한국시뮬레이션학회 논문지
Vol. 18, No. 4, pp. 67-79 (2009. 12)

PLC 기반 제어정보 모델링 방법론

고민석[1] · 곽종근[1] · 왕지남[2] · 박상철[2†]

Control Level Process Modeling Methodology Based on PLC

Minsuk Ko · Jonggeun Kwak · Ginam Wang · Sangchul Park

ABSTRACT

Because a product in the car industry has a short life cycle in recent years, the process planning and the manufacturing lines have to be changed frequently. Most of time, repositioning an existing facility and modifying used control information are faster than making completely new process planning. However, control information and control code such as PLC code are difficult to understand. Hence, industries prefer writing a new control code instead of using the existing complex one. It shows the lack of information reusability in the existing process planning. As a result, to reduce this redundancy and lack of reusability, we propose a SOS-Net modeling method. SOS-Net is a standard methodology used to describe control information. It is based on the Device Structure which consists of sensor information derived from device hardware information. Thus, SOS-Net can describe a real control state for automated manufacturing systems. The SOS-Net model is easy to understand and can be converted into PLC Code easily. It also enables to modify control information, thus increases the reusability of the new process planning. Proposed model in this paper plays an intermediary role between the process planning and PLC code generation. It can reduce the process planning and implementation time as well as cost.

Key words : FB (Function Block), LD (Ladder Diagram), PLC (Programmable Logical Control), SOS-Net, State

요 약

자동차 제조 기업은 생산되는 제품 수명 주기가 짧기 때문에, 공정계획 및 라인 변경이 빈번히 일어난다. 따라서 새로운 공정을 계획하는 경우보다 기존 공정계획을 바탕으로 라인의 설비를 재배치하거나, 제어정보를 수정하는 경우가 많다. 하지만 생산라인의 제어 정보를 기술하는 표준 방법론이 없기 때문에 기존 공정의 정보를 해석하고 수정하는데 많은 노력이 요구된다. 따라서 본 논문에서는 자동화 생산라인에서 일반적으로 사용할 수 있는 제어레벨 공정 모델링 방법론(SOS-Net)을 제안하고자 한다. 제안된 방법론은 실제 현장라인과 동일한 Low Level의 정보들을 체계적으로 표현함으로써 모델링 결과가 현업에 직접 사용될 수 있도록 고려하였다. 본 논문에서 제안하는 SOS-Net은 쉽게 작성할 수 있으며, 기존의 High Level 모델링 방법들이 갖는 한계점을 극복하고, 현업에서 사용할 수 있는 FB(Function Block) 제어 코드를 생성하는 것을 목적으로 한다.

주요어 : 제어코드, 시뮬레이션, 공장 모델, 이산사건, 상태 다이어그램

* 본 연구의 일부는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.(UD080042AD)
2009년 8월 5일 접수, 2009년 9월 21일 채택
[1] 아주대학교 산업공학과 대학원
[2] 아주대학교 산업공학과
주 저 자 : 고민석
교신저자 : 박상철
E-mail; sebaminsuk11@ajou.ac.kr

1. 서 론

최근 소비자의 기호가 다양해짐에 따라 가전, 자동차, 의류 등 사회 전반에 걸쳐, 제품의 수명주기(Product Life Cycle)가 짧아지고 있다. 이 같이 급변하는 사회적 추세에 발맞추기 위하여 기업은 신제품의 개발 및 생산시간 단축을 위해 많은 노력을 기울이고 있다. 최근까지 기업은 제품개발에 있어서의 각 단계별 비용절감 및 개발기간 단축에 초점을 두고 있다[1]. 특히 자동차 산업의 경우 시설 및 시장 규모에 비해 프로세스가 자주 변경되는 제조산업이다. 자동차 제조 기업은 글로벌 마켓을 겨냥하는 대기업이 많고, 생산되는 제품 수명 주기가 짧기 때문에, 공정계획 및 라인 변경 횟수가 다른 산업에 비해 크다. 따라서 새로운 공정을 계획하는 경우보다 기존 공정계획을 바탕으로 라인의 설비를 재배치하거나, 제어정보를 수정하는 경우가 많다. 하지만 생산라인의 제어 정보를 기술하는 표준 방법론의 부재로 인해 과거 공정에 대한 체계적 정보가 존재하지 않는 것이 일반적이다. 이로 인해 기존 공정의 정보를 해석하고 수정하는데 많은 노력이 요구되는 실정이다. 실제로 공정 계획 및 변경에 따른 설비 및 제어 코드 수정은 직접 관련된 작업자가 동반되지 않는 한 기존 정보를 재사용, 수정하는 것이 매우 어렵다.

자동차와 같은 제조 분야에서는 시장요구사항을 정확히 수렴하여 이를 제품 모델의 변경에 신속히 반영하는 것이 기업 경쟁력 결정에 매우 중요한 역할을 한다. 이러한 시장 상황에 민감하게 대응하기 위해서는 신속한 생산라인 수정 및 안정화 기술이 필요하다. 이 같은 생산라인의 수정 기간은 Fig. 1과 같은 두 가지 요소로 구성된다. 첫째는 라인 정지 기간(Down Time)이고, 다른 하나는 라인 안정화 기간(Ramp-Up Time)이다.

먼저 라인 정지기간이란, 제품 생산에 들어가기에 앞서, 생산에 필요한 설비를 설치하는데 필요한 시간이다. 따라서 이 기간에는 모든 설비들이 정지된 상태에서 계획된 프로세스에 맞춰 잘 작동 할 수 있도록 조율한다. 그리고 라인 안정화기간은 설비 설치가 종료된 후에, 전체 라인을 안정화 시키는 기간으로, 이 기간에는 제어프로그램을 수정하고, 설치된 설비를 시운전함으로써 프로그램 코드를 검증한다. 공정 수정 빈도가 잦아지는 현대 제조시스템에서 이 두 기간 은 기업의 이익과 직접 연결되는 중요한 시간이다. 왜냐하면, 제품에 대한 시장의 요구가 최고조에 이르렀을 때, 시장에 제품을 출시하지 못하면, 판매 수익과 함께 기업 인지도가 함께 추락하기 때문이다. 이를 해결하기 위해서는 기존 공정 정보를 체계화하여 새로운 공정이 기존 생산 라인에 적응하기 위한 시간을 최소로 줄여야 한다. 이 같은 체계화를 위해서는 생산 공정에 투여되는 리소스를 명확히 정의하고, 라인 구동을 위한 제어프로그램을 논리적으로 정리하여, 프로세스 변경 및 설비 변경 시 소요되는 시간을 최소화해야 한다.

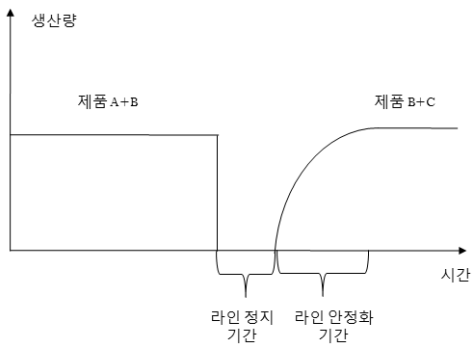

그림 1. 라인 정지 기간과 라인 안정화 기간

체계적 프로세스 모델링을 위해서 Petri-Net과 FSA (Finite State Automata)를 통한 공정 모델링 방법론이 꾸준히 연구되어왔다. 이 같은 프로세스 모델링 연구들의 주된 목표는 공정 정보를 추상화 시켜 목적 모델을 만들고, 이를 분석, 시뮬레이션 하여 문제를 해결하는 것이었다. 특히 공정 제어 분야의 모델링 연구의 주된 목적은 추상화 시킨 모델링 정보를 기반으로 사실적 제어코드를 생성하고자 하는 것이었다. 특히, Petri-Net모델을 제어 코드 정보로 변환하려는 노력이 많이 시도되었지만, 현업에 미치는 효과는 미비하였다. Park은 Petri-Net 을 통해[2], Remadege과 Wonham[3]은 SCT 모델링 결과를 PLC(Programmable Logical Controller) 제어코드로 변환하는 연구를 수행하였고, Lauzon 등[4]은 Automata를 통해 FB(Function Block)코드를 자동생성 하려고 하였다. 하지만 이러한 연구를 통해 얻어진 제어정보는 실용성 측면에서 한계를 갖는다[5]. 이러한 활용상의 한계의 주된 원인은 Abstraction Level의 차이에서 찾을 수 있다. 즉 Petri-Net과 FSA는 일반적으로 High Level에서 제어 정보를 모델링 하고 있지만, 실제 현업의 제조는 Low Level의 센서, 공압, 전기 회로 등의 제어로 구성이 되기 때문이다[5,6,8].

