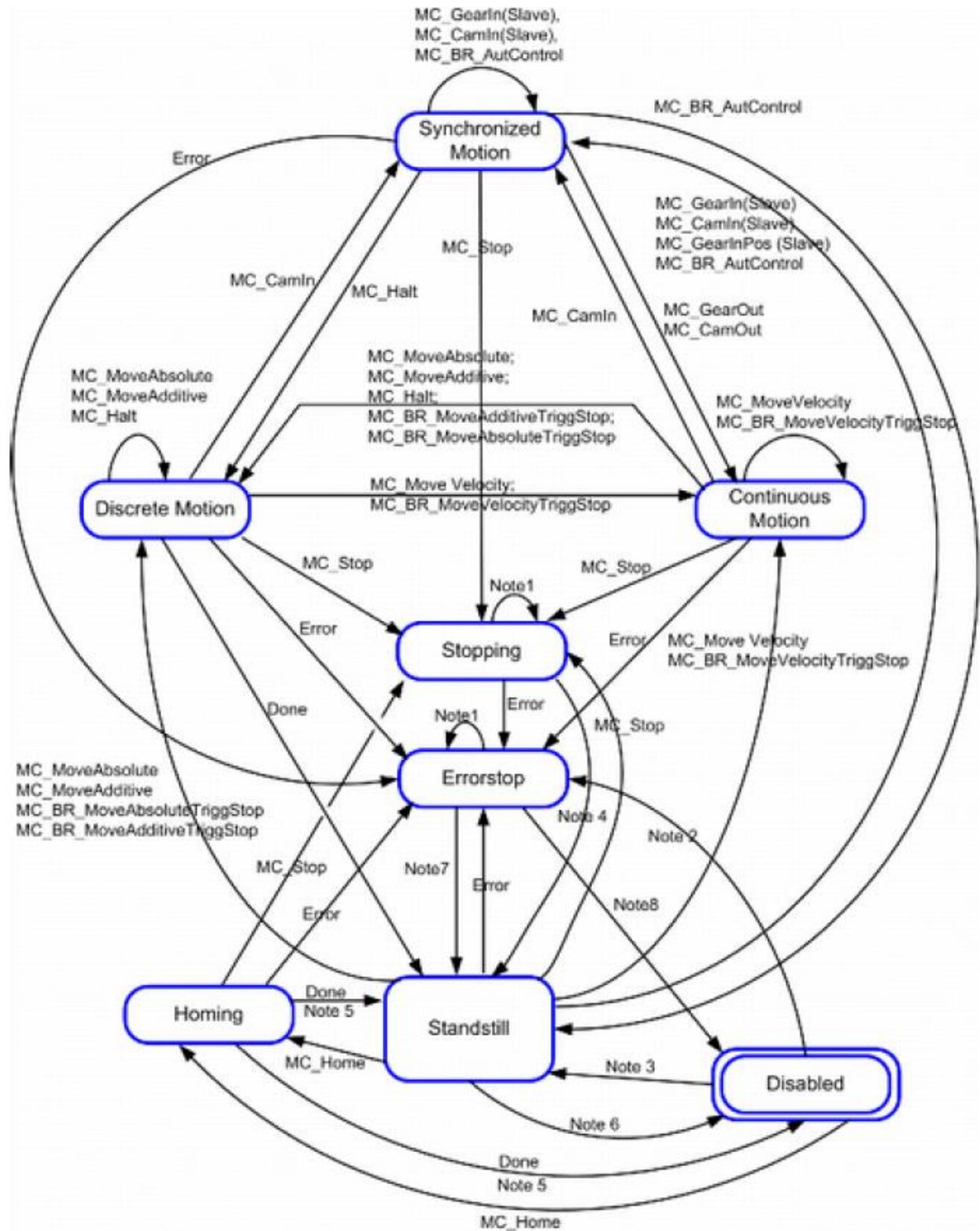


그림 16 PLCopen 모션 제어 다이어그램의 상태



Programming W Libraries W Motion libraries W ACP10_MC W Concept W State diagram

가장 중요한 드라이브 상태는 다이어그램에 포함되어 있다. 그들은 위치결정 시퀀스를 조정하기 위해 사용될 수 있다. MC_ReadStatus 펄스 블록은 현재 축 상태를 읽는데 사용된다.



가상 축은 드라이브 제어기에 포함하지 않는다. 실제 축과 달리 가상 축은 Disabled 상태가 없다. NC 오프젝트 “가상 축”은 대기 상태(Standstill)에서 시작하고 활성화를 위한 MC_Power 펄스 블록이 필요하지 않다.

예제: 상태 모니터링

현재 드라이브 상태는 MC_ReadStatus 펄스 블록을 사용하여 결정될 수 있다. 이 펄스를 당신의 테스트 프로그램에 추가하고 다른 위치결정 액션이 실행될 때 상태 변화를 모니터링 하라.

5.3 주기적인 축(Periodic axes) – 회전 축

위치 값들을 조정을 위해 추가적인 설정을 할 수 있다.

위치 스케일링 뿐만 아니라 주기적인(Periodic) 위치 동작은 간단한 구성을 사용하여 만들 수 있다. 예를 들어서 이 설정은 턴테이블(회전반) 어플리케이션에 사용될 수 있다.

첫 단계를 위해 무엇이 필요한가?

