<응용논문> DOI http://dx.doi.org/10.3795/KSME-A.2016.40.11.955 ISSN 1226-4873(Print)
2288-5226(Online)

다중 동작 모드를 가진 PLCopen 표준 호환 모션 응용을 위한 모션 레시피 개념 설계 및 구현§

김상현*· 이경현*· 김태현*†· 최 철**· 강동구**
* 서울시립대학교 기계정보공학과, ** 하이젠모터㈜ 기술연구소

Design and Implementation of Motion Recipe for PLCopen-Compliant Motion Applications with Multiple Operation Modes

Sanghyun Kim*, Kyunghyun Lee*, Taehyoun Kim*†, Cheol Choi** and Donggu Kang**
* Dept. of Mechanical and Information Engineering, Univ. of Seoul,
** R&D Center, Higenmotor Co., Ltd.

(Received May 18, 2016 ; Received September 13, 2016 ; Accepted September 13, 2016)

Key Words: Operation Modes(동작 모드), PLCopen-compliant(PLCopen 표준 호환), Motion Application(모션 응용), Motion Recipe(모션 레시피), Industrial Automation(산업 자동화)

초록: 최근 산업 자동화 분야에서는 확장성을 고려한 표준화된 소프트웨어 기반 모션 제어 시스템 개발 방법과 다품종 소량 생산을 위한 다양한 동작 모드 지원에 대한 요구가 증대되고 있는 추세이다. 소프트웨어 기반 모션 시스템은 단일 장비 상에서 다양한 동작 모드 전환이 용이하게 하지만 다중 동작 모드를 쉽게 정의하는 도구와 동작 모드 변경에 대한 표준화된 동작 절차가 정의되어 있지 않은 상태에서 다중 동작 모드를 지원하는 모션 제어시스템을 구성하는 것은 쉽지 않다. 본 논문에서는 PLCopen 표준 호환 모션 블록들을 이용해 다중 동작 모드를 구성하고 모드 변경을 외부에서 구동하기 위한 프로토콜을 포함하는 모션 레시피 개념을 제안한다. 제안된 모션 레시피 개념은 IEC 61131-3 표준 호환 통합개발 환경인 Beremiz의 기능을 확장하여 구현하였으며, 실제 테스트베드 상에서 그 동작을 검증하였다.

Abstract: In recent years, there have been emerging needs for standardized software-based motion application development for better scalability and support for multiple operation modes for small quantity batch production. Although a software-based motion system provides a basis for constructing multiple operation modes on a machine, it is not easy to construct such systems without tools for defining multiple motion operation modes and standardized mode-change protocols. This paper proposes a motion recipe concept to overcome this problem; the concept includes the authoring of multiple motion operation modes using the PLCopen-compliant motion function blocks and communication protocols to trigger operation mode changes from an external interface. The motion recipe was implemented by extending an IEC 61131-3 compliant IDE called Beremiz, and the correctness of the motion recipe-based application behavior was verified on a real testbed.

1. 서 론

최근 산업 자동화 응용의 핵심 요소인 모션 제어 시스템을 구성할 때 표준화된 프로그래밍 언어와 통신 구조를 사용하여 제품간 호환성을 높여 시스템의 확장성을 높이려는 추세가 두드러지고 있다.(1) 특히 전용 PLC 모션 하드웨어를 대체하여 범용 컴퓨터 상에서 모션 응용 작성을 위한 표준 언어와 표준 모션 블록 라이브러리를 소프트웨어로 구현하는 접근 방법이 널리 쓰이고 있다.(2)

현대 산업 현장에서는 다양한 소비자들의 욕구를 충족시키기 위하여 다품종 소량 생산을 위한 생산, 제조 설비를 구성해야 할 필요성이 있으나

§ 이 논문은 2016년도 대한기계학회 IT융합부문 춘계학술대회(2016. 5. 19-20., 서울시립대학교) 발표논문임.
† Corresponding Author, thkim@uos.ac.kr

소량 생산을 위해 다양한 생산 라인을 구성하는 것은 비용 측면에서 효율적이지 않다. 따라서, 다양한 소프트웨어 모션 동작을 내장한 하나의 생산, 제조 장비를 가지고 다양한 품종의 제품을 가공 또는 조립할 수 있는 시스템 구성 방법이 확산되는 상황이다. 이처럼 요구사항에 따라 다양한 동작 모드가 필요하고 동작 모드 간 전환이 자주 요구되는 모션 제어시스템을 구성하려면 동작 모드를 변경할 때 마다 다른 동작을 수행하는 응용 프로그램을 모션 제어시스템에 다시 전송하거나 외부 제어 명령을 받아 운전 도중 동작 모드를 변경할 수 있는 복잡한 응용 프로그램을 처음부터 다시 설계, 구현해야 한다는 부담이 있다.