본 논문에서는 이 같은 제어 정보 재사용에 관한 문제를 해결하고자 생산라인 제어레벨 공정 모델링 방법론(SOS-Net)을 제안하고자 한다. 제안된 방법론은 실제 현장라인과동일한 Low Level의 정보들을 체계적으로 표현함으로써 모델링과 현업 활용의 한계를 극복하였다. 본 논문에서 제안하는 SOS-Net은 쉽게 작성할 수 있으며,

기존의 High Level 모델링 방법들이 갖는 한계점을 극복하고, 현업에서 사용할 수 있는 제어 코드를 생성하는 것을 목적으로 한다.

본 논문에서 제안하는 방법론은 고민석 외 3인이 제안한 모델링 방법론인 SOS-Net을[12] 실용성 측면과 모델 확장성 측면을 고려하여 개선한 것으로 기존 방법론과의 차이점은 다음과 같다. 첫째, 디바이스의 기하적 특성(Geometry property)과 별도로 제어 정보를 기준으로 디바이스 집합을 구분하기 위한 Device Structure 개념을 통해 SOS-Net 정보가 라이브러리로 관리될 수 있음을 설명하였다. 둘째, 디바이스가 목적하는 행동을 수행할 때, 디바이스가 전이하는 상태를 정적, 동적 및 관측 상태로 분류하고 수학적으로 기술함으로써 보다 사실적인 디바이스 상태설명이 가능함을 보였다. 셋째, 디바이스가 진행하는 프로세스를 본 논문에서 제안하는 SOS-Net 모델로 모듈화 시킬 수 있기 때문에 작업장 공정 흐름에 따라 기술되었던 LD(Ladder Diagram) PLC 코드 보다 단위 디바이스별 모듈화를 통한 공정진행이 기술되는 FB(Function Block) PLC 코드로 변환할 수 있음을 보였다. 넷째, 디바이스의 센서 개수 및 특성으로부터 얻어지는 Paired sensor과 Single sensor 집합 개념을 통해 디바이스 상태를 체계적으로 분류하여 모델링에 사용할 수 있음을 설명하였다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 PLC의 정의 및 제어 정보의 중요성을 소개한 후 제 3장에서 SOS-Net의 정의, 및 구성요소를 설명할 것이다. 제 4장에서는 SOS-Net의 이해를 돕기 위해, 간단한 AGV(Automated Guided Vehicle) 셀을 예로 설명할 것이다. 제5장에서는 SOS-Net을 통한 PLC 코드 생성에 대한 내용을 말하고, 결론에서는 전체적 정리와 향후 연구 방향을 제시 한다.

2. 제어 프로그램과 PLC

생산시스템에 있어서 PLC(Programmable Logical Controller)는 교차로의 신호등과 같은 역할을 한다. 왜냐하면 생산 레이아웃에 맞춰 배치된 설비들은 PLC가 보내는 신호에 따라 동작하고, 정지하기 때문이다. 설비들은 PLC 시작 신호를 통해 대기상태(idle state)에서 동작 상태(working state)로 이동한다. 그리고 동작이 완료된 설비는 자신의 동작 완료 상태를 나타내는 출력신호를 PLC에 보내고, PLC는 이 완료 신호를 이후 작업의 시작신호로 사용한다.

Fig. 2는 PLC, 설비간의 신호 입, 출력 관계를 나타내

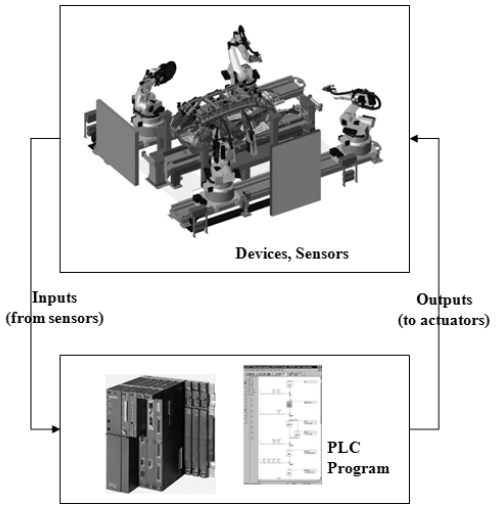

그림 2. PLC에 의해 제어되는 생산시스템 관계도

는 생산 시스템 관계도를 나타낸다. 공정을 구성하는 설비는 입력 포트, 출력 포트 그리고 다수의 센서들을 통해 PLC와 통신을 한다. 그리고 PLC는 공정 설비들과의 통신을 통해 신호를 수집한 뒤 PLC 코드에 적용시켜 프로세스를 순차적으로 진행 시킨다. 설비는 PLC 출력 신호를 바탕으로 작업을 진행하고, 작업완료 상태는 센서를 통해 다시 PLC로 전달된다. 이 같은 프로세스를 갖는 PLC는 스캐닝 사이클(Scanning Cycle)을 통해 프로그램 진행 중에도, 다음 프로세스를 계속 업데이트 하는데, 이는 입, 출력 정보 및 내부 연결(Internal Relay) 신호를 이어주는 불리안 로직 (Boolean Logic)을 스캔 하는 것이다. 사이클을 스캔 하는 동안 출력 및 내부 연결테이블은 계속 업데이트 되어 다음 사이클을 위한 새로운 신호를 준비한다[7].

하지만, 공정 제어에 중추적 역할을 하는 PLC의 코드 작성 및 유지 보수 과정에는 몇 가지 문제점이 존재한다.

① PLC 제어대상은 Low Level의 설비이고, 프로그램 전개가 불리안 언어로 표현되기 때문에 사람이 직관적으로 이해하기는 매우 힘들다[5,13].

② 공정 계획 및 모델링 결과는 사람이 이해하는 High Level의 언어이다. 따라서 모델링 결과를 Low Level의 PLC 언어로 변환하는 과정에서 정보의 손실 및 오류가 발생하기 쉽다[6,12].

③ 기존에 작성된 PLC 코드에 오류를 수정할 경우, 코드로직을 이해하는 과정이 선행되어야 한다. 이때 코드로직의 이해도는 작성자의 경험에 의존되기 때문에 코드를 직접 작성하지 않은 작업자는 오류를 수정하기 위해 많은

시간과 노력이 필요하다.

④ 코드가 체계적으로 모듈화 되어있지 않으면, 유사 공정 간에 코드 재활용이 불가능하다. 따라서 생산라인에 수정 부분이 작을지라도, 매번 코드를 다시 작성해야 한다.

이 같은 문제점을 해결하기 위해 IEC에서는 재사용성, 모듈화를 고려한 IEC-6113 표준 코드 규정을 발표하였다[9,11]. 하지만, PLC 코드 이해, 코드 모듈화 설계 및 유지 보수의 문제는 아직도 해결되지 못하였다. 왜냐하면 코드 작성자는 공정 계획 정보를 하달 받아 로직을 구성하고 코드를 설계, 작성하는데, 코드 설계 및 관리를 위한 특별한 방법이 없기 때문이다. 따라서 공정계획이 변경되면 코드작성자는 반복적으로 코드를 수정하고 생산라인에 테스트 하는 과정을 진행해야 한다.

