FIELDBUS Tutorial

- SMAR

1. 소개

Fieldbus의 역할이 필드장치를 제어실 장비(단일 스테이션 제어기 또는 DCS에 상관 없이)에 연결하는 4-20mA 신호의 디지털 대체로 만 이해하였다면 실제 기능을 알고난 후 놀랄 것이다. 그것은 Fieldbus는 우리가 알고 있는것보다 훨씬 많은 기능을 제공하기 때문에 상기 예는 극히 일부에 지나지 않기 때문이다.

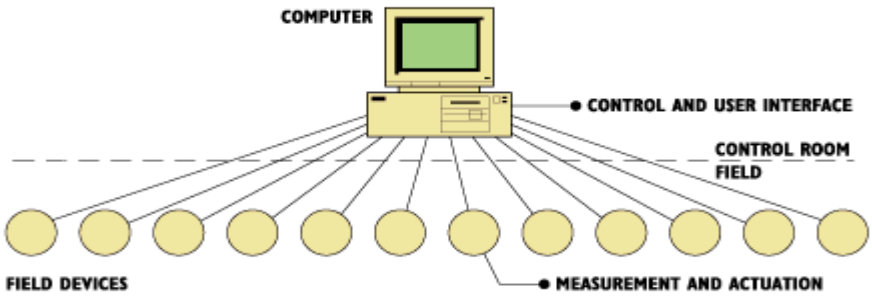

그림 1.1

DDC(Direct Digital Control) 시스템 제어실의 단일 컴퓨터에 제어가 집중되어 있었던 초기의 컴퓨터 시스템.

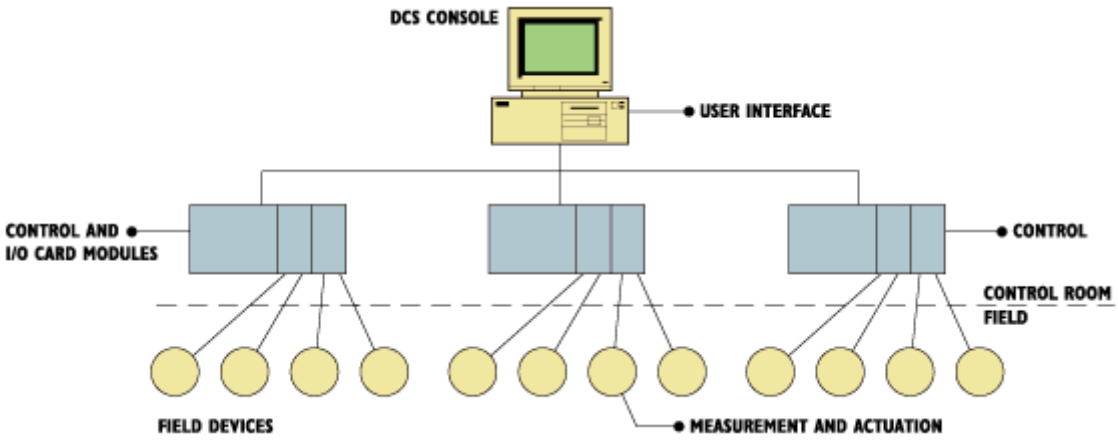

그림 1.2

DCS(Distributed Control System) 제어는 여전히 제어실에서 이루어지지만 몇 개의 제어 카드에 일부가 분산되어 있으며 각각에 몇 개의 루프가 존재한다.

Fieldbus 선택 이유

Fieldbus는 새로운 지능형/스마트 전송 프로토콜일 뿐만 아니라 다음 4가지의 특징적인 표준을 제공해주고 있다.