이러한 한계를 극복하기 위해 상용 드라이브에서 외부 제어기의 명령을 받아 미리 정의된 특정한 동작 단위를 실행하는 기능을 제공하는 사례가 있다.[3] 그러나, 이 접근 방법에서 제공하는 동작 단위인 Trajectory는 미리 정의된 5가지 간단한 동작 방식 중 하나를 선택하고, 그 동작을 수행하는데 필요한 위치, 속도, 가감속도 등의 파라미터 값을 설정하는 수준의 매우 간단한 동작만을 지원한다. 따라서 하나의 Trajectory로는 실제 생산공정에 사용될 복잡한 동작 모드를 구성하기에는 무리가 있으며, 복잡한 동작을 실행하려면 여러 개의 Trajectory를 사용하여 하나의 동작 모드를 위해 외부에서 지속적으로 실행 명령을 전송하여 Trajectory를 교체해야 한다는 문제가 있다. 또한 표준 프로그래밍 언어를 사용하지 않고 업체에서 제공하는 자체 규격을 사용하여 응용 프로그램을 작성해야 하기 때문에 다양한 장치로의 확장이 힘들다는 문제가 남아 있다.

본 연구에서는 이러한 문제를 해결하기 위해 모션 응용을 개발하기 위한 통합개발환경인 Beremiz[4]의 기능을 확장한 동작 모드 편집기능과 외부 제어기와의 통신 프로토콜까지 결합한 모션 레시피(Motion Recipe) 개념을 제안한다. 모션 레시피는 응용 개발에서 IEC 61131-3 표준 프로그래밍 언어를 사용하도록 하며, 다중 동작 모드를 지원하는 모션 제어시스템의 기본 동작 단위이다. 모션 레시피를 사용하면 운전 도중에 외부 제어기의 명령 또는 운전자의 외부 입력에 따라 실행 중이던 모션 레시피를 중지하고 다른 동작을 수행하는 모션 레시피를 실행하는 형태의 모션 제어시스템을 매우 쉽게 구성할 수 있다. 결과적으로 본 논문에서 제안한 모션 레시피 개념을 적용하면 다품종 소량 생산이 필요한 산업 현장에서 다중 동작 모드를 가지는 모션 제어시스템을 쉽게 구성할 수 있다. 본 논문의 구성은 다음과 같다. 2장에서는 연구 배경에 대해 소개하고, 3장에서 모션 레시피 지원 기능 구현의 자세한 내용을 다룬다. 4장에서는 모션 레시피를 이용한 응용 작성방식에 대하여 기술하고, 5장에서 실제 모션 제어시스템에 적용한 결과를 제시한 후 6장에서 결론을 맺는다.

2. 연구 배경

2.1 Beremiz 통합개발환경

Beremiz는 IEC 61131-3 표준 호환 모션 응용 통합개발환경으로, 오픈 소스 소프트웨어의 형태를 가지고 있으며 플러그인(Plug-in) 구조를 지원해서 필요에 따라 다양한 통신 프로토콜, HMI, 외부 모듈과의 연동 등 기능 확장이 용이하다.[5] 본 연구에서 제안한 모션 레시피는 Beremiz의 기능을 확장하여 외부 제어 명령 프로토콜을 지원하는 응용 프로그램 자동 생성 플러그인과 모션 레시피 지원 응용 개발을 위한 모션 레시피 라이브러리 및 모션 레시피 편집기를 구현하였다.

2.2 소프트웨어 기반 모션제어 응용 개발 표준

모션 장비 또는 응용의 호환성 보장을 위한 소프트웨어 개발 관련 대표적인 표준으로는 IEC 61131-3[6]과 PLCopen[7]이 있다. IEC 61131-3 표준은 PLC 응용 개발용 프로그래밍 언어와 개발 방법론을 정의하고 있다. IEC 61131-3 표준에 정의된 언어로는 IL(Instruction List), ST(Structured Text), LD (Ladder Diagram), FBD(Function Block Diagram), SFC (Sequential Function Chart) 등 5가지가 있다.

표준에 근거하지 않는 기존의 모션 응용 개발 방식은 응용 요구사항이나 운용 플랫폼이 바뀔 때 마다 기존의 개별 모션 동작을 매번 새로 작성해야 하는 번거로움을 피할 수 없었다. PLCopen TC2 표준은 이러한 제약사항을 극복하기 위해 플랫폼에 상관 없이 재사용 가능한 표준 모션 함수 블록(Motion Function Block)들로 구성된 모션 제어 라이브러리를 정의하고 있다. 또한 정의된 모션 함수 블록들은 IEC 61131-3 표준과 호환되므로 PLC 응용에 쉽게 통합할 수 있다는 장점이 있다.