3. SOS-Net

기존의 PLC 코드 작성 및 정보 재사용에 대한 문제를 해결하고자 본 논문에서는 제어코드 생성을 위한 제어레벨 공정 모델링 방법론인 SOS-Net제안하고자 한다. SOS-Net은 Sequence of State Network의 약자로써, 센서 및 H/W structure 기반의 모델링 방법론이다. 본 논문에서 제안하는 SOS-Net은 기존 고민석 외 3인이 제안한 모델[12]을 수정 보완 한 것으로써, 주된 차별성은 Device Structure를 통한 FB(Function Blcok) 코드 생성과 디바이스 상태 표현의 정형화이다. 이는 기존의 SOS-Net이 표현한 상태 집합의 표현을 동적, 정적, 관찰 상태로 분리하였으며, 이를 디바이스 하드웨어 특성으로부터 추출할 수 있고, 에러 상황 처리가 가능한 관찰 상태가 더해졌다.

Fig. 3는 기존의 공정 설계 및 구현 (Process Design & Realization) 개념에 SOS-Net을 추가했을 때, 이를 활용하기 위한 방법을 나타낸 그림이다. 좌측의 SOS-Net Information은 SOS-Net을 구성하는 설비 정보를 나타내고 우측은 공정을 설계하고 구현하기 위해 진행되는 일반적 절차를 나타낸다. SOS-Net information은 3가지 요소로 구성되는데, 공정에 투입되는 설비 목록(device list), 설비별 하드웨어 구조(device H/W structure) 그리고 설비 상태 목록(device state list)이다. 만약 자동화 라인의 공정 구성 및 레이아웃을 수행할 때, 설비 목록 과 이에 대한 SOS-Net 정보가 있다면, 설비를 눈으로 확인하지 않더라도 설비 동작의 변화를 쉽게 이해 할 수 있다. 따라서 프로세스 디자인 및 OLP 단계에서도 작업자는 개략적 공정 진행 및 검토를 수행할 수 있게 된다.

공정에 투입되는 설비 목록과 각 설비에 대한 정보를

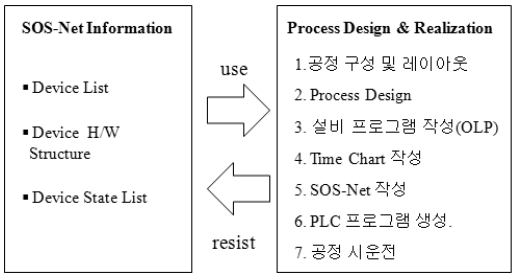

그림 3. SOS-Net의 활용 방안

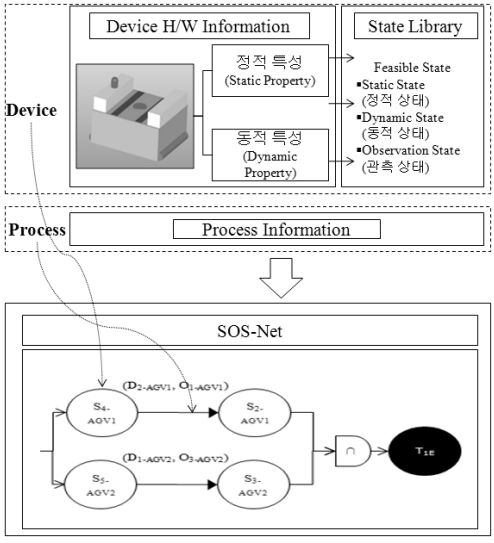

그림 4. SOS-Net 개념 관계도

상태단위로 정리 하는 것은 설비 별 모듈화를 위한 작업이고, 이 모듈화 된 정보는 공정 설계에 직접 사용된다. 만약 공정 정보의 변경이 일어나면, 기존 정보를 수정하여 SOS-Net information에 등록할 수 있고, 이 후 공정에 이를 반영할 수 있게 된다. 이 같은 정보 구조를 이용하여 정보를 체계화 시킨다면, 공정 설계 단계와 코드 작성 단계에 존재 하는 정보 갭을 줄여 PLC 코드 작성 및 유지 보수에 존재하는 문제점을 쉽게 해결할 수 있다.

3.1 상태 라이브러리

Fig. 4는 설비, 프로세스와 SOS-Net의 관계를 나타내는 기본 개념이다. 설비의 하드웨어 정보를 기반으로 만들어진 상태 라이브러리(State Library)와 공정 및 작업 순서를 나타내는 프로세스 정보를 더하여 SOS-Net이 만들어진다. 단일 설비의 예로 직선운동을 통해 파트를 운송하는 AGV를 들 수 있다. AGV의 하드웨어 특성으로부

터 만들어진 상태 라이브러리는 같은 특징을 갖는 AGV 간에는 서로 공유할 수 있다. 반면 프로세스 정보는 작업의 순서를 말하는 정보로써 공정 별로 달라질 수 있다.

설비 하드웨어 정보는 설비의 물리적 구조를 정적 특성과 동적 특성으로 분리시키는 것을 말한다. 이 정보를 바탕으로 설비 상태를 3가지로(정적 상태, 동적 상태, 관측 상태) 분류 하여 상태 라이브러리에 등록할 수 있다. 여기서 정적, 동적 상태는 설비 움직임, 센서 특성을 기반으로 만든 고유의 상태정보이며, 관측 상태는 사용자 정의에 따라 정적, 동적 상태를 조합하여 만든 상태를 말한다.

① 정적 상태(Static State)

정적 상태는 설비가 움직이지 않고 임의의 장소에 정지해 있음을 나타낸다. 즉, 설비가 위치 할 수 있는 정적 위치 집합 중 특정 위치에 놓여 있음을 설명한다. 예를 들어 AGV가 움직일 수 있는 장소의 집합이 {시작위치, 도착위치}일 때, "시작위치"에 정지해 있는 상태를 설명할 수 있다.

② 동적 상태(Dynamic State)

동적 상태는 설비가 시작위치에서 출발하여 도착 위치로 움직이는 상태를 설명한다. 즉, 정적 상태 사이를 전이할 때, 설비가 어떤 구동요소를 사용하여 움직이는지 설명할 수 있다. 예를 들면 AGV 시작위치에서 Motor 1을 사용하여 도착위치로 이동하는 상황을 설명하는 것이 설비의 동적 상태이다.

③ 관측 상태 (Observation State)

관측 상태는 정적, 동적 상태의 조합으로 만들어지는 상태로 설비 동작 오류를 감시하기 위해 사용된다. 상태 전이에서 절대 일어나서 안 되는 상태를 관측하기 위해 사용자가 임의로 조합하여 만드는 상태이다.

설비의 상태를 위와 같이 3가지 상태로 분류한 이유는 다음과 같다. 첫째, 설비 상태를 사람이 직관적으로 이해할 수 있다. 디바이스가 동작하는 상태와 정지해 있는 상태를 직관적으로 기술할 수 있기 때문에 abstract level의 모델링 방법론이 갖는 단점을 극복 할 수 있다. 둘째, 현업에서 PLC 코드를 작성할 때 사용되는 개념을 계승할 수 있다. 현업의 PLC 코드의 70% 정도가 디바이스 및 프로세스의 에러 처리 및 모니터링을 위한 코드이다. 따라서 프로세스 실행이전에 디바이스, 셀, 라인 단위 별 시작 조건 만족을 위해 처리해야 하는 신호가 존재하는데, 이 때 사용되는 개념이 동적, 정적 상태의 개념이다. 예를 들어 인터락(Inter-lock) 신호는 두 개 이상의 디바이스가 협업하는 경우 충돌을 방지하기 위해 사용하는 신호인데, 이 신호가 만들어지기 위해서는 자신 및 상대방의 정적, 동적 상태를 확인해야 한다.