유닛 스케일링 당 회전수 기본 설정은 축의 엔코더 인터페이스 파라미터를 사용하여 만들어진다.

Name	Value	Unit	Description
scaling			Scaling
load			Load
units	3600	Units	Units at the load
rev_motor	5		Motor revolutions

그림 17 초기 파라미터 모듈, 엔코더 인터페이스

상기 이미지에서 3600 유닛(units)을 다섯번의 축 회전으로 나누면, 1 회전당 720 유닛이 된다. 이것은 요구 사항에 따른 회전 수로 나눌 수 있다(엔코더 최대 해상도 유지) DINT 는 위치결정을 위한 데이터 타입(data type)이다.

모든 PLCopen 펄스 블록은 위치결정 스펙을 위해 REAL 데이터 타입을 사용한다.

위치 주기(position period)와 위치 팩터(position factor) 값은 NC 맵핑 테이블(NC mapping table)에 위치 값 적용을 위해 PLCopen_ModPos="<Period>,<Factor>"로 기입할 수 있다.

NC INIT Parameter	ACOPOS Parameter	Additional Data	Description
gaxis01i	gaxis01a	PLCopen_ModPos="<period>,<factor>"	

그림 18 NC mapping table, advanced settings

Axis period

연속적인 축 움직임을 위해, 위치 값을 주기적으로 결정하는 것은 종종 중요하다. 주기에서 0 보다 큰 값이 사용되었다면, 위치 값은 이 입력에 따라 적용된다. 그것은 엔코더 파라미터에 있는 축 스케일링을 직접 나타낸다. 모든 PLCopen 평션 블록은 주기적인 위치와 함께 “작업”한다.

예를 들어, <Period> = 1000 이라면, 0 으로 리셋되고 999 까지 다시 증가하기 전에 positive 움직임을 하는 동안은 위치 값은 0 부터 999 까지 계속 증가할 것이다.

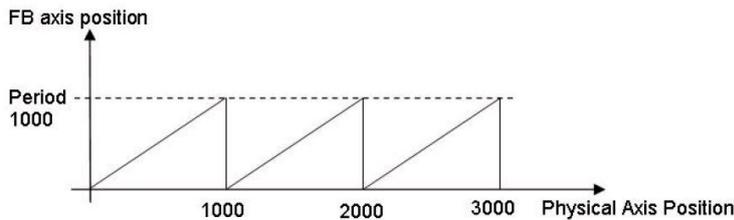


그림 19 주기적인 축 위치

주기적 위치 변환은 <Period> = 0 이 입력되면 비활성화 될 수 있다. 주기적 운동이 요구되지 않더라도 특정 팩터는 여전히 사용될 것이다. 이 경우, 축 동작 범위는 ±8,388,608 유닛(units, ±2²³)까지 제한된다. 왜냐하면 이것은 REAL 데이터 타입에서 정확성을 잃지 않고 디스플레이 될 수 있는 가장 큰 값이기 때문이다.

? Programming W Libraries W Motion libraries W ACP10_MC W Concept W Implementation W Axis period

Axis factor

PLCopen 평션 블록은 축 위치결정을 위해 REAL 데이터 타입을 사용한다. 이 데이터 타입은 소수점 자리를 사용할 수 있으며, 간단하고 매우 매우 흥미롭게 스케일링을 변환할 수 있다.

“스케일링(**scaling**)”이란 정확히 무엇인가?

스케일링은 변환이라고도 한다. 이미지로 표현된 아래 예제가 설명해준다.

특정 어플리케이션은 1 μm이하의 정확도의 위치결정을 요구한다. 우리가 설정한 파라미터가 위치결정 유닛 당 1 μm거리로 설정 할 수 있다고 가정하자.

지금 우리가 PLCopen 평션 블록으로 위치결정 업무를 밀리미터(mm) 단위로 구체화 할 수 있다면 이득일 것이다. 그리고 이 팩터로 얻은 것들은 정확하다. 방정식은 아래와 같다.

$$PLCopenUnits = \frac{AxisparameterUnits}{Factor}$$

우리가 factor 를 1000 으로 설정할 수 있다면, 우리는 밀리미터(mm)까지 스케일링 할 수 있을 것이다. 지금 우리가 PLCopen 평션 블록에 상응하는 1 값 거리로 움직인다면, 우리 축은 실제로는 1000 축 파라미터 유닛으로 움직일 것이다.

<Factor>	PLCopen units [REAL]	Axis parameter units [DINT]
1	1	1
1000	1	1000
1000	0001	1

표 4 설정 factor 에 따른 축 유닛과 PLCopen 유닛 비교



속도와 가속도 값은 이 스케일링을 참조한다.



Programming W Libraries W Motion libraries W ACP10_MC W Concept W Implementation W Axis factor

예제: 회전 축 구성

간단한 예는 아래 설정을 실행하고 가능성을 증명 할 것이다.

테스트 정의:

피벗팅 캐리어(A pivoting carries)는 반드시 제품 프로세싱을 위해 다른 스테이션으로 움직여야한다. (specific angular positions 360° 회전 내)

위치 결정은 0.1° 이내의 정밀도가 되어 한다. MC_MoveAbsolute 펄스 블록은 위치 접근을 위해 사용될 것이다.