- 4-20mA의 완벽한 디지털 대체
- 제어, 경보, 경향 및 필드 장치에 할당된 기타 기능
- 상호 운용 가능한 타사 제품과의 호환성
- 개방 시스템 : 사용권 제한 없이 사용 가능

한 규정

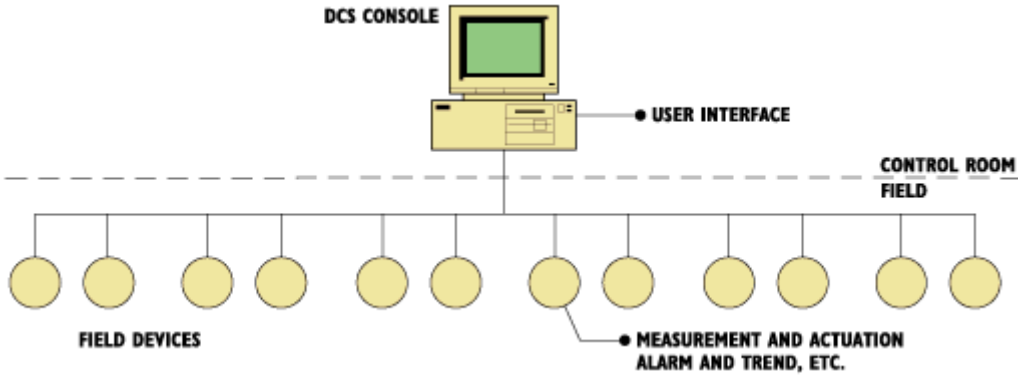

그림 1.3

Fieldbus 시스템 - 각 장치루프와 함께 제어가 필드에 완전히 분산되어 있다.

Fieldbus는 기존 지능형 전송기로는 불가능한 기능을 제공하기 때문에 Fieldbus를 스마트 장치라 칭하는 경우 이전과 특이할 것이 없는 것으로 오해하기 쉽다. Fieldbus는 제어 기능이 필드 장비에 분산되어있는 완전한 시스템으로서 디지털 통신을 통해 마찬가지로 제어실에서 작동과 튜닝을 수행할 수 있다. Fieldbus는 제어 기능이 하나 이상의 “제어용카드”에 집중되어 있었던 기존 4-20mA 및 전통적인 DCS를 대체할 수 있다.

일부 제조 업체는 자사의 DCS가 수년 전부터 Fieldbus 기능을 제공하였다고 주장하지만 어떤 경우도 앞에서 기술한 4 가지 기준을 만족하지는 못하였다.

Fieldbus는 여러 공급 업체간에 상호 운용이 가능한 프로토콜로서 4-20mA 표준과 마찬가지로 독일과 미국의 관련단체 및 기업의 표준화 작업으로 시작되었으며 현재 세계의 많은 현장에 채택되고 있다. 대부분의 주요 계측기 제조 업체는 Fieldbus의 단일 표준 제정을 제창하여 왔고 대부분의 표준과 마찬가지로 이들이 취한 접근 방식은 상용제품의 출시 전에 표준을 확립하는 것 이었지만 이는 잘못된 방식이었던것으로 판단된다.

4-20mA를 통한 양방향 디지털 통신의 일부 장점과 기타 우수한 특성은 기존 스마트 전송기 프로토콜에서도 찾을 수 있다.

- 높은 신뢰성 및 데이터 신뢰성
- 다중 변수(Multi-variable) 액세스
- 원격 구성 및 진단
- 배선 감소
- 기존 “아날로그 배선” 활용

높은 신뢰성은 디지털 통신에서 기인하는데 예를 들어, 전송기 및 컨트롤러 등의 마이크로프로세서는 D/A 및 A/D 변환을 거치지 않고 직접 통신할 수 있다. 측정 및 제어 데이터와 함께 상태가 전달되기 때문에 정보의 신뢰성 여부를 판단할 수 있다. 노이즈 또는 아날로그 미검출 신호에 영향을 줄 수 있는 임피던스 불일치로 인한 왜곡에 대해 모든 데이터를 점검하여 이러한 왜곡이 발생하지 않도록 한다. 다중 변수 액세스라는 것은 압력 전송기를 예로 들면 단지 압력에 대한 출력만 전송하는 것이 아니라 공정 온도에 대한 정보도 알려줄 수 있다는 것을 의미한다. 또 다른 예

로는 동일 장치에서 컨트롤러의 조작 변수나 세트포인트에 대한 액세스 또는 온도 전송기에 대한 다중 채널 입력을 들 수 있다.