2.3 다중 동작 모드를 가진 응용 개발

표준을 준수하여 응용을 개발하면 유지, 보수가 편리하고 추후 확장이 쉽다는 장점이 있다. 하지만 다중 동작 모드를 지원하는 응용 프로그램 작성을 위한 표준은 별도로 없으므로 응용 프로그램

을 작성하려면 하드웨어 수준에서 외부 명령과의 인터페이스를 구성해야 한다. 따라서 IL, ST와 같은 텍스트 기반의 언어 사용 비중이 매우 높아지고, 장치 별 통신 방식 분석과 응용의 명령 처리 과정 등을 처음부터 설계해야 하는 부담이 있다.

본 연구에서는 이러한 문제점을 해결하기 위해 다중 동작 모드를 지원하는 모션 응용 프로그램의 자동 생성이 가능하도록 외부 제어 명령 프로토콜을 정의하고 외부 제어 명령과 응용간 인터페이스를 추상화하는 모션 레시피 라이브러리를 구현하였다. 개발자는 하나의 동작 모드를 하나의 모션 레시피에 대응하여 모션 응용 프로그램을 개발하여 외부 제어기를 통해 동작 모드를 전환할 수 있다. 또한, 외부 제어기와 모션 레시피가 내장된 모션 구동장치간 통신 프로토콜이 단순하여 소프트웨어적으로 프로토콜 구현만 하면 티칭 펜던트(Teaching Pendant), 태블릿(Tablet), 범용 PC 등 다양한 형태의 외부 제어기와 연동할 수 있다는 장점이 있다.

3. 모션 레시피 지원 기능 설계 및 구현

모션 레시피는 다중 동작 모드를 갖는 모션 제어 시스템의 동작 단위이다. 각 모션 레시피는 고유한 번호로 구분되며 외부 제어기를 통해 제어 명령을 모션 응용 프로그램으로 전송하여 모션 레시피를 실행할 수 있고 운전 도중 실행중인 모션 레시피를 중지하고 다른 레시피를 실행하여 모션 제어시스템의 동작 모드를 변경할 수 있다.

Fig. 1은 두 개의 동작 모드를 가지는 모션 제어 시스템을 외부에서 본 관점과 개발자가 본 내부 관점을 나타낸 것이다. 각 동작 모드는 하나의 모션 레시피에 대응되고, 그 동작의 내용은 모션 블록에 사용자가 구현하며 모션 레시피 라이브러리를 통해 사용자가 외부 제어기를 통해 전송한 명령을 받아 동작 모드를 변경할 수 있다.

3.1 다중 동작 모드를 지원하는 시스템 구조

다중 동작 모드를 지원하는 시스템의 구조는 Fig. 2와 같이 모션 제어기(Motion Controller)와 모션 제어기에 명령을 내리고 응용의 동작 상태를 모니터링할 수 있는 외부 제어기(External Controller)로 구성된다.

모션 제어기에서는 모션 제어시스템을 위한 응용 프로그램이 실행되고 있으며, 외부 제어기로부터 제어 명령을 받아 실행할 레시피의 정보를 읽어 모션 제어시스템의 동작 모드를 변경하고 현재 동작 상태를 외부 제어기에 전송하여 사용자가 확인할 수 있다. 모션 제어기의 내부 구성 요소는 다음과 같다.

모션 제어기의 상태 머신(State Machine)은 외부 제어기와의 통신을 담당하는 프론트엔드(Front-end)로서, 현재 동작 모드의 상태를 파악하여 외부 제어기에 전송하거나 올바르지 않은 입력이나 내부 에러 발생에 대한 예외처리의 역할을 수행하고 외부 제어 명령을 모션응용 내부의 변수로 전달하는 역할을 수행한다. 모션 레시피 라이브러리(Motion Recipe Library)는 외부 제어 명령을 받아 상태 머신과 연계하여 응용 내부의 모션 레시피 실행 과정을 담당한다. 개발자가 외부 제어 명령을 받는 과정을 구현할 필요가 없도록 지원하여 모션 응용 프로그램을 개발하기 용이하도록 한다.

3.2 외부 제어기와의 연동을 위한 통신 프로토콜

외부 통신 프로토콜은 모션 제어기의 동작을 변경하고 모션 제어기의 상태를 모니터링 할 수 있도록 Table 1과 같이 정의하였다. 외부 제어기로부

Table 1 External communication protocol

데이터		타입	설명
모션 명령	RRN (Request Recipe Number)	8bit 정수	실행하고자 하는 레시피의 번호
	Execute	1bit	레시피 실행 신호
모션 상태	ARN (Active Recipe Number)	8bit 정수	현재 실행되고 있는 레시피의 번호
	Status	3bit	모션 제어기의 상태