이 절차를 조금 더 쉽게 하기 위해 위치는 각도로 구체화된다.(소수점 첫째 자리)

```
MoveAbsolute_0.Position := 135.0; (* goal: 135° *)
```

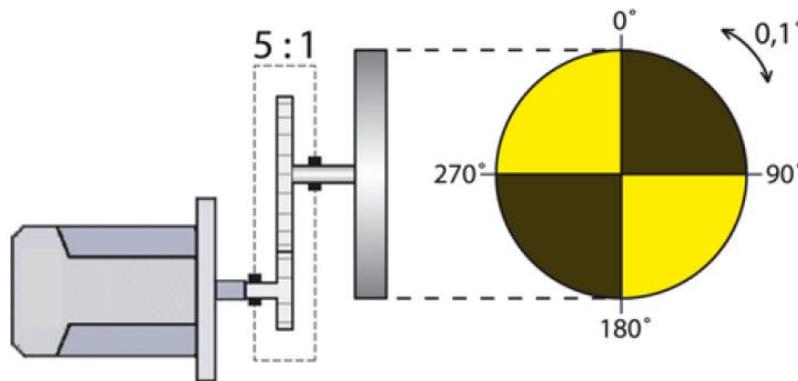


그림 20 기구 구조 스케치

캐리어는 기어로 연결된 서보 모터로 움직인다. (기어비 = 5:1)

적합한 엔코더 값과 예정된 위치 설정 값을 찾아라.

가능한 해결책:

엔코더 기본 설정은 초기화 파라미터 모듈(Init parameter module)에서 설정한다. 그러므로 0.1° 스텝 정밀도를 가진 위치를 위해서, 회전 캐리어의 1 회전당 3600 유닛이 요구된다. 이 유닛은 5 번 모터 회전수 이상 분배된다.

Name	Value	Unit	Description
scaling			Scaling
load			Load
units	3600	Units	Units at the load
rev_motor	5		Motor revolutions

그림 21 Init parameter module, encoder interface, configuration

3600 유닛 동작이 현재 실행되면, 모터는 정확하게 5 회전을 수행하고 캐리어는 정확하게 360° 회전한다. 이제 기어는 완전히 설정되었다.

위치 결정 펄스를 위해 요구되는 스케일링을 얻기 위해서, (엔코더 셋팅에 직접 기반한) Period 는 3600 을 설정하고 factor 는 10 으로 정의되어야 한다.

```
Additional Data  
PLCopen_ModPos="3600,10"
```

그림 22 맵핑 테이블에 주기적인 축 응용 설정

6 프로그래밍 팁

어플리케이션 프로그램은 드라이브 제어를 위한 자동 시퀀스를 만드는 방법이다. 이 과정에서 모든 평선 블록을 프로그램에서 불러내는 것이 중요하다. 에러 이벤트는 반드시 고려해야 하며, 필요하다면 적절하게 응답해야 한다.

어플리케이션 프로그램에서 구조화된 시퀀스를 위해, 기계 상태는 하이 레벨 언어를 사용 할 수 있다.



그림 23 어플리케이션 프로그램 상세하기 살펴보기

6.1 제어 구조 사용하기

각각의 평선 블록은 필요한 만큼 실행될 수 있다. 프로세스는 명령을 사용하여 프로그램에서 지정된 위치에서 시작될 수 있다.

상태 출력과 출력 파라미터는 현재 상태에 대한 정보를 제공한다:

- 실행이 성공적인가?
- 그렇지 않다면 에러가 발생하였는가?
- 프로세스 상태:
 - 축이 동작 중인가?
 - 타겟 위치에 축이 도달하였는가?
 - 홈인 절차가 성공적인가?

이 정보는 프로그램 시퀀스에서 다음 스텝을 제어하는데 사용된다. 만약 에러가 발생하면, 프로그램은 에러 발생에 따라 다르게 반응해야 한다.

스텝 시퀀서는 이러한 평선 시퀀스 타입을 관리하는데 매우 적합한 제어 구조이다.

이러한 구조는 스텝 인덱스 사용에 의해 결정될 수 있는 각 스텝 시퀀스 실행을 허용한다.

다이아그램은 이러한 제어 구조를 적용한 가능성 있는 디자인을 보여준다.

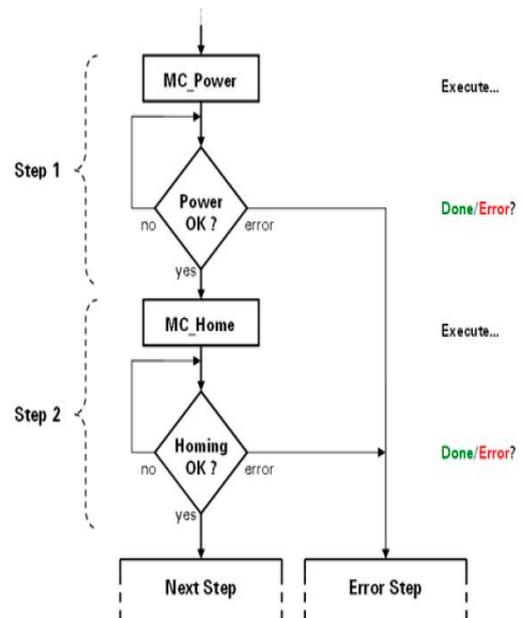


그림 24 제어 구조 및 구조화 프로그램 예시

평선 블록은 제어구조 각 스텝에서 Execute로 실행될 수 있다. Error, Done, ErrorID 같은 상태 파라미터를 통해 어플리케이션이 지속되는 스텝을 결정할 수 있다.