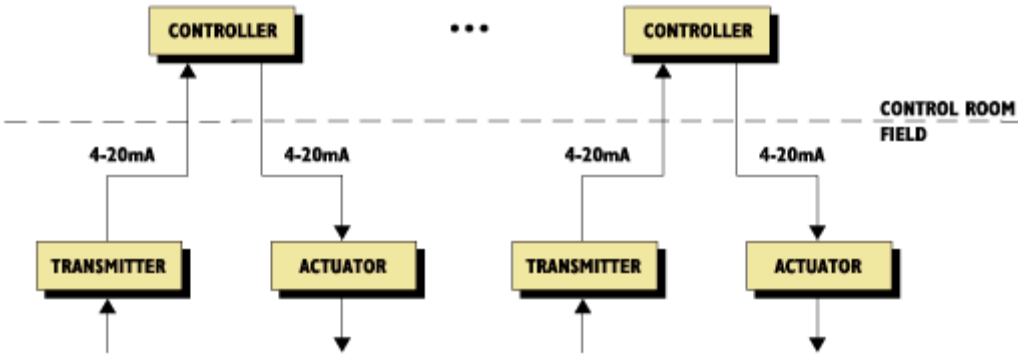

그림 1.4

아날로그 신호 각각의 서로 다른 기기들이 물리적으로 4-20mA로 연결된다

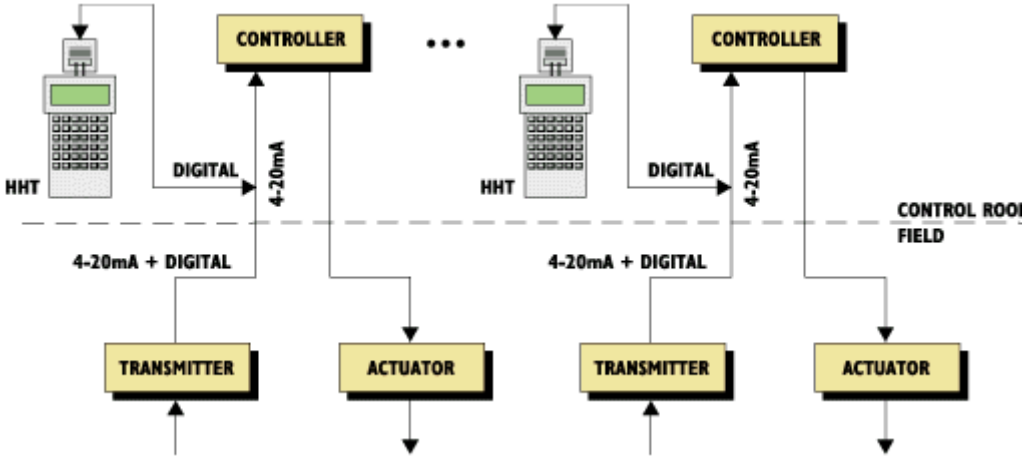

그림 1.5

아날로그와 디지털 신호의 지능형/스마트 혼합형-- 폐쇄 루프 제어에 관계하는 개별 기기가 물리적으로 4-20mA에 연결된다. 구성 및 진단에 대한 디지털 신호도 함께 나타내었다.

디지털 통신을 통해 전체 구성을 원격으로 변경할 수 있고 교정은 어떤 입력이나 출력 측정 없이도 작동 중 수행할 수 있으며 자가 진단 상태도 점검할 수 있다.

여러장치를 단일 배선쌍으로 연결(multidropping) 함으로써 배선의 양과 복잡성을 줄일 수 있고 모든 연결은 병렬 형태로 단순화 되었고 터미널 수 일치가 최소이기 때문에 비용을 줄이고 구형 전송기를 쉽게 교체할 수 있다.

4-20mA와 비교하여 기존 프로토콜의 문제점 및 단점도 일부 제기되었는데 다음과 같다.