Fig. 4와 같이 상태라이브러리에 등록된 정보를 통해서 디바이스가 놓일 수 있는 모든 상황을 상태의 조합으로 설명할 수 있다. 라이브러리에 등록된 디바이스 상태들은 같은 동작을 하는 설비 간에는 호환되어 사용되기 때문에 보다 효율적으로 운용될 수 있다. 왜냐하면 AGV의 모양과 크기가 다르더라도 정적, 동적 특성이 같다면, 동일한 상태 조합을 사용할 수 있기 때문이다.

설비들은 프로세스에 계획된 정보(Process information)에 따라 순차적으로 진행된다. 따라서 어떤 설비가 어떤 상태를 변화하면서 작업하는지를 명시해 줄 수 있다면, 공정 내에서 일어나는 설비들의 작업 진행을 설명할 수 있다.

이렇게 만들어진 상태 라이브러리와 프로세스 정보를 이용하면 SOS-Net을 만들 수 있다. SOS-Net은 프로세스 순차정보에 따라 디바이스 상태 변화를 설명하기 때문에 설비단계의 Low level의 정보를 보다 사실 적으로 설명할 수 있다. 따라서 기존 High level 모델링 방법론이 갖는 Abstraction level의 문제점을 해결 할 수 있다.

3.2 디바이스 제어 정보 프로토타입

Device Structure(: DS)는 설비의 정적 특성 및 동적 특성을 체계적으로 분류하기 위한 기준 제어 정보이다. 이 정보를 바탕으로 동일한 동적, 정적 특성을 갖는 설비들을 제어 정보 프로토타입(Prototype)으로 그룹화 할 수 있다. 비록 설비 모양이 각기 다를 지라도 같은 Device Structure에 속하는 설비들이라면, 동일한 제어정보를 사용할 수 있다. Fig. 5의 좌측의 AGV 프로토타입은 서로 다른 기하적 특성을 갖는 AGV들을 포괄한다. 왜냐하면 길이, 폭, 높이가 다른 AGV 일지라도 제어 입장에서는 단일 모터에 의해 '전진' 및 '후진' 기능을 하는 하나의 AGV로 볼 수 있기 때문이다.

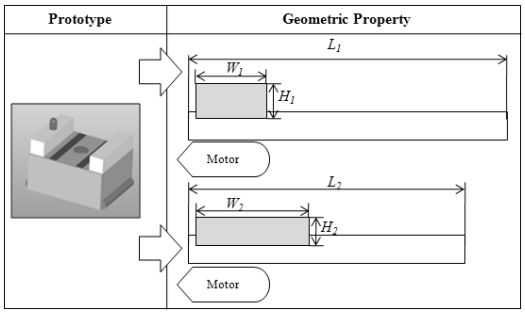

그림 5. Device structure를 이용한 AGV 프로토타입

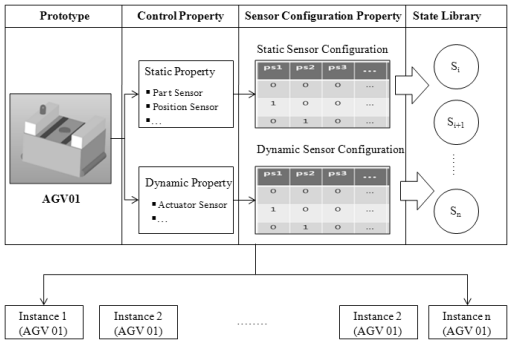

그림 6. Device structure의 개념

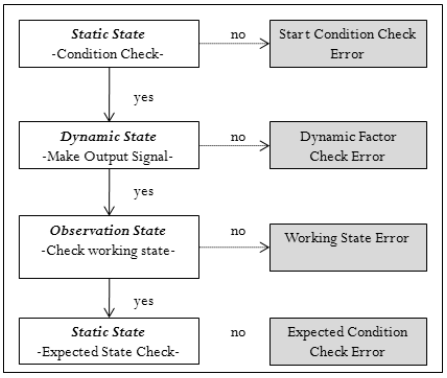

그림 7. SOS-Net 상태 전이 사이클의 일반적 전개

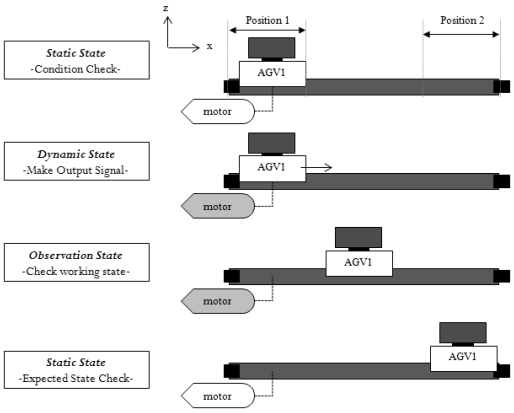

그림 8. 상태 전이 사이클의 AGV 적용

PLC가 설비를 제어할 때, 설비의 기하적 특성과는 상관없이, 센서의 입력신호, 엑츄에이터(actuator)의 출력신호 만을 다룬다. 따라서 제어 정보를 기반으로 디바이스를 분류 하여 제어 프로토타입을 만든다면, 공정에서 다뤄야 하는 디바이스의 종류가 많이 줄어들 수 있다.

본 논문에서 제안하는 Device Structure는 제어 정보를 기반으로 설비들을 분류한 후, 같은 분류의 디바이스가 상태 라이브러리를 공유하도록 하는 구조이다. Fig. 6이 그 구조에 대한 개념도로써, 프로토타입은 제어 정보 그룹을 대표하는 설비 정보 이다. 이 프로토타입은 정적, 동적 특성에 대한 정보를 체계화 시켜 갖고 있고, 이를 바탕으로 정적, 동적 센서 조합 정보를 만들어 낸다. 만들어진 센서 조합은 설비 상태 라이브러리의 개별 상태로 저장되어 관리된다. Fig. 6과 같이 제어 프로토타입에 대한 특성을 체계화 시켜 설비를 분류한다면, 모델링 대상의 수가 줄어들고, 모듈화 된 제어정보가 관리될 수 있다. 즉 같은 프로토타입 으로부터 상속 받은 객체(Instance)들은 동일한 상태정보를 사용할 수 있게 된다.

4. 상태 정의

본 장에서는 각 상태들의 특성에 대한 설명과 AGV 셀 예제를 이용한 SOS-Net 모델링 방법을 설명할 것이다.

SOS-Net은 총 4단계의 상태변화를 거쳐서 하나의 작업 사이클(task cycle)을 완성시키는데, Fig. 7은 사이클의 일반적 전개를 나타내고, Fig. 8은 AGV에 적용한 예이다.

① 초기 조건 확인: 작업(task)를 수행하기 전에 시작 조건 만족(start condition)을 확인한다. 설비의 초기 위치, 직전 작업(pre-task) 완료 조건 등 시작 조건의 만족 여부를 확인하는 상태로써, 정적 상태(static state)로 기술한다.

② 구동 신호 출력: 설비 동작 명령을 위해 구동 신호를 출력시킨다. 출력 신호에 따라 설비가 구동되면, 엑츄에이터 센서에서 동작의 만족을 확인한다. 실제 디바이스의 동작과 관련된 신호로써, 동적 상태(dynamic state)가 이를 기술한다.

③ 구동 상태 관측: 관측 상태는 설비 동작 상태를 모니터링하고, 에러상태를 확인하기 위해 사용된다. 공정 진행 중에 설비의 모든 상태를 확인하는 것은 어렵다. 따라서 사용자가 지정한 특정 에러 조건 및 관리 상태를 포인트로 지정하고 관측하는 것이 모든 상황을 모니터링 하는 것 보다 효율적이다. 즉, 관측 상태는 정적, 동적 신호를 조합하여 만들어 지는 조합 중에서 사용자가 원하는 관측 시점에서의 상태를 확인하고, 만족하지 못하면, 에러 상황으로 판별하는 역할을 한다.