이는 어플리케이션에 깔끔한 디자인을 제공하고 향후 확장성을 개방시켜준다.

이 컨셉을 사용하는 샘플은 Automation Studio 에 포함되어 있다. (“샘플 프로그램 사용하기”)

6.2 어플리케이션 프로그램에서 에러 핸들링

어플리케이션을 프로그래밍할 때, 프로그램 시퀀스에서 에러 평가를 고려하는 것은 항상 중요하다. 중요한 질문은 ‘어떤 에러를 고려해야 하는가?’ 이다.

두 가지 에러 타입이 있다:

- 펄스 블럭을 불러올 때 에러
- 드라이브 에러



그림 25 어플리케이션 프로그램에서 에러 평가

?	<p>Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Error numbers</p> <p>Motion W Reference manual W ACP10 W ACOPOS error texts</p>
---	--

어떻게 에러를 모니터링 할 수 있는가?

PLCopen 펄스 블럭은 상태에 대한 직접적인 피드백을 제공한다. 실행하는 동안 발생한 에러는 에러 상태 출력에 표시 된다. 에러가 발생하면 해당 에러번호가 더욱 정확한 에러를 국부화하기 위해 펄스블럭의 ErrorID 를 나타낸다.

MC_ReadAxisError 펄스 블럭은 드라이브 에러를 모니터링 하는데 사용된다.

!	<p>펄스 블럭을 불러올 때 에러가 발생하면, 에러 상태가 남아있기 때문에 다음 사용 전에 블럭을 리셋시키는 것이 절대적으로 필요하다. 이는 Enable=0 또는 Execute=0 으로 펄스 블럭을 실행함으로써 완료된다.</p>
---	---

에러 평가에 대한 가능성 있는 접근:

드라이브 에러 모니터링을 위해 주기적으로 프로그램에서 MC_ReadAxisError 평선 블록 사용을 추천한다.

평선 블록은 Enable=1 일때 즉시 동작되어야 한다. 필요한 요청(Error?)은 cyclic 프로그램 영역에 위치한다.(예: 제어 구조 이전)

드라이브에 에러가 발생될 때, 규칙적인 프로그램 수행은 정지되어야하며 에러 핸들링 위한 스텝에 들어가야 한다.

실제 평선 블록을 부르는 동안 에러가 발생하는 것 또한 동일하다. 이 상황을 다루기 위한 “error step”이 실행되어야 한다.

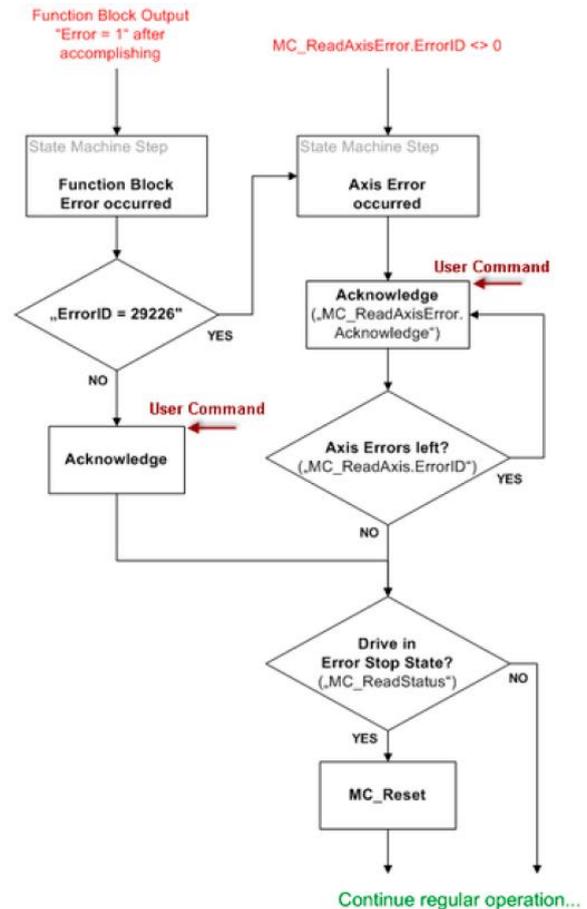


그림 26 위치 제어 어플리케이션의 에러 핸들링 절차



부가적으로 MC_ReadAxisError(MC_BR_ReadAxisError, MC_BR_AxisErrorCollector) 평선 블록은 STRING 변수를 사용하여 현재 드라이브 에러와 관련된 정보를 제공한다.

이 옵션 사용을 위해 “NC error text table“ 객체는 프로젝트에서 원하는 언어로 관리될 필요가 있다. 이 모듈 이름은 평선 블록과 연결되어야 한다. 평선 블록이 수행되면, 현재 에러 넘버에 해당되는 텍스트를 평선 블록에 명시된 STRING 변수로 전달한다. 스텝 시퀀서가 축을 제어하기 전에 에러 평가 운영을 추천한다. 축 에러를 1 사이클 이후가 아닌 즉시 어플리케이션에서 읽는다.