- 폐쇄 루프 제어에 대한 통신 속도가 너무 느리다.
- 상이한 형태나 상이한 제조 업체 장치 간에 호환성이 거의 없다.
- 상태 정보를 얻으려면 장치에 폴링

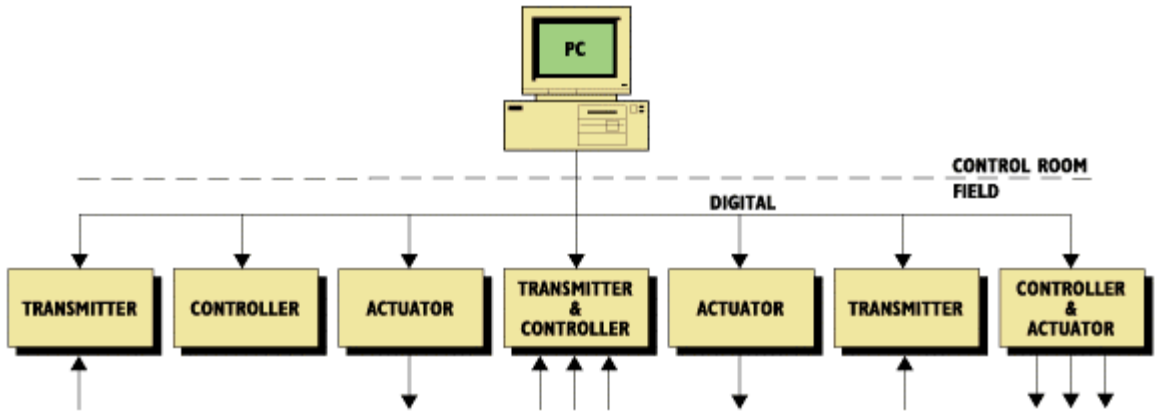

그림 1.6

완전한 Fieldbus 디지털 통신　하나의 물리적인 배선이 다중변수를 논리적으로 연결하는 매개 역할을 한다.

(polling)을 수행해야 한다.
- 장치 간에 연동하지 않는다.

Fieldbus는 최저속 옵션의 경우에도 대부분의 일반 스마트 전송기 프로토콜보다 25배 빠르고 훨씬 효율적이며 기밀한 폐쇄 루프 제어를 보장하고 있다. 저속 Fieldbus 버전은 아날로그 및 스마트 전송기와 동일한 배선 형태를 사용하도록 설계되었기 때문에 대체가 용이하나 멀티드롭 기능을 사용하려면 전송기를 반드시 병렬로 연결해야 한다.

대부분의 스마트 전송기 프로토콜은 범용성이 없으며 제조업체마다 다르다. 사용자는 이러한 표준 결여로 인해 한 시스템을 선택하는 경우 그 제조업체에서 생산한 제품만 사용해야 하는 상황에 처하게 된다. 만약 그 제조업체가 긴급한 부품 교체를 제공할 수 없거나 특정 형태의 전송기를 구비하고 있지 않은 경우 사용자는 4-20mA 방식으로 되돌아갈 수 밖에 없다. 즉, 사용자는 최신 기술이나 기능을 제공하지 못하는 단일 공급 업체에 의존해야 하며 이는 공급업체에게 가격을 임의대로 책정할 수 있는 빌미를 제공할 수 있다.

고유 시스템의 상반된 개념은 "개방" 시스템이다. 개방 시스템은 곧 바로 사용 가능한 표준에 기반함으로써 여러 공급 업체가 상호 운영성이 있는 하드웨어나 소프트웨어를 제공할 수 있도록 한다.

4-20mA 장치가 기타 모든 동일 형태의 장치를 대체할 수 있는 능력을 "상호 운용성"이라 하며 호환성과 유사한 의미로 사용되며 Fieldbus도 동일한 기능을 제공한다. 상호 운용성을 통해 "Y"라는 전송기 제품을 "X" 라는 동일 형태의 전송기로 기능 손실 없이 언제든지 교체할 수 있으며 또 다른 "Z"라는 장치에 인터페이스할 수 있다. Fieldbus는 이러한 표준을 준수하는 장치 간에 상호 운용성을 필수적으로 요구하며 사용권 계약 없이 모든 제조업체가 사용할 수 있다. Fieldbus는 "개방"적이며 완전히 노출되어 있어서 "비밀"이 존재하지 않는다. 사용자나 제작사는 고유한 구성