④ 기대상태 확인: 설비 동작이 완료되면, 동작이 완료된 상태가 실제 원하는 기대 상태인지를 확인한다. 만약 기대 상태 조건을 만족하면, 하나의 작업 사이클은 종료되고, 작업 완료 신호를 출력한다.

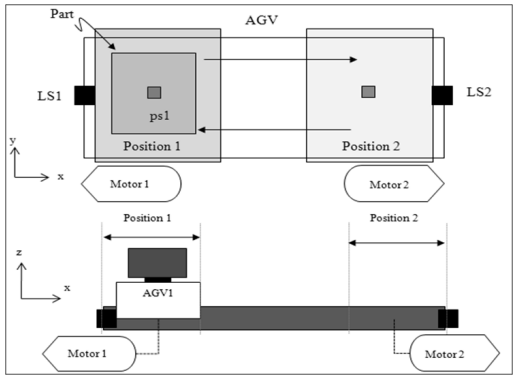

그림 9. AGV 디바이스의 구조 설명

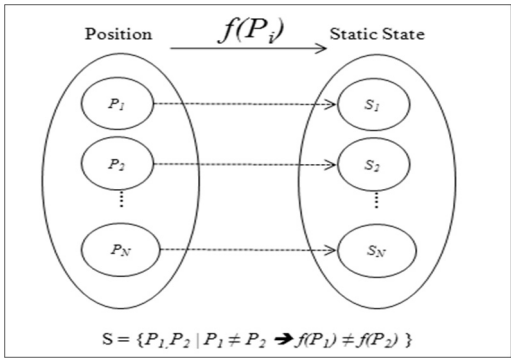

그림 11. Paired sensor set의 관계 정의

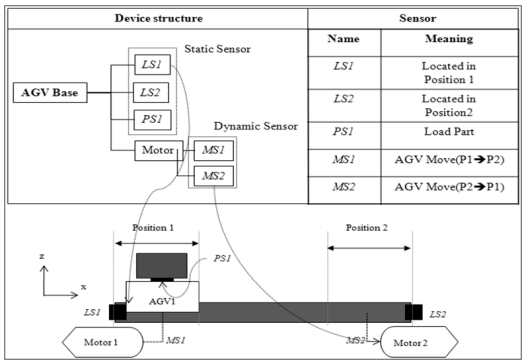

그림 10. AGV의 Device structure

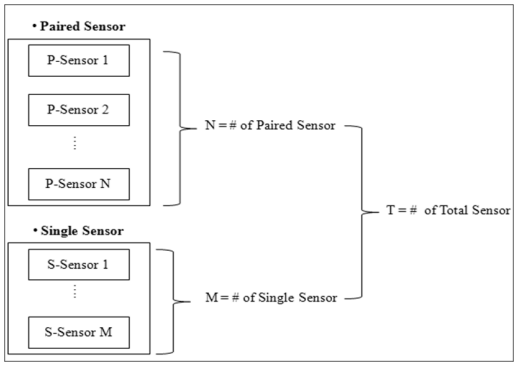

그림 12. Paired sensor set 과 single sensor set

4.1 AGV Cell

본 논문에서는 두 개의 모터를 갖고 파트를 운송하는 AGV 셀을 예로 하여 SOS-Net적용을 설명할 것이다(Fig. 9). AGV는 두 개의 모터 "Motor 1", "Motor 2"로 구동되며, "Position 1", "Position 2"를 위치 집합으로 갖는다.

본 예제의AGV 작업은 Position 1에서 파트를 실은 뒤, 모터1동력을 통해 Position 2로 이동한 뒤, 파트를 내려놓고 모터 2동력을 통해 빈 상태로 Position 1로 복귀하는 것으로 정의한다.

4.2 Sensor signal configuration

Fig. 10은 Fig.9 의 AGV에 대한 Device Structure를 나타낸다. 왼쪽에 AGV Base에 연결된 LS1, LS2는 Position 도착을 감지하는 센서로써, Position과 1:1로 연결된다. 이 센서들은 정적 특성을 감지하는 센서이므로 static sensor로 분류된다. 아래 Motor에 연결된 MS1, MS2는 모터의 구동 여부를 확인하는 엑츄에이터 내부 센서이다. 이 센서들은 동적 특성을 감지하는 센서이므로 Dynamic Sensor로 분류된다. 각 센서의 의미는 Fig. 10의 오른쪽의 표에 정리 되어있다.

Fig. 9 AGV의 제한된 정적 위치의 집합은 {Position 1, Position 2} 이며, 구동요소 집합은 {MS1, MS2} 이다. 각 위치는 static sensor, 구동요소는 dynamic sensor와 1:1로 연결되어 있기 때문에, 동시에 두 개의 센서가 "1(True)"가 될 수 없다는 특성을 갖는다. 즉, 정적 위치 집합 및 구동요소 집합의 n개 원소에 대응하는 n개의 센서는 동시에 1이 될 수 없음을 말한다. 본 논문에서는 이 같은 특성을 갖는 센서의 집합을 "Paired sensor set"이라고 명한다.

Fig. 11은 Paired sensor set의 정의를 나타내는데, 좌측은 설비의 정적 위치 집합, 우측은 정적 센서 집합이다. 예를 들어 설비가 놓여있는 위치가 P1일 때 "true"가 되는 센서는 유일한 *f(P1)*로 정해지기 때문에, 동시에 두 위치센서가 "true"가 될 수 없다.

반면, AGV가 파트를 싣고 있는 지를 감지하는 PS1은 복수 개 원소로 집합을 이루는 센서가 아니고, 센서 하나가 단일 의미를 지니는 센서이다. 따라서 이러한 특징의

표 1. Paired sensor와 single sensor 개수

	Static Sensor	Dynamic Sensor
Paired Sensor	PS (Pared Static)	PD (Paired Dynamic)
Single Sensor	SS (Single Static)	SD (Single Dynamic)
Total Sensor	TS (=PS+SS)	TD (=PD+SD)

표 2. 유효상태 개수

	Static Sensor	Dynamic Sensor
Observation State	2^{SS}	2^{SD}
Feasible State	$PS*2^{SS}$	$PD*2^{SD}$
Error State	$2^{TS}-2^{SS}-PS*2^{SS}$	$2^{TD}-2^{SD}-PD*2^{SD}$

표 3. AGV 디바이스의 의미 있는 상태 개수

Static state	LS1	LS2	PS1
S_1	1	0	0
S_2	0	1	0
S_3	1	0	1
S_4	0	1	1

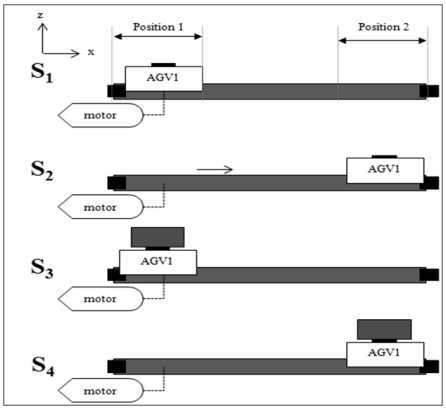

그림 13. AGV의 유효 정적 상태

센서는 Paired sensor와는 구분된 Single sensor로 명한다. 이처럼 Paired sensor 와 Single sensor의 특징은 정적 센서, 동적 센서 모두에서 나타나는 특징으로 Fig. 12 같이 분류할 수 있다.