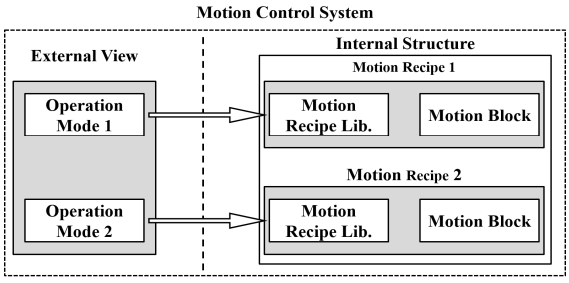


Fig. 1 Motion control system

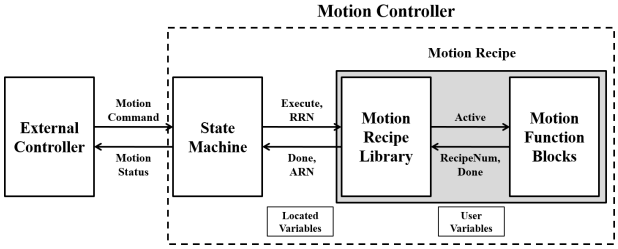


Fig. 2 Motion control system architecture

Table 2 Motion status value

상태 이름	값	설명
Disabled	0x0	레시피 비활성화
Completed	0x1	레시피 실행 완료
Running	0x2	레시피 실행 중
Error	0x4	에러 상황

터 모션 제어기가 받는 데이터는 모션 명령(Motion Command) 으로 동작 모드를 변경하기 위한 정보를 담고 있다. 모션 제어기가 외부 제어기로 전송하는 데이터는 모션 상태(Motion Status)로 현재 모션 제어기의 동작 상태 정보를 담고 있으며, 지원되는 상태 정보의 종류는 Table 2와 같다.

각 레시피는 레시피 번호(Recipe Number)로 구분된다. 새로운 레시피를 실행하고자 할 때 실행하고자 하는 레시피의 번호를 RRN에 저장하고 Execute 신호를 True로 하여 모션 제어기에 전송한다. Execute 신호는 상승 엣지로 동작하며 모션 제어기에서는 Execute 신호의 상승 엣지를 감지할 때마다 RRN을 읽어서 해당 레시피를 실행한다. 따라서 Execute 신호가 False라도 현재 레시피는 중단하지 않고 계속 실행되며 현재 레시피를 다시 실행하고 싶을 때에는 RRN을 유지한 채 Execute 신호에 다시 상승 엣지를 생성하면 된다.

3.3 통합 개발환경을 이용한 모션 레시피 저작 기능 구현

외부 통신 프로토콜의 데이터는 가장 먼저 모션 제어기의 상태 머신으로 전달된다. 상태 머신의 역할은 두 가지이다.

첫 번째는 외부 명령 프로토콜 데이터를 응용 프로그램 개발에 사용할 수 있도록 통신 데이터나 외부 메모리에 직접 매핑해 응용 프로그램에서 사용할 수 있는 변수인 위치 변수(Located Variables)로 만드는 것이다. 즉 상태 머신은 외부 통신 데이터와 내부 위치 변수간의 변환을 담당한다.

두 번째는 현재 모션 제어기의 상태를 판단하여 Table 2의 상태 정보를 모션 상태의 Status필드에 저장하여 외부 제어기로 전송하는 것이다. 기본적으로 상태 머신은 외부 명령 프로토콜의 모든 데이터를 위치 변수로 변환하지만 Status의 상태 정보는 상태 머신에서 직접 판단하므로 이것은 위치 변수로 변환하는 대신에 모션 레시피의 완료 신호를 받아서 Completed 상태 판단을 위해 Done이라는 위치 변수를 둔다. 따라서 위치 변수는 RRN, Execute, ARN, Done의 네 개가 된다.

Table 3 Motion recipe library

라이브러리	설명
RecipeReader	레시피의 실행 신호를 읽어서 해당하는 레시피에 Active 를 출력
DoneReceiver	실행중인 레시피의 Done 을 업데이트

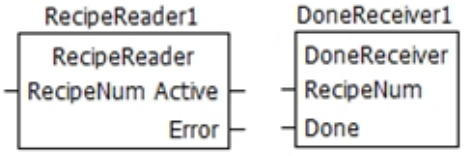


Fig. 3 Motion recipe library prototypes

위치 변수는 응용 프로그램 개발에서 직접 사용할 수 있으므로 상태 머신만 있어도 모션 레시피 응용 프로그램을 개발할 수 있다. 하지만 위치 변수는 외부 명령 프로토콜과 밀접한 관련이 있기 때문에 프로토콜 규칙을 지켜야 한다. 예를 들어 Execute 신호의 상승 엣지가 관찰되면 RRN을 확인하여 해당 번호를 가진 레시피가 있으면 ARN을 그 번호로 업데이트하고, 없으면 에러의 의미로 ARN을 0으로 업데이트 해야 한다. 하지만 응용 프로그램을 작성할 때 이러한 규칙을 준수하게 하려면 개발자가 규칙을 미리 알고 있어야 하고 코드 레벨에서 동일한 규칙의 반복 적용에 의한 비효율성과 응용 프로그램에 오류가 생길 여지가 있는 등 유지 보수 측면에서 어려움이 있다.