및 소프트웨어를 제작할 수 있으며 가격, 성능, 품질 및 공급 시기에 기반하여 장치를 선택하고 4-20mA와 마찬가지로 각 형태의 최상품을 선별하여 구성할 수 있다. 사용자는 이미 설치된 것과 동일 형태의 장치를 구성하기 위해 특정 회사에 의존할 필요가 없는것이다. (Fieldbus를 사용하지 않는 경우, 사용자는 특수한 통신 드라이버를 개발해야 한다). 상호 운용성이 주는 또 다른 혜택은 새로운 제품을 도입할 때 시스템 소프트웨어를 업그레이드할 필요가 없다는 것이다.

표준이 없는 경우, 기존 지능형 필드 장치의 스마트 기능을 폭 넓게 사용할 수 없다. 예를 들어, 기존 시스템 중 스마트 전송기의 멀티드롭 기능을 활용한 경우는 매우 드물다. 스마트 장치에 자가 진단 기능이 있지만 장치에 폴링을 수행하는 경우에만 상태를 알 수 있다. 통신은 통상적으로 교정 시에만 수행되기 때문에 대부분의 어플리케이션에는 멀티드롭 기능이 없다. 장애를 발생 전에 발견하려면 사전에 이를 파악하고 Query 해야 한다.

Fieldbus가 사용자에게 주는 혜택은 너무나도 명백하여 사용자가 결정권이 있었다면 Fieldbus는 이미 훨씬전에 국제 표준으로 채택되었을 것이다. 최종 사용자는 표준의 개발에 대한 정보나 이에 기여할 만한 기회를 거의 갖지 못했다. 최종 사용자는 일단의 제조업체가 제공하는 한 가지 기술을 사용할지 또는 앞으로 대두될 더욱 진보된 기술을 사용할지를 결정해야 하는 어려운 상황에 처해있는 것이다. 이는 몇 년 전 가정용 비디오 녹화 기술로 어떤 것을 선택해야 할지 고민해야 했던 상황과 유사하다. 사용자가 우려하는 것은 "잘못된" 비디오 기술의 선택과 마찬가지로 "잘못된" 프로토콜을 선택하면 어쩌나 하는것이다.

많은 제조 업체도 나름대로의 목적으로 Fieldbus를 채택하고자 한다. 제조 업체가 Fieldbus를 통해 진정한 혜택을 얻으려면 양질의 전송기나 밸브를 사용하여 완벽한 제어 시스템 패키지를 구성해야 하며 이를 위하여 시스템을 새로 설계하여 완전한 DCS를 개발할 필요는 없다. 구성 및 관리 소프트웨어와 같은 MMI(man-machine interface) 인터페이스는 관련 전문 업체가 작성할 수 있기 때문에 필드 장치 제조 업체는 이러한 노고를 덜 수 있으며 MMI 인터페이스 전문 제작 업체의 경우에도 새 프로토콜에 대한 계속적인 통신 드라이버 개발 노력을 덜 수 있다. 제조 업체는 사용자들이 "잘못된" 시스템 구입을 우려하여 Fieldbus가 적용되지 않은 시스템의 구매를 꺼리는 것을 인식하고 있기 때문에 자사제품이 Fieldbus 지원이 가능하도록 하고 있다.

HART는 세계적 범위로 채택된 스마트 전송기 프로토콜이며 Fieldbus 기세가 주춤하는 틈틈이 입지를 강화하고 있다. HART는 자체 안전성 요건을 충족하고 상호 운용성이 상당한 수준에 이르렀지만 알려진 것 이상으로 전송속도가 큰 제약 요소로 작용하고 있다. 개방형 프로토콜로 빠른 속도에 적합한것으로는 PROFIBUS 및 FIP가 있고 PROFIBUS의 경우 본질적인 안전을 위해 IEC 표준을 적용하였고 버스가 장치에 전원공급도 두가닥선만으로 전원 및 데이터 전송이 가능하.

Fieldbus로 주목

사용자가 Fieldbus를 결정하여 얻을 영향은

지대할 것이다 다음 주요 관건을 살펴본다.