분류된 Paired sensor와 Single sensor 신호의 조합을 이용해 상태를 보다 효율적으로 정의할 수 있다. 일반적 센서는 "True(1)"와 "False(0)"의 두 가지 값을 갖기 때문에 Total Sensor의 수를 "T"라고 가정할 경우, 신호의 조합(=상태 수)은 총 2^T개가 된다. 따라서 센서 수가 증가하면, 그 조합의 수는 기하급수적으로 증가하게 된다. 하지만, 본 논문에서 제안하는 Paired sensor와 Single sensor의 개념을 이용하면, 많은 센서 조합 중에서 유효한 센서 조합을 가려낼 수 있기 때문에 상태 모델의 복잡성을 줄일 수 있게 된다.

Table 1은 Device structure에 의해 분류된 paired, single 센서의 수를 표로 정리한 것이고 Table 2는 의미 있는 상태의 개수를 나타낸 것이다. Table 1에서 알 수 있듯이 총 센서의 수는 Paired sensor와 Single sensor의 합으로 나타내 진다. Table 2는 Table 1에 표시된 센서의 수를 토대로 한 관측 상태(Observation state), 유효 상태(Feasible state), 에러 상태(Error state) 개수를 계산하는 식을 나타낸 것이다. Device structure에서 얻어진 센서특성을 이용해 3가지 상태 개수를 계산한 것이기 때문에 상태가 쉽게 설명 가능하며, 계산이 명확해진다.

4.3 AGV 정적 상태(Static state)

예제 AGV는 3개의 Static sensor를 갖고 있는데, 그 중 Paired sensor는 {*LS1,LS2*}2개이고, Single sensor는 1개 {*PS1*}이다. 따라서 Table 2의 식을 이용하면 유효한 정적 상태의 수는 $4(=2*2^1)$개를 구할 수 있다. 본래 3개의 Static sensor를 모두 반영한다면 $8(=2^3)$개를 고려해야 하지만 Paired sensor 개념을 이용해 정적 상태 수를 반으로 줄이는 결과를 얻을 수 있다.

Table 3은 Table 2로부터 유도된 AGV의 4가지 유효 정적 상태(Feasible Static State)를 나타내며, Fig. 13은 각 상태에 해당하는 그림을 나타낸 것이다. 예를 들어S1은 파트 없이Position 1에 놓여 있는 AGV의 정적 상태를 설명하며, S_4는 파트를 갖고 Position 2에 놓인 정적 상태를 설명한다. LS1과 LS2는 Paired sensor 이므로 만약, 두 센서가 동시에 "1"이 된다면, 이는 에러 상태로 분류된다.

4.4 AGV 동적 상태(Dynamic State)

예제 AGV는 2개의 Dynamic Sensor{*MS1,MS2*}를 갖고 있으며, 두 센서는 Paired sensor 이다. 따라서 Table 2의 식을 이용하면 유효한 동적 상태의 수는 $2(=2*2^0)$개를 갖는다.

Table 4는 AGV의 2가지 유효 동적 상태(Feasible Dynamic State)를 나타낸 것으로써, SOS-Net의 동적 상

표 4. AGV 디바이스의 유효 동적 상태

Dynamic state	MS1	MS2
D1	1	0
D2	0	1

표 5. AGV 디바이스의 관측 상태

Observation state	LS1	LS2	PS1	MS1	MS2
O1	0	0	0	1	0
O2	0	0	0	0	1
O3	0	0	1	1	0
O4	0	0	1	0	1
O5	1	0	0	0	0
O6	0	1	0	0	0
O7	1	0	1	0	0
O8	0	1	1	0	0

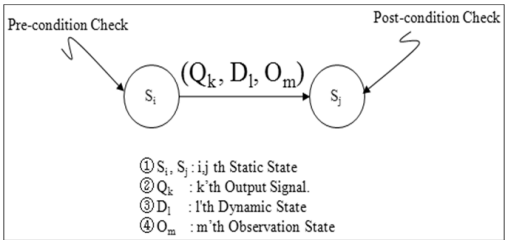

그림 14. SOS-Net의 상태 표현 다이어그램

태 표기와 연결된다. 예를 들어 D1은 AGV가 Position 1에서 Position 2로 이동하는 동작을, D2는 Position 2에서 Position 1로 이동하는 동작을 명령하는 상태이다. Static state와 같은 식으로 MS1과 MS2가 Paired sensor이기 때문에 만약 동시에 "1"이 되면 에러 상태로 분류된다.

4.5 AGV 관측상태(Observation state)

설비의 동작은 특정 정적 위치에서 다른 위치로 이동할 때 일어난다. 예를 들어 AGV의 경우 Position 1에서 Position 2로 혹은 그 반대로 이동할 때, Motor가 동작한다. 만약, 디바이스의 동작 신호가 "1"이 되었음에도 불구하고, 정적 상태를 벗어나지 못하면 올바른 동작수행이 일어나지 않는 에러 상태이다.

정적, 동적 센서를 이용해 에러 상태를 정의하고, 이를 모니터링 하기 위한 것이 관측 상태의 목적이다. 따라서 관측 상태는 Paired Set이 갖는 제약 아래에서, 정적, 동적 센서 모두를 이용해 정의한다. 관측 상태는 크게 두 가지 경우로 나눠지는데, 첫째, 디바이스가 동작을 수행하는 도중에 정적 상태 확인을 위한 관측(=2^{SS}*PD*2^{SD}), 둘째, 디바이스가 정적 상태에 있을 때 동적 상태를 확인하기 위한 관측(=2^{SD}*PS*2^{SS})이 있다.

Table 2 식을 이용하면 관측 상태의 개수는 아래 식 (1)과 같이 정의 할 수 있다.

$$n(\text{Observation State}) = (2SS*PD*2SD)+(2SD*PS*2SS) \quad (1)$$

식 (1)을 이용해 AGV에 적용하여 계산하면 Table 5와 같이 8개의 관측 상태를 구할 수 있다. 즉, 8가지 관측 상태는 5개의 센서를 조합해서 만들 수 있는 총 32(= 25)개 상태 중에서 실행가능 한 8개 상태를 정리한 것이다. 따라서 사용자가 에러 상황을 보고자 하는 프로세스 시점에 관측 상태를 넣어서 모니터링 할 수 있다.

예를 들어 Table 5의 O1은 Fig. 13의 AGV가 파트를 갖지 않고 Position 1에서 Position 2로 이동하는 동작을 관측하는 상태이다. 만약 사용자가 O1을 관측 하려는 구간에서 이 상태를 지나지 않는다면, 이는 에러 상황으로 처리할 수 있다.

4.6 AGV SOS-Net 정의

Device Structure, 상태(Static state, Dynamic state, Observation state)를 정의하면, SOS-Net을 이용해 프로세스를 모델링 할 수 있다. 이를 위해 각 상태가 SOS-Net에서 표현되는 상태 다이어그램 요소는 Fig. 14와 같다.

Fig. 14의 좌, 우측 원호는 정적 상태를 의미하며, 원 내부에 표기된 S_i, S_j 는 단일 디바이스의 i, j 번째 정적 상태를 가리킨다. S_i에서 S_j 로 상태가 전이될 때, PLC에 Q_k 신호를 출력하고 그 결과 D_l의 동적 상태에 놓이게 된다. 이때, 에러 상태 확인을 위해 O_m 의 관측 상태를 체크한다. 즉, 좌측의 정적 상태를 만족 시키면, Q_k 신호를 출력하고, D_k의 상태에 놓인다. 그 후, 동작이 종료 되면, 상태가 우측의 정적 상태(S_j)인지 확인한다. 이 같은 전개는 Fig. 7의 SOS-Net의 일반적 전개와 그 내용을 같이한다.

Fig. 15는 SOS-Net을 구성하는 3종류의 노드이다. 정적 상태와 연결되는 정적 노드(Static Node), 단위 작업 시작과 끝을 나타내는 터미널 노드(Terminal Node), 입력 노드들의 신호를 Bit 연산하는 연산노드(Compute Node)로SOS-Net이 구성된다.