모션 레시피 라이브러리는 이러한 규칙들을 지키도록 미리 구현해 놓은 것으로 응용 프로그램 개발자가 더 쉽게 모션 레시피 응용 프로그램을 개발할 수 있도록 한다. 모션 레시피 라이브러리는 Table 3에 나타낸 함수 블록(Function Block)으로 구현되었으며 RecipeReader와 DoneReceiver로 구성되며 모션 함수 블록의 프로토타입은 Fig. 3과 같다. 두 함수 블록은 각각의 레시피마다 하나씩 필요하다. 따라서 두 함수 블록은 한 번의 제어 주기마다 레시피의 수만큼 실행된다. RecipeReader의 역할은 ARN을 업데이트하고 RRN에 해당하는 레시피를 찾아 Execute 신호를 전달하는 것이다. DoneReceiver의 역할은 레시피의 완료 신호를 받아서 그 중 ARN에 해당하는 레시피의 완료 신호만을 선택한 후 위치변수 Done에 그 값을 전달하는 것이다.

4. 모션 레시피를 사용한 응용 개발

본 연구에서는 모션 레시피를 사용하는 응용 프

Table 4 Built-in motions in motion recipe editor

동작	기능
Power On	해당 축에 전원 인가
Power Off	해당 축에 전원 해제
Move Absolute	지정된 절대위치까지 이동
Move Relative	지정된 상대위치까지 이동
Move Velocity	지정된 속도로 이동
Homing	Home 위치로 이동
Stop	비상 정지
Halt	정상 정지
Reset	해당 축 초기화

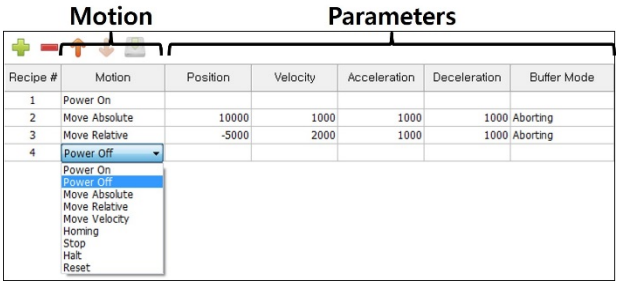


Fig. 4 GUI of motion recipe editor

로그램 개발을 쉽게 할 수 있도록 통합 개발환경인 Beremiz의 기능을 확장, 구현하였다. Beremiz는 모션 레시피를 적용한 응용 프로그램의 두 가지 개발 방법을 지원한다. 첫 번째는 모션 레시피 편집기를 사용하여 관련 표준에 익숙하지 않은 개발자라도 비교적 간단한 응용 프로그램을 자동 생성할 수 있는 방법이고, 두 번째는 복잡한 응용 프로그램을 구현하고자 하는 개발자가 모션 레시피 라이브러리를 활용하여 직접 IEC 61131-3과 PLCopen 표준을 준수하는 응용 프로그램을 작성하는 방법이다.

4.1 모션 레시피 편집기를 통한 응용 개발

비교적 간단한 응용 프로그램을 개발하려면 Fig. 4와 같이 구현한 모션 레시피 편집기를 사용하여 관련 표준에 익숙하지 않은 개발자도 쉽고 빠르게 다중 동작 모드를 지원하는 모션 응용을 개발할 수 있다. 개발자는 모션 레시피 편집기를 사용하여 프로그램을 자동으로 생성할 수 있다. 모션 레시피 편집기에서 각 행은 하나의 레시피를 의미하는데, 각 레시피에 대한 추가/삭제 및 위치 이동 그리고 현재 구성한 시스템을 저장하는 기능을 제공한다. 레시피 번호는 각 행의 첫 번째 열에서 확인할 수 있으며 레시피를 정의하기 위해 동작

```
/* Execute recipe number 2 part */
/* Setup parameters */
__SET_VAR(rf_2., AXIS, *__IW0_0);
__SET_VAR(rf_2., RECIPE_NUM, 2);
__SET_VAR(rf_2., POSITION, 10000.0);
__SET_VAR(rf_2., VELOCITY, 1000.0);
__SET_VAR(rf_2., ACCELERATION, 1000.0);
__SET_VAR(rf_2., DECELERATION, 1000.0);
  SET VAR(rf 2., BUFFER MODE,
      MC_BUFFER_MODE__MCABORTING);

/* Call built-in motion function */
RF_MOVE_ABSOLUTE_body__(&rf_2);
```

Fig. 5 Auto-generated C code of motion recipe

(Motion)과 각 동작의 파라미터를 설정할 수 있다. 동작은 레시피가 어떤 종류의 동작을 할지 결정하는 것으로 모션 레시피 편집기에서 지원하는 내장(built-in) 동작 중에서 선택할 수 있다.