- 훨씬 높은 작동 안정성
- 제한 없는 유연성
- 장비 투자 비용 절감
- 설치 비용 절감
- 대량의 정보

Fieldbus를 통한 시장 점유 속도는 주로 사용자의 교육 방식에 따라 달라집니다. 어떤 물건의 작동 방법을 모르는 경우, 그 물건을 구매하려는 용기가 생기지 않는 것과 마찬가지입니다.

아날로그 기술의 단순성으로 이해가 용이하며 이것이 사용자의 편안함을 이끌어내는 주된 이유입니다. 십자 드라이버만 사용하여 4-20 mA 장치를 작동시킬 수 있으며 아주 단순한 전류계로 테스트를 수행할 수 있습니다. 누구라도 이러한 장치를 구성하고 문제를 해결할 수 있습니다.

필드 장치는 장애 및 문제 발생을 즉각 보고하기 때문에 유지관리 요원은 오류를 곧바로 정확히 찾아내고 피해를 미연에 방지할 수 있습니다.

Fieldbus의 다중 변수 기능을 통해 필드에서 제어, 가산(totalization) 및 기타 신호 처리가 가능하기 때문에 별도의 컨트롤러나 기타 신호 조절 장치가 필요하지 않습니다. MMI 소프트웨어를 설치한 일반 PC를 호스트로 사용합니다. 단일 배선에 여러 장치를 멀티드롭하면 필요한 배선 양을 크게 줄일 수 있습니다. 많은 작업장의 경우, 제어실에서 장치까지의 거리는 수 km 또는 그 이상이 될 수 있습니다. 모든 루프에는 적어도 두 쌍의 배선(전송기용 하나, 액츄에이터용 하나)이 필요하기 때문에 정유소에는 이러한 루프가 수백 개 존재할 수 있습니다. 중소 규모 사업장의 경우 배선에서 얻는 비용 절감은 상당히 큽니다. 전송기 및 액츄에이터는 보통 서로 인접해 있으나 멀티드롭에 이상적인 장소인 오퍼레이터 콘솔에서는 멀리 떨어져 있습니다. Fieldbus 장치의 초기 소유 비용은 높은 편이지만, 장치 및 배선 그리고 관련 배선함 및 마셜링(marshaling) 상자의 감소를 통해 더 저렴한 비용으로 시스템을 구성할 수 있습니다. 제조업체는 더 이상 고 비용의 고유 기술에 의존할 수 없으며 Fieldbus가 개방된 경쟁을 조성하여 궁극적으로 가격 인하를 유도합니다.

다중 변수 액세스로 얻는 정보는 제어실을 가득 채우고도 남기 때문에 전통적 리코더로는 이를 처리할 수 없으며 전자 리코더 및 컴퓨터 저장이 이를 대체합니다. 이러한 정보는 통계 처리 제어 및 기타 공정 관리에 사용할 수 있습니다.

그 외에, Fieldbus에는 소프트웨어 함수 블록이 포함되어 있어 현재 하드웨어로 수행되는 많은 함수를 대체합니다. 이를 통해 하드웨어를 다시 연결하거나 변경할 필요 없이 제어 방식을 수정할 수 있기 때문에 매우 큰 유연성을 얻을 수 있습니다. 물리적 연결을 마친 후 함수 블록 간에 논리 연결을 변경하거나 함수 블록을 추가 및 제거할 수 있습니다. 더 발전된 장치는 거의 무제한으로 함수 블록을 실행할 수 있습니다. 시스템을 확장하거나 성능을 향상시켜야 하는 경우 단지 기존 장치가 블록을 추가적으로 실행하도록 하여 하드웨어 추가를 최소화할 수 있습니다.

Fieldbus 요건

Fieldbus의 주요 요건은 4-20mA 표준의 장점(4-20mA의 주요 장점은 폐쇄 루프를 기밀하게 제어하는 것입니다)을 그대로 살리면서 스마트 전송기 프로토콜의 문제점을 극복하는 것이었습니다.

통신 속도 및 장치 전력 공급에 대한 다양한 옵션을 제공하면 자체 안전성 및 최소 통신 지연에 대한 요건을 충족시킬 수 있으며 네트워크 사용을 최적화하면 자체 안전성이 요구되는 곳에서 기밀한 폐쇄 루프 제어를 얻을 수 있다.