에러 평가는 일련의 단계에서 발생할 수 있다:

- 순차적으로 드라이브 에러가 확인되고 인지될 수 있다. 이 평선 블록은 마지막 에러가 인지될 때까지 다른 에러를 하나씩 반환한다.
- PLCopen 평선 블록 에러가 발생하면, 시스템은 먼저 드라이브에 의해 오류가 발생한 것인지 확인한다 (ErrorID 29226). 그 다음 에러를 인지하고 에러 스텝에서 나가게 해준다.
- 두 개 루틴이 실행된 후, 드라이브 상태를 확인한다. 특정 경우에, 드라이브는 “Error stop” 상태로 설정될 수 있다. 이 상태는 리셋(MC_Reset)이 실행되기 전까지 지속될 것이다.(“드라이브 상태 개요” 참조)

어플리케이션은 모든 에러가 조치되고 인지되어야만 지속될 수 있다.



이 프로그램 시퀀서는 에러 처리 구현하기 위한 지침으로 간주되어서는 안된다. 대신, 이 것은 단지 보편적인 접근이거나 이러한 로직 타입 구현에 기본이다. 상세한 정보는 샘플 프로그램에서 확인 할 수 있다. (참조 “[샘플 프로그램 사용하기](#)”)

게다가, Automation Studio 설치시 포함되어 있는 다축 제어 샘플 프로그램은 에러 핸드링 구현을 위한 다양한 아이디어를 제공한다. (“ErrorHandling” 프로그램)



Programming W Examples W Motion W Motion control W Muti-axis operation

7 샘플 프로그램 사용하기

Automation Studio 설치시 ACP10_MC 라이브러리에 가장 중요한 평션을 사용하는 위치 결정 어플리케이션용 샘플 프로그램이 포함되어 있다. 이미 축이 구현된 프로젝트에 기능 어플리케이션 섹션을 추가할 수 있다.

기본 동작 타입은 부가적인 프로그래밍 없이 제어 어플리케이션에서 다를 수 있다.



Programming W Examples W Motion W Motion Control

7.1 개요 - 샘플 적용

Automation Studio 설치 시 드라이브 제어를 쉽고 빠르게 사용할 수 있도록 구현된 몇 개의 샘플 프로그램도 같이 설치된다.

샘플은 리더 다이어그램, Structure Text, ANSI C 프로그래밍 언어로 제공한다.

축, 축 연결(axis linking), 캠 프로파일(cam profiles)과 다른 어플리케이션을 위한 샘플도 이용가능하다.

샘플 추가하기

이용 가능한 샘플은 Automation Studio 프로젝트 / 로직컬 뷰 / 위자드를 이용해서 추가할 수 있다. 위자드에는 **Samples** 영역의 하위 그룹 **Motion samples** 가 있다.

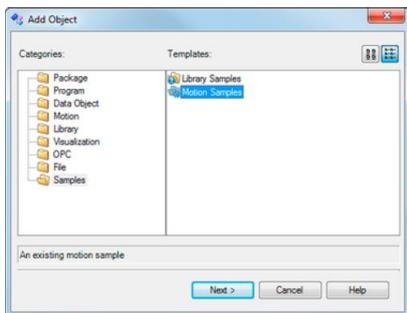


그림 27 Automation Studio 프로젝트에 모션 샘플 추가

모든 요구 사항은 Automation Studio 프로젝트에 추가되고, 샘플 프로그램은 로지컬 뷰에 분리된 패키지에 저장된다.

Object Name	Description
plcOpenLibrary	Standardprojekt
Global.typ	Global data types
Global.var	Global variables
Libraries	Global libraries
gAxis01obj	
acp10etxen	
LibACP10MC_SingleAx_LD	
Basic	Basic logical control
basicCyclic.Id	cyclic code
basicInit.Id	int code
basic.typ	Local data types
basic.var	Local variables
basic.fun	
functions.st	
powerOff.Id	action
AxisErrorIDUpdat...	action
AxisError.Id	action

그림 28 별도 패키지에 저장된 샘플 프로그램

이 샘플의 정확한 개요와 이름은 Automation Studio 도움말에서 찾을 수 있다.



Programming W Exampels W Motion W Motion control

샘플 기능

샘플들은 예전에 거론되었던 프로그래밍 추천에 따라 설정된다. (“프로그래밍 팁” 참조)
큰 구조에서는 기계를 동작시킬 수 있는 명령, 축 파라미터, 상태 메시지의 하위 구조가 있다. 참조 축은 샘플의 초기화 서브 프로그램(Init subprogram)에서 수정할 수 있다.
필요 스텝의 설명과 제어 구조를 디자인 하는 방법은 Automation Studio 도움말 문서에서 샘플 별로 찾아볼 수 있다.

예제: 샘플 프로그램 사용하기

- 1) ST 언어로 되어 있는 기본 동작을 위한 샘플 프로그램을 추가하라.
- 2) Automation Studio 도움말 문서에서 설명 확인
- 3) 로직컬 뷰에 샘플 추가
- 4) 샘플 프로그램에서 필요한 만큼 초기화 서브 프로그램에 있는 참조 축 수정
- 5) 어플리케이션 업로드
- 6) CPU 재시작
- 7) 커맨드를 테스트하고 샘플프로그램의 결과 확인

8 드라이브 파라미터 관리

몇 개의 평션 블록은 드라이브 파라미터를 관리하기 위한 ACP10_MC 라이브러리를 제공한다. 이 파라미터들은 위치결정 테스크를 핸들링 하는 것 뿐만 아니라 드라이브를 설정하는데 사용될 수 있다. 일반적으로 이 파라미터들은 NC operating 시스템에 의해 관리된다. 그러나 어떤 어플리케이션들은 런타임 동안 다른 파라미터가 요구되는 접근(읽고 쓰기)을 직접적으로 한다.