모션 레시피 편집기에 내장된 동작은 Table 4와 같이 비교적 단순한 모션 동작을 구성하는 데 필요한 PLCopen 표준 호환 모션 함수로 구성하였다. 각 동작별로 사용할 수 있는 파라미터의 종류는 목표 위치, 속도 등 동작 구현 시 사용된 함수에 따라 달라진다. 한편 각 동작은 그 동작을 구성하는 함수의 변수들 중에서 레시피의 완료를 의미하는 신호를 선정하여 동작 완료 시 모션 상태를 통해 알 수 있다.

본 연구에서는 통합 개발환경 Beremiz에서 응용을 개발하는 것과 별도로 모션 레시피를 이용하여 작성한 모션 제어 응용은 자동으로 C언어로 작성된 코드를 Fig. 5와 같이 생성하도록 구현하였다. 생성된 C 코드는 모션 레시피 편집기를 통해 정의한 내용에 따라 파라미터들을 초기화하고 각 동작 별 내장된 함수를 호출하는 방식으로 구성되어 있다. 이를 위해 각 내장 동작 별 C 언어로 작성된 함수를 구현하였고 이 함수의 구성은 모션 레시피 라이브러리를 사용한 것과 같다. C 코드 자동 생성 이후 컴파일을 진행하여 실제 사용될 응용 프로그램을 생성하고 모션 제어기에 전송하여 바로 구동이 가능하다. 그러나, 내장된 동작만을 이용해 모션 제어 응용을 구성해야 하므로 복잡한 동작이 요구되는 모션 시스템을 구성하기에는 한계가 있다.

4.2 모션 레시피 저작 기능을 통한 개발 방법

개발자는 IEC 61131-3 및 PLCopen을 준수하는 프로그래밍을 통해 직접 모션 레시피의 동작을 구현하여 다중 동작 모드를 지원하는 모션 응용 프

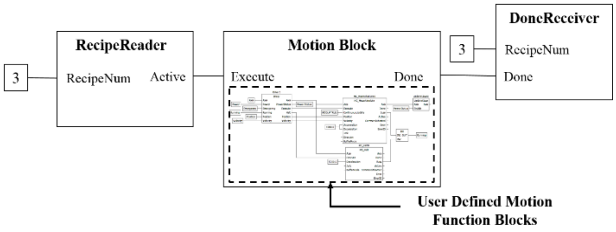


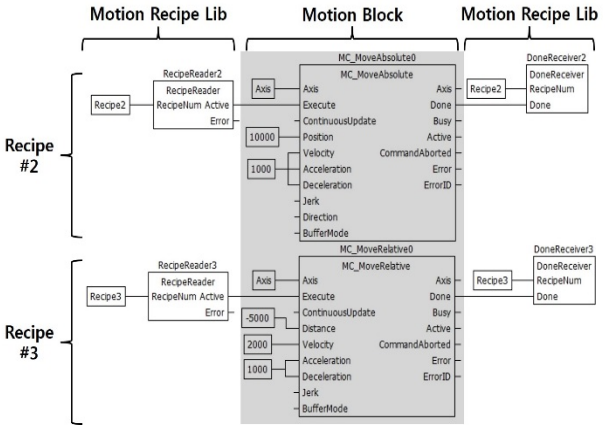
Fig. 6 Basic form of the recipe

로그램을 자유롭게 개발할 수 있다. 개발자가 직접 구현한 모션 레시피의 기본 구조는 Fig. 6과 같이 동작 모드를 구별하기 위한 모션 레시피 번호, 모션 명령을 받는 RecipeReader, 모션 레시피의 동작 완료 상태를 알리기 위한 DoneReceiver, 개발자가 직접 구현한 모션 블록으로 구성된다.

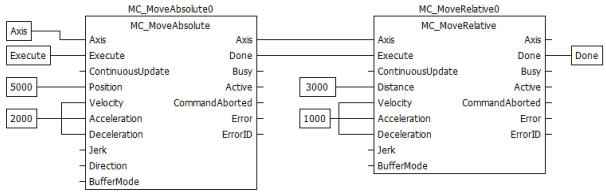
모션 제어시스템의 동작 모드에 대응하는 모션 레시피 하나를 구현하려면 먼저 각 레시피의 번호를 할당해야 한다. 레시피 번호는 전체 응용 프로그램에서 유일한 양의 정수로 정의해야 한다. 다음으로 모션 블록 영역에 아래와 같은 방식으로 3장에서 설명한 레시피 실행 규칙을 만족하는 응용 프로그램을 작성한다.

외부 제어기의 모션 명령과 연동하기 위해 개발자는 RecipeReader와 DoneReceiver 한 쌍으로 구성된 모션 레시피 라이브러리를 모션 블록에 연결하여 모션 레시피를 구성해야 한다. 모션 명령을 받으면 모션 레시피 라이브러리의 RecipeReader가 RecipeNum 입력과 모션 명령의 레시피 번호와 비교하여 일치할 경우 Active 신호를 출력한다. Active 신호는 모션 블록의 Execute 입력으로 전달되어 동작을 수행하고 사용자가 정의한 동작 완료 조건 도달 시 모션 블록의 Done 신호를 출력한다. Done 신호는 모션 레시피 라이브러리의 Done Receiver의 입력으로 전달되어 모션 레시피의 실행 완료 상태를 모션 상태에 업데이트한다. 만약 모션 레시피의 동작 특성상 실행 완료 상태가 존재하지 않는 경우가 있다. 이 때 DoneReceiver는 생략 가능하다.