Fig. 16은 AGV 셀의 SOS-Net 모델이다. 모델의 진행 방향은 좌에서 우측방향이며, 모델에 표기된 상태 이름은

Name	State	Meaning
Static Node	s_1	Static State 의 State 들 중에 "s_1" 상태
Terminal Node	T_{1S}	"Task 1번의 시작(S)", "Task 1번의 끝(E)"
Compute Node	∩	입력신호의 Bit 연산을 수행하는 노드.

그림 15. SOS-Net의 상태 표현의 의미

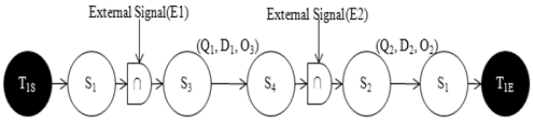

그림 16. AGV 셀의 SOS-Net 모델

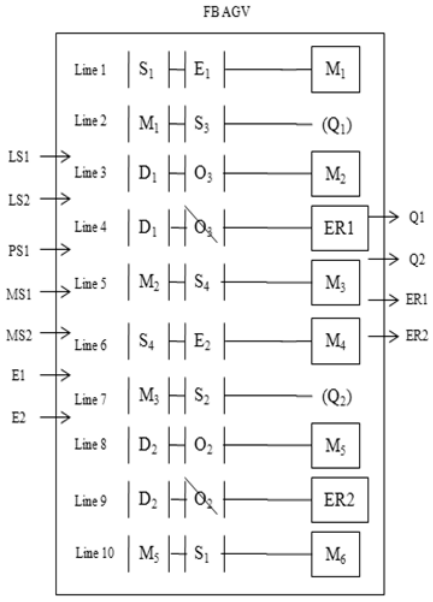

그림 17. AGV 셀에 사용되는 PLC FB code

Table 3, 4, 5의 내용을 사용하였다.

① T_{1S}: AGV를 이용한 공정의 시작을 나타내는 터미널 노드로써, 작업 시작을 위해 만족시켜야 하는 조건을 체크한다. 즉, 안전 조건 체크, 직전 작업 완료 체크 등 작업 진행을 위해 사용자가 정의한 시작 조건을 만족하는지 확인하는 역할을 한다.

② S_1: 파트를 갖지 않은 AGV가 Position 1에 놓여있는 정적 상태이다.

③ E_1: 외부 디바이스의 작업 완료 신호로써, AGV 위에 파트를 올려놓는 작업이 완료되었음을 알리는 신호이다.

④ S_3: 파트를 갖고 있는 AGV가 Position 1에 놓여있는 정적 상태를 이다.

⑤ Q_1: S_1, E_1, S_3 상태를 만족시키면, D_1이 가리키는 출력신호 Q_1 신호를 산출한다. 산출된 Q_1 신호는AGV Motor 1을 구동시켜 AGV를 Position 1에서 Position 2로 이동시킨다.

⑥ D_1: Q_1 신호를 통해 움직이고 있는 AGV의 동적 상태를 나타낸다.

⑦ O_3: 디바이스가 Motor 1을 이용해 이동할 때, 관측하고자 하는 상태이다. 만약 AGV 가 O_3의 신호 조합을 만족시키지 못하면 에러 신호(ER)를 출력한다.

⑧ S_4: 파트를 갖지 않은AGV가 Position 2에 놓여있는 정적 상태이다.

⑨ E_2: 외부 디바이스의 작업 완료 신호로써, AGV위에 파트를 내려놓는 작업이 완료되었음을 나타내는 신호이다.

⑩ S_2: 파트를 갖고 있지 않은 AGV가 Position 2에 놓여있는 정적 상태이다.

⑪ Q_2: S_4, E_2, S_2 상태를 만족시키면, D_2가 가리키는 Q_2 신호를 산출한다. 산출된 Q_2신호는 AGV Motor 2를 구동시켜 AGV를 Position 2에서 Position 1로 이동시킨다.

⑫ D_2: Q_2 신호에 의해 AGV가 정상 작동할 경우 놓이는 동적 상태이다.

⑬ O_2: 디바이스가 Motor 2를 이용해 이동할 때, 관측하고자 하는 상태이다. 만약 AGV 가O_2의 신호 상태를 만족시키지 못하면 에러 신호(ER)를 출력한다.

⑭ T_{1E}: SOS-Net을 통한 디바이스 모델링 사이클 종료를 나타내는 노드이다.

5. PLC 코드 생성

PLC 코드는 PLC 기기의 입력 점과 출력 점으로부터 나오는 신호를 이용해서 로직을 전개시키는 역할을 한다. 가장 널리 사용되는 PLC 코드 종류는 LD(Ladder Diagram), SFC(Sequence Function Chart), FB(Function Block)가 있다. 특히 FB는 코드의 모듈화가 가능하기 때문에, 유지 관리 측면에서 장점을 갖는다[10].

디바이스 하드웨어 정보와 신호 정보를 기반으로 제어 모델을 만들고, 이를 통해 PLC 코드를 생성하는 것이 SOS-Net의 목적이다. 사용자 마다 PLC 코드의 종류를 달리 할 수 있겠지만, 본 논문에서는 모듈화 특성을 갖는FB에 AGV 셀을 적용 하였다.

Fig. 16의 SOS-Net모델을 FB로 변환하면 Fig. 17와 같은 FB 코드를 얻을 수 있다. 우선 입력 신호의 집합은

AGV의 정적 센서 신호{LS1, LS2}, 동적 센서 신호{MS1, MS2}, 외부 신호{E1, E2}로 정의되며, 출력신호의 집합은 디바이스 구동 신호{Q1, Q2}, 에러 신호{E1, E2}로 정의된다. 이 같이 외부 입력, 출력 요소는 Device Structure에 정의된 센서 정보를 기반으로 하기 때문에 체계적으로 정의될 수 있다.

Fig. 17의 FB 로직 전개와 Fig. 16의 SOS-Net 상태 정보 의미를 연결하면 다음과 같다.

① Line 1: S_1 와 E_1 을 동시에 만족 시키면, 메모리 신호 M1이 "1"이 된다.

② Line 2: M_1 와 S_3 을 동시에 만족 시키면, 디바이스 구동신호 Q1을 출력한다.

③ Line 3: D_1 와 O_3 을 동시에 만족 시키면, 메모리 신호 M2가 "1"이 된다.

④ Line 4: D_1 을 만족시키는 상태에서 O_3을 만족 시키지 못하면 에러신호 "ER1"을 출력한다.

⑤ Line 5: M_2 와 S_4를 동시에 만족 시키면, 메모리 신호 M3이 "1"이 된다.

⑥ Line 6: S_4 와 E_2를 동시에 만족 시키면, 메모리 신호 M4이 "1"이 된다.

⑦ Line 7: M_3 와 S_2를 동시에 만족시키면, 디바이스 구동신호 Q_2를 출력한다.

⑧ Line 8: D_2 와 O_2를 동시에 만족시키면, 메모리 신호 M_5이 "1"이 된다.

⑨ Line 9: D_2을 만족시키는 상태에서 O_2를 만족 시키지 못하면 에러신호 "ER2"을 출력한다.

⑩ Line 10: M_5 와 S_1을 동시에 만족 시키면, 메모리 신호 M_6이 "1"이 된다.

각 내부 메모리 신호는 라인간의 로직 연결과 조건 만족여부를 모니터링 하기 위해 사용한다. SOS-Net과 PLC 코드의 직접 연결은 제어정보 모델링 결과를 실제 코드로 사용한다는 점에서 의를 갖는다.