ParIDs 를 필요한 만큼 조정하여, 드라이브 구성은 특정 상황을 핸들링하는데 적용될 수 있다.

이 평션은 B&R 드라이브 솔루션에만 제공된다. 즉, 그들은 특수한 B&R 평션 블록을 사용하여 수행된다. 이 평션 블록들(operation, status check 등)의 사용은 여전히 PLCopen Motion Control 표준을 준수한다.



Motion W Reference manual W ACP 10 W ACOPOS parameter IDs

8.1 초기화와 개별 파라미터 읽기

개별 파라미터는 한번 또는 주기적으로 초기화 되고 읽을 수 있다.

한번에 초기화되고 읽히는 평션 블록들:

- MC_BR_WriteParID
- MC_BR_ReadParID

개별 ParID 는 사이클릭 트랜스퍼를 설정하여 자동으로 각 TC#1 사이클로 초기화되거나 읽을 수 있다:

- MC_BR_InitCyclicWrite
- MC_BR_InitCyclicRead
- MC_BR_CyclicWrite
- MC_BR_CyclicRead



초기화는 드라이브에 있는 파라미터를 특정 값("Operational")을 만드는 것을 의미한다. 초기화는 전송 이후 항상 자동으로 발생되지 않는다. 어떤 평션들은 오직 파라미터 값을 드라이브로 전송한다. 이 경우에, 초기화는 이후에 수행될 수 있다.

POWERLINK 노드 간의 통신을 위한 평션 블록:

- MC_BR_InitSendParID
- MC_BR_InitReceiveParID
- MC_BR_ReceiveParIDOnPLC

별도의 평션 블록은 개별 ACOPOS 디바이스 간의 ParID 를 위한 독립적인 주기적인 통신을 설정할 수 있다.:

- MC_BR_InitMasterParIDTransfer

이 평션 블록은 캠 프로파일 오토맷(cam profile automat)의 추가적인 축 과 Smart Process Technology 평션을 위해 사용된다.



Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Administration of ACOPOS ParIDs

8.2 파라미터 전송과 설정 초기화

드라이브 파라미터 관리를 위한 다른 기능 블록들은 다수의 ParID 를 전송하고 초기화를 지원한다.

이를 위한 세가지 방법이 있다.

- 파라미터 테이블 (Parameter tables)
- 파라미터 리스트 (Parameter lists)
- 파라미터 시퀀스 (Parameter sequences)

파라미터 테이블 (Parameter tables)

ACOPOS 파라미터 테이블에 있는 파라미터들은 각 NC Manager task 사이클에 있는 드라이브로 독립적으로 전송되고 **MC_BR_InitParTableObj** 평션 블록을 사용하여 즉시 초기화된다.

Name	ID	Value	Unit	Description
Parameters				
CMD_SIMULATION	110	ncSW_ON		Simulation mode: Command
MOTOR_TEST_MODE	866	7		Motor: Test mode
PHASE_MON_IGNORE	80	1		Power mains: Ignore phase failure
EDC_NOMINAL	390	24	V	CTRL DC bus: Nominal voltage

그림 29 ACOPOS 파라미터 테이블

파라미터 리스트 (Parameter lists)

파라미터 설정과 값 할당은 파라미터 리스트를 사용하여 런타임 동안 쉽게 조정될 수 있다.

파라미터 ID 요소를 포함하는 목적에 추가적인 데이터 타입이 있다. 프로그램에서 배열로 이 데이터 타입을 사용할 수 있다. 리스트의 항목은 런타임 중 응용 프로그램 내부에서 변경 될 수 있다.

ACOPOS 파라미터 테이블과 같이, 파라미터 리스트에 있는 모든 파라미터들은 각 NC Manager 테스트 사이클에서 드라이브로 전송되고 **MC_BR_InitParList** 평션 블록을 사용하여 즉시 초기화 된다.



사전 정의된 ACOPOS 파라미터 테이블을 사용하는 초기화 변경과 달리, 파라미터 배열을 사용하면 어플리케이션이 동작하는 동안 배열 내부에 포함된 파라미터와 해당 값을 모두 변경할 수 있다.

파라미터 시퀀스 (Parameter sequences)

파라미터 시퀀스는 파라미터 리스트와 동일한 배열을 사용한다.

하지만 ACOPOS 파라미터 테이블 혹은 파라미터 리스트와 달리, 파라미터 시퀀스의 파라미터들은 NC Manager 아이들 타임 테스트 동안 데이터 블록(개별적으로가 아니라)으로 드라이브에 전송된다(**MC_BR_DownloadParSequ**). 한번 발생하면, 즉시 초기화 되지 않고 “recipe”로서 저장된다.

이 파라미터들은 “Initialize sequence” 커맨드(**MC_BR_InitParSequ**)로 활성화 될 수 있다.

각 파라미터가 인덱스로 할당되기 때문에, 드라이브에 다수의 파라미터를 설정하고 선택적으로 초기화하는 것이 가능하다.