Fig. 7(a)는 모션 레시피를 직접 작성하여 구현한 모션 응용 프로그램이다. 두 개의 모션 레시피를 이용하여 구현하였고 각 동작의 내용을 담은 모션 블록은 회색 음영으로 표시하였다. Fig. 7(b)는 Fig. 7(a)의 모션 블록 내부 구현을 나타낸다. 모션 블록을 따로 함수 블록으로 구현하여 응용 프로그램의 가독성을 높일 수 있다. 모션 블록의 내용은 모션 레시피 작성 규칙을 만족한다면 규모에 관계없이 작성할 수 있다. 따라서 모션 레시피 편집기



(a) Using user-defined motion recipes



(b) Internal structure of user-defined motion recipe

Fig. 7 Motion recipes using IEC 61131-3 motion FBD

Fig. 8 GUI of external control/monitoring module

와 비교하여 복잡한 응용을 개발 요구사항대로 제한 없이 구현할 수 있다는 장점이 있다.

5. 모션 레시피 동작 검증

4장에서 다중 동작 모드를 지원하는 모션 제어시스템을 구현하기 위하여 응용 프로그램을 개발하는 방법 두 가지를 제시하였다. 본 논문에서는 각각의 방법으로 구현한 응용의 동작을 검증하기 위하여 실제 모터 드라이브와 모터를 이용한 테스트베드 상에서 동작 결과를 확인하였다.

모션 레시피를 지원하는 시스템은 외부 제어기의 모션 명령을 받아 운전 도중 동작 모드를 변경할 수 있고 모션 응용으로부터 모션 상태를 전송받아 동작 상태를 모니터링할 수 있다. 본 논문에서는 개발용 호스트 PC에서 동작하는 Beremiz 통합개발환경의 플러그인 형태로 Fig. 8과 같이 외부 제어기의 기능을 구현하였다. 구현된 외부 제어 모듈의 동작은 다음과 같다. 먼저 사용자가 실

행하고자 하는 레시피 번호를 RRN에 입력하고 Execute 버튼을 클릭하면, 외부 제어 모듈에서 응용 프로그램으로 모션 명령을 전송한다. 또한 모션 제어시스템에서 외부제어기로 전송된 모션 상태를 읽어 ARN과 Motion Status에 출력한다.

본 연구에서는 모션 레시피를 이용한 다중 동작 모드 지원 모션 응용 프로그램의 동작을 확인하기 위하여 Fig. 9와 같이 ARM 기반 임베디드 모션 제어기와 모터 드라이브, 모터로 이루어진 테스트 베드를 구성하였다. 앞에서 언급한 바와 같이 외부 제어기 역할은 통합개발환경인 Beremiz에 구현한 플러그인 모듈로 대체하였다.

모션 레시피를 이용한 다중 동작 모드의 검증을 위한 응용 프로그램은 Fig. 10(a)와 같이 “Move Absolute - Move Relative - Move Velocity - Halt”의 순서로 레시피를 순차적으로 실행하도록 구현하였다. 장치의 동작 안정성을 위해서 실제 레시피 구성에는 PowerOn - PowerOff 시퀀스도 필요하지만, 지면관계상 생략하였다. 동작의 정확성 검증은 2번 레시피부터 시작하여 각 레시피를 순차적으로 실행하면서 Beremiz가 지원하는 실시간 디버깅 인터페이스를 통해 각 레시피 수행에 따른 모터의 위치 값과 속도 값, 그 외 제어변수 값들을 확인하는 방법으로 진행하였다. 확인 결과, 모션 레시피 편집기를 활용한 구현과 모션 라이브러리를 이용한 사용자 정의 레시피를 이용한 구현 모두 Fig. 10(b)와 같이 정확하게 동작함을 확인하였다.

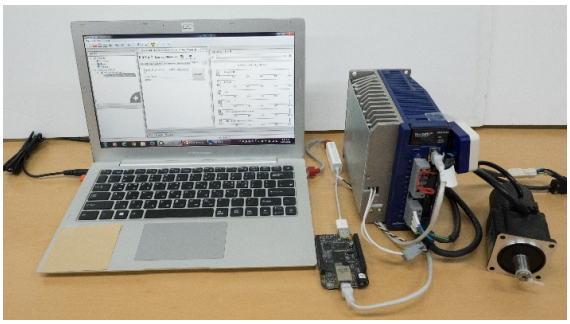
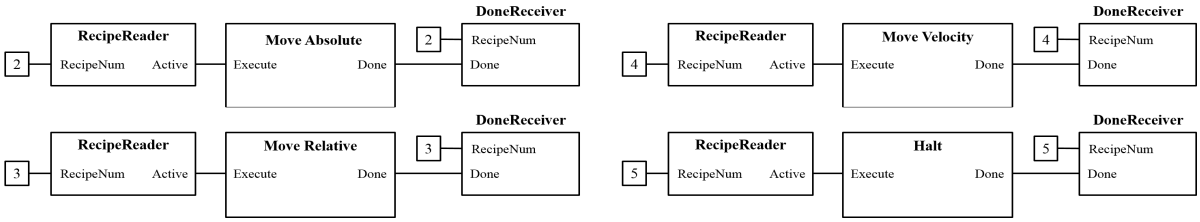