6. Conclusion

기존의 PLC 코드 작성에 있어서 가장 큰 문제는 체계적인 프로그램 설계 및 개발 프로세스 방법론의 적용이 미흡하고, 프로그래머의 경험에 의존하여 작성된다는 점이었다. 따라서 일반 배치도, 공압 회로도, 타임테이블 등이 포함되어 있는 기계 도면과 공법관련 도면, 설비 사양서를 참고하여 작성자 본인만 알아볼 수 있는 방법으로 문서화 하거나 문서화 없이 코드를 직접 작성하는 경우가 빈번하게 발생한다. 이 같은 코드 작성 체계화의 부재로 인하여 유사 공정의 PLC 코드의 재활용이 일어나지 못하고 있으며, 코드 해독 및 검증에 소요 되는 시간이 전체 공정 설계 및 검증에 소요되는 시간 중 대부분을 차지하고 있다[12]. 특히 PLC 코드 작성 및 검증에 소요되는 시간은 공장 시운전 시간과 직결되기 때문에 기업 입장에서 많은 손해를 감수할 수밖에 없다. 따라서 만약 코드 작성 및 관리의 체계가 확립된다면 많은 경제적 이익을 얻을 수 있다.

공정 및 구성 설비에 대한 정보 체계화를 통해서 제어 코드를 생성하기 위해 본 논문에서는 제어정보를 기술하는 표준 방법론인 SOS-Net을 제안하였다. FSA, Petri-Net과 같은 표준화된 모델링 방법론을 이용한 공정 정보의 체계화 연구가 많이 있었다. 하지만 그 범위가 너무 넓고, 추상적이기 때문에 실제 현업의 제어 정보 및 코드에 사용되기에는 무리가 있다.

SOS-Net은 공정을 구성하는 디바이스에 대한 하드웨어적 정보를 기반으로 Device Structure를 작성한 후, 이 정보를 기반으로 제어 정보를 기술한다. 여기서 제어 정보란 디바이스가 갖는 신호를 정적, 동적 신호로 분류한 뒤, 이들 중 사용가능 한 제어 신호를 선별하여 모델링에 사용하는 것을 말한다. 디바이스 동작정보를 기반으로 Device Structure를 만들기 때문에, 공정내의 많은 디바이스 들이 같은 그룹으로 묶이게 되며, 결과적으로 다뤄야 하는 정보의 양이 적어진다. 또, 이렇게 작성된 제어정보는 후에 지속적으로 사용가능하기 때문에, 새로운 라인 작성 시 참조할 수 있는 정보의 틀이 된다.

Device Structure는 실제 디바이스의 동작, 센서 정보를 바탕으로 작성된다. 따라서 이를 이용해 모델링 된 SOS-Net은 제어 정보의 표준 모델로 사용될 수 있다. 그러므로 SOS-Net을 이용해 작성된 PLC 제어 코드는 쉽게 해석 될 수 있고, 후에 재사용 될 수 있다.

본 논문에서는 SOS-Net을 이용한 코드생성의 예시로 모듈 화된 PLC 코드의 한 종류인 FB 형식을 사용하였다. 간단한 AGV에 대한 Device Structure, 신호 조합 그리고 상태를 이용하여 SOS-Net을 만들고, 이를 바탕으로 AGV FB를 생성하는 일련의 과정을 보였다. 비록 간단한 예제이지만, 디바이스 정보로부터 제어 정보를 체계화 시키는 일련의 과정을 체계적으로 기술하였다.

추후 연구 과제는 SOS-Net을 통해 생성된 제어코드가 현업에 적용될 수 있는 방안을 연구할 것이다. 특히 에러 처리 및 모니터링에 대한 내용을 연구할 것이다. 왜냐하면 현업의 제어코드 중 90%이상이 에러 처리에 관한 내

용이기 때문에 이를 배제한 코드 생성은 큰 의미를 갖지 못하기 때문이다.

참 고 문 헌

1. J. S. Yang, S. H. Han, D. H. Mun, "Sharing Product Data among Heterogeneous PDM System Using Open PDM", *Transactions of the Society of CAD/CAM Engineers*, Vol. 13, No. 2, pp. 89-97, 2008.
2. E. Park, D. Tilbury, P. P. Khargonekar, "Modular logic controllers for machining systems : formal representation and performance analysis using Petri nets.", *IEEE Trans Robot Automat*, Vol. 15, No. 6, pp. 1046-1061, 1989.
3. P. J. Remadge, W. M. Wonham, "The control of discrete-event systems", *Proc. of IEEE*, Vol. 77, No. 1, pp. 81-98, 1989.
4. S. C. Lauzon, A. K. Ma, J. K. Mills, B. Benhabib, "Application of discrete-event-system theory to flexible manufacturing.", *Control Syst Mag IEEE*, Vol. 16, No. 1, pp. 41-48, 1996.
5. J. Richardson, M. Fabian, "Modeling the control of flexible manufacturing cell for automatic verification and control program generation", *Int J Flex Manuf Syst*, Vol 18, No. 10, pp.191-208, 2006.
6. J. Richardson, M. Fabian, "Automatic Generation of PLC programs for control of flexible manufacturing cells" *Proc. of the IEEE International Conference on Emerging Technologies and Factory Automation, San Francisco*, Vol. 2, pp. 337-344, 2003.
7. S. C. Park, C. M. Park, G. N. Wang, "A PLC programming environment based on a virtual plant", Int *J Adv Manuf Technol*, Vol. 39, pp. 1262-1270, 2008.
8. W. B. Lee, C. H. Roh, "Design of A PLC Program Simulator for nuclear Plant Using Compiler Technology", *한국시뮬레이션 학회 논문지*, Vol. 15, No. 1, pp. 11-17. 2006.
9. G. Frey, T. Hussain, "Modeling Techniques for Distributed Control System based on the IEC 61499 Standard-Current Approached and Open Problems", *Proc. of the 8th International Workshop on Discrete Event Systems*, Michigan, USA, pp. 176-181, 2006.
10. A. Rullan, "Programmable logic controllers versus personal computers for process control.", *Computers and Industrial Engineering*, Vol. 33, No. 5, pp. 421-424, 1997.
11. B. M. Younis, G. Frey, "Formalization of existing PLC Programs: A Survey", *Proceedings of CESA 2003*, France, NO.S2-R-00-0239, 2003.
12. M. S. Ko, S. H. Hong, G. N. Wang, S. C. Park, "SOS-Net for generation of PLC program", Transactions of the Society of CAD/CAM Engineers, Vol. 14, No.1, pp. 60-68, 2009.
13. W. B. Lee, C. H. Roh, "Design of A PLC Program Simulator for nuclear Plant Using Compiler Technology", *한국시뮬레이션 학회 논문지*, Vol. 15, No. 1, pp. 11-17. 2006.

고 민 석 (sebastianminsuk@hanmail.net)
2007 아주대학교 산업정보시스템공학부 학사
2007 아주대학교 산업공학과 대학원 석사
2009 아주대학교 산업공학과 대학원 박사 과정

관심분야 : Programmable Logic Controller(PLC), Simulation, CAD/CAM, FMS, Factory Automation

곽 종 근 (jkwak@ajou.ac.kr)
1989 포항공대 전자계산학과 학사
1994 포항공대 컴퓨터공학부 석사
2007~ 아주대학교 산업공학과 대학원 박사 과정

관심분야 : Computer science, FMS, Factory Automation, Simulation, Image Processing

왕 지 남 (gnwang@ajou.ac.kr)
1983 아주대학교 산업공학과 학사
1985 한국과학기술원 산업공학과 석사
1992 미 Texas A&M 산업공학과 박사
1993～현재 아주대학교 산업정보시스템공학부 정교수

관심분야 : Intelligent Information & Manufacturing System, System Integration & Automation

박 상 철 (scpark@ajou.ac.kr)
1994 한국과학기술원 산업공학과 학사
1996 한국과학기술원 산업공학과 석사
2000 한국과학기술원 산업공학과 박사
2004～현재 아주대학교 산업정보시스템공학부 부교수

관심분야 : Digital Manufacturing System, CAD/CAM, CAPP, Manufacturing System Modeling & Simulation