?
 Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Administration of ACOPOS ParIDs

8.2.1 추가 정보

파라미터 리스트 혹은 파라미터 시퀀스에서 파라미터 그룹을 위해 각 펄스 블록은 “DataAddress” 입력 파라미터를 포함한다. ACP10DATBL_typ 구조체 변수는 여기서 여전히 ‘중개자(intermediary)’로 제공되어야 한다.

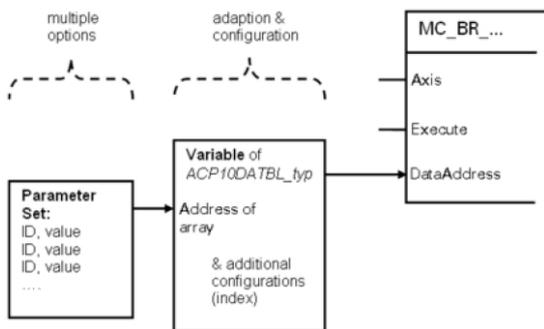


그림 30 ACP10DTBL_typ 변수로 연결된 파라미터 그룹

다이아그램은 이들의 간략한 관계를 보여준다. 파라미터 그룹은 파라미터 배열로 왼쪽에 위치한다. 이 그룹은 전송을 위한 변수 구조(ACP10DATBL_typ)를 통해 펄스 블록으로 연결된다. 이 파라미터들은 올바르게 할당을 하면(“Execute=1”), 다운로드 할 수 있다.

8.3 주기적으로 파라미터 읽고 쓰기

어플리케이션에 따라서, 부가적인 드라이브 정보 또는 제어 프로그램에서 페루프 드라이브 제어 변경이 요구될 것이다. 전형적으로, 이는 주기적으로 파라미터를 읽고 쓰는 것으로 수행된다. 이전 장에서 이미 한가지 방법이 설명되었다. (“초기화와 개별 파라미터 읽기” 참조)

파라미터 IDs 를 읽고 쓰는 평션 블록 뿐만 아니라, 직접 연결된 평션 블록도 있다.

예제 : 샘플 프로그램 확장 - 읽기

아래 기능을 포함하여 샘플 프로그램을 확장하라:

- 모터 전류 읽기
- 현재 모터 온도 읽기
- 현재 토크 읽기

ParID	설명
214	Actual stator current of the quadrature component
381	Temperature sensor: Temperature
277	Motor: Torque

표 5 읽을 드라이브 파라미터 개요

예제 : 샘플 프로그램 확장 - 쓰기

아래 기능을 포함하여 샘플 프로그램을 확장하라:

- MC_BR_WriteParID 로 토크 제한
- Override 그룹(ncPAR_TYP_VOID)의 설명을 위한 USINT[1..4] 배열 생성

ParID	설명
343	CTRL torque limiter: Override



4 개의 파라미터 모두가 쓰여지기 때문에 데이터는 ncPAR_TYP_VOID 데이터 타입의 배열로 전송되어야 한다.

주기적인 설정 값 정의

어떤 어플리케이션은 설정 값이 제어기에서 계산되는 것을 요구한다. 설정 값을 따르는 축은 반드시 주기적으로 이 데이터를 공급해야 한다

아래 평션 블록은 이 목적으로 이용 가능하다.

- MC_BR_MoveCyclicPosition
- MC_BR_MoveCyclicVelocity
- MC_BR_VelocityControl



Programming W Libraries W Motion libraries W ACP10_MC W Function blocks W Cyclic set values

9 여러 개 단일 축 제어-소프트웨어 구성

제어 프로젝트에서는 종종 다 축 동작이 필요하다. 로직컬 뷰에서 각 드라이브 축을 위한 분리된 프로그램 생성이 가능할지라도, 이는 매우 비효율적이다. 예를 들면, S/W 변경이 여러번 수행되어야 하며, 사용성과 유지 보수에 있어서 에러를 발생시키기 매우 쉽다.

아래 내용은 싱글 프로그램(single program)을 이용하여 다수 축을 제어하기 위한 몇가지 제안하는 솔루션을 설명한다.

루프(loop)을 사용한 프로그램

어플리케이션 프로그램은 명령, 파라미터, 상태, PLCopen 평선의 평선 블록 인스턴스를 포함한 구조로 구성되어 있다.

프로그램에서 축은 동작되고 평선 블록은 루프에서 불러진다.

초기 코드에서 프로그램 이름 식별

Automation Studio 에서 S/W 구성에서 한번 이상 프로그램을 매핑하는 것이 가능하다. 여기서 드라이브 제어를 위한 프로그램은 Cyclic 시스템으로 여러번 링크되어 있다.

Task Name	Period	Storage	Library
Cyclic #4 - [100 ms]			
Basic	1.00.0	UserROM	0 LibACP10MC_SingleAx_LD Basic
Basic1	1.00.0	UserROM	0 LibACP10MC_SingleAx_LD Basic
Basic2	1.00.0	UserROM	0 LibACP10MC_SingleAx_LD Basic

그림 31 소프트웨어 구성에 여러번 맵핑된 프로그램

프로그램은 컴파일되고 제어기에서 여러번 실행된다. 어플리케이션에서 참조 축은 반드시 각 프로그램에 할당되어야 한다. S/W 구성에 할당된 테스크 중 어느 초기화 서브 프로그램에서 참조되는지 판별하는 것이 필요하다. 이것은 "SYS_lib" 라이브러리에 있는 평선을 이용하여 결정될 수 있다. 프로그램 코드와 테스크 이름 사이에 링크로 이루어진다.