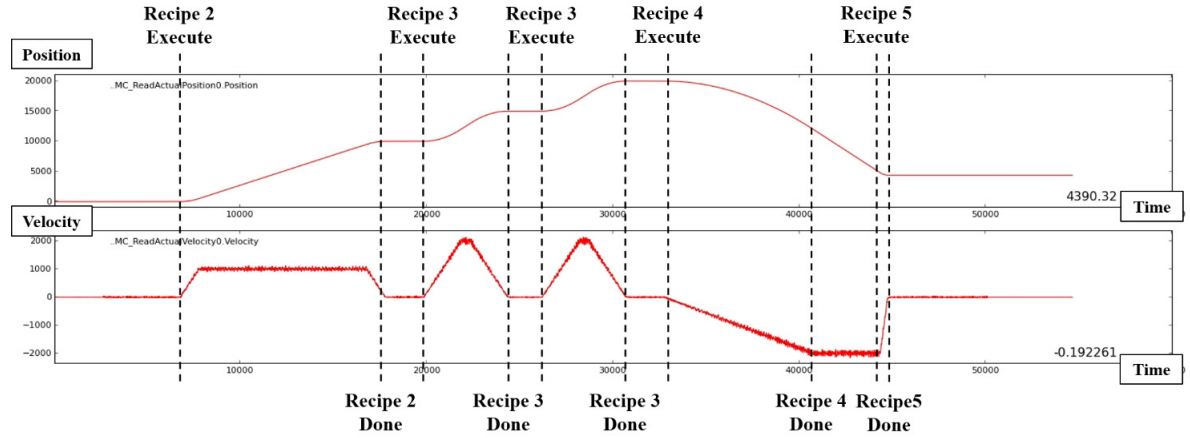
Fig. 9 Testbed for motion recipe evaluation

6. 결 론

본 연구에서는 다중 동작 모드 지원을 위한 모션 레시피 개념을 제시하고, 기존의 Beremiz 통합개발환경의 기능을 확장하여 간단한 모션 응용을 구현할 수 있는 모션 레시피 편집기와 복잡한 동작 모드의 구현이 요구될 때 모션 레시피를 직접 작성할 수 있도록 모션 레시피 라이브러리를 구현하여 외부 제어 명령에 따라 동작 모드를 변경하



(a) Motion recipes for test application



(b) Position & velocity graphs

Fig. 10 Test application and experimental result

는 응용을 쉽게 작성할 수 있도록 하였다. 또한 실제 모션 제어시스템을 구성하여 올바르게 동작함을 확인하였다.

후 기

본 논문은 2015 년도 산업통상자원부 및 한국산업기술평가관리원(KEIT)의 지원을 받아 로봇산업융합핵심기술개발사업의 일환으로 수행한 연구임. (No. 10060112, 과제명: 로봇전용 All-in-One 중공형 엑츄에이터 시리즈 개발)

참고문헌
(References)

(1) Sung, M. and Choi, C., 2013, “Employing Open-source for IT Convergence Industrial Automation,” *Journal of the KSME,* 53(11), pp. 59~62.

(2) Lee, J., Kim, C., Kim, I., Kim, Y. and Kim, T., 2014, “Implementation and Validation of EtherCAT Support in Integrated Development Environment for Synchronized Motion Control Application,” *Trans. Korean Soc. Mech. Eng. A*, Vol. 38, No. 2, pp. 211~218.

(3) SERAD, IMD Series, http://www.serad.fr/Brushless-drives-IMD-Series.htm.

(4) Kim, I., Kim, T., Sung, M., Tisserant, E., Bessard, L. and Choi, C., 2013, “An Open-source Development Environment for Industrial Automation with EtherCAT and PLCopen Motion Control,” *Proc. of the 18th IEEE ETFA*, pp. 1~4.

(5) Kim, I. and Kim, T., 2012, “Employing Open-source Software for Development of Open Industrial Automation Systems,” *Proc. of the KSME IT Convergence Division Spring Conference,* pp. 139~140.

(6) IEC, 2013, “IEC 61131-3, 3rd Ed. Programmable Controllers-Programming Languages,” International Electrotechnical Commission.

(7) PLCopen, http://www.plcopen.org.