선언

```

VAR
  sTaskName : STRING[80];
  status : UINT;
  AxRef : UDINT;
END_VAR

```

프로그램 코드

```

(*Determining the particular task names*)
status := ST_name(0, ADR(sTaskName), 0);

(* Program name Axis1? *)
IF sTaskName = 'Axis1' THEN
  AxRef := ADR(gAxis01);
ELSE
  AxRef := ADR(gAxis02);
END_IF

```

표 6 Structured text 언어에서 "ST_name"으로 특정 테스크 이름 결정하기

로컬 커맨드 변수는 하이 레벨 관리 프로그램에서 개별 축에서 사용하는 변수 할당을 제어해서 연결될 수 있다. 이 할당 변수 윈도우는 CPU의 바로가기 메뉴를 사용하여 로지컬 뷰에서 열 수 있다.

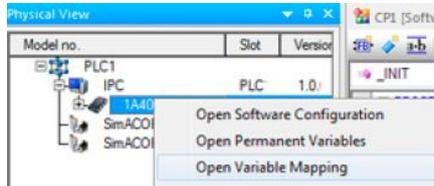


표 7 CPU 바로가기 메뉴를 사용해서 변수 할당 창 열기

게다가, 관리 프로그램이 포인터를 사용하여 단일 축 프로그램의 명령 구조를 투여하게 접근하고 제어할 수 있다.

?

Programming W Libraries W Configuration, system information, runtime control W SYS_lib

W Function blocks W Handling software objects

Programming W Editors W Configuration editors W Assigning variables

전처리기 지시문 사용

이 방법은 참조 축이 싱글 축 프로그램으로 할당되는 것을 결정하는 SYS_lib 라이브러리에 있는 평션 대신 사용되는 전처리기의 다양함과 “초기 코드(Init code)에서 프로그램 이름 식별”와 다르다.

여기서, 컴파일시 프로그램에서 접근 할 수 있는 소프트웨어 구성의 테스트에 대한 컴파일러 옵션에 식별자가 지정된다.



소프트웨어 구성에서, “정의”는 태스크 속성에 지정되어 있다.

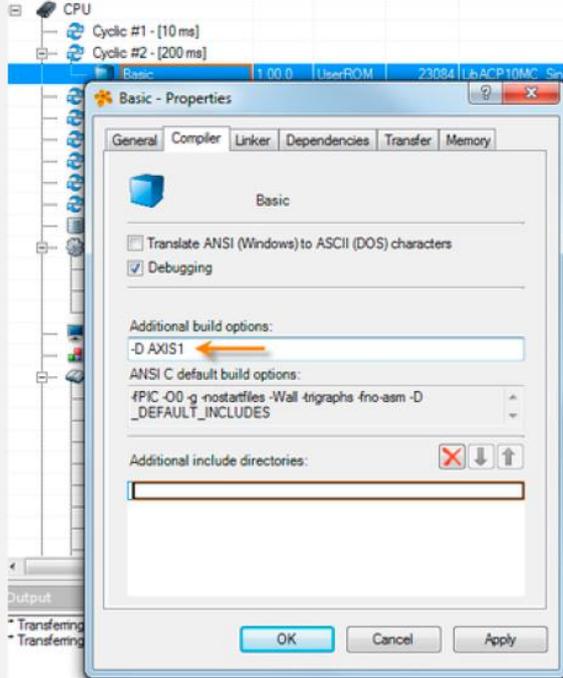


그림 32 소프트웨어 구성의 태스크 속성에 “-D AXIS1” 옵션이 정의되어 있다.

프로그램 코드에서, 어떤 “정의”가 지정되었는지 질의하는 것이 가능하다.

```

프로그램 코드 | #ifdef AXIS1
                |     AxRef := ADR(gAxis01);
                | #else
                |     AxRef := ADR(gAxis02);
                | #endif
    
```

표 8 Init 서브 프로그램에서 „정의“ 질의



Project management W The workspace W General project settings W Settings for IEC compliance

Programming W Programs W Preprocessor for IEC programs

10 요약

강력한 펄션 블록은 B&R 드라이브 솔루션을 제어하기 위해 제공됩니다. 이들은 PLCopen Motion Control 표준을 기반으로 생성되고 기능 사용과 상관 없이 동일한 디자인 특징을 지닌다.

사용자가 이 표준화된 동작에 익숙해지면 모션 제어 라이브러리 (ACP10_MC)에 포함된 펄션을 조합하여 필요한 프로세스를 구현 할 수 있다.

자동화 시퀀스는 스텝 시퀀스를 사용하여 최적으로 실행될 수 있다. 에러 핸들링 루틴은 시퀀스를 완전한 위치결정 어플리케이션으로 바꾼다.

드라이브 제어를 위해 표준에 기능을 더한 특별한 펄션 블록이 제공됩니다. 그들은 B&R 드라이브 솔루션과 관련된 특별한 기능들을 다룬다. 이는 프로그래머에게 어떤 위치결정 테스크도 해결할 수 있는 전반적인 접근 범위를 제공한다.



그림 33 PLCopen Motion control logo