4-20mA 기술의 경우 전송기, 컨트롤러 및 액츄에이터만으로 제어 루프를 구성할 수 있다. Fieldbus 장치도 이러한 기능이 있어야 할 뿐만 아니라 더 큰 규모의 제어 시스템에서 작동할 수 있어야 한다. Fieldbus는 다용도성이어야 하며 4-20mA 못지 않은 다기능성을 제공해야 한다. 따라서, Fieldbus 장치는 소규모 시스템에서 경제성을 높이기 위해 사용자 인터페이스를 통해 자체적으로 작동할 수 있어야 한다. 전용 소프트웨어를 탑재한 호스트 컴퓨터 및 DCS는 저 비용이더라도 소규모 시스템에 적합하지 않다. 사용자 및 제조 업체가 소규모 시스템에 아날로그 기술을 계속 사용할 수 밖에 없는 경우 이들이 직면하게 되는 논리상의 문제도 존재하기 때문이다.

많은 장치를 연결하는(각 장치가 기존의 여러 장치 기능을 수행할 수 있는) 시스템에서 발생 가능한 복잡성을 해결하기 위해 사용이 용이한 사용자 인터페이스가 필요하다. 스마트 전송기 프로토콜에서 볼 수 있는 수동 주소 할당 및 PLC에서 수행하던 비트, 바이트, 워드 및 메모리 주소에 대한 고된 추적 작업을 위해 사용자가 고생하는일을 덜어주어야 한다.

제안된 모든 Fieldbus에서는 함수 블록 모델을 사용하고 있다. 여기서 장치는 ISA 및 SAMA의 블록과 마찬가지로 블록으로 표현되기 때문에 사용자가 장치를 쉽게 작동시킬 수 있다. 물리적 배선은 블록 간의 연결인 논리 연결 또는 "소프트 배선" 형태로 이루어진다. 기술적으로 다르지만, 이는 매우 유사하며 사용자의 편이를 제공한다. 장치 주소 및 파라미터 인덱스는 자동으로 할당되며 Smar의 CD600 컨트롤러를 포함하는 일부 시스템은 이미 유사한 방식을 구현하고 있다.

표준화는 상호 운용성을 보장하지만 너무 엄격할 경우 사용자에게 역효과를 가져올 수 있다. 제조업체간의 차별화를 위한 가능성은 남겨놓아야 한다. 표준이 모든 세부 조항까지 준수할 것을 요구하는 경우, 모든 제조 업체의 장치가 획일화 되므로 사용자는 실질적으로 선택의 여지를 잃게 되며 어떤 제조업체가 우수한 발상을 하더라도 구현이 불가능하거나 표준에 포함시키는 과정에서 경쟁사에 이를 노출시키게 된다.

Fieldbus가 기본적 기능 요건을 설정하더라도 제조업체는 그들 장치에 고유한 기능을 추가할 수 있어야 한다. 이러한 측면은 사용자에게 혜택을 제공하며 제조업체는 이를 마케팅 수단으로 사용할 수 있기 때문이다. 마찬가지로 새로운 모델이 이전 모델과 다른 경우 이와 통신하는 장치는 모델을 분별할 수 있어야 한다. 요컨대, 제품 개발 및 성능 개선 과정에서 Fieldbus가 방해 요소로 작용해서는 안

된다는 것이다. 이에 따라 Fieldbus는 제조업체별 기능에 대해서도 상호 운용성을 보장할 수 있는 메커니즘을 제공하고 있다.

시스템 중단으로 인한 비용은 생산 손실의 측면에서 매우 크기 때문에 Fieldbus는 작동 중에도 시스템을 재구성할 수 있는 기능을 제공해야 한다.

소규모뿐만 아니라 대규모 시스템에서 Fieldbus 구현의 필요성은 전체 Smar 302 Fieldbus 장치 라인을 개발할 때 고찰하였다. 이때 Fieldbus 장치는 네트워크에서 마스터로 작용하고 자기(magnetic) 도구를 사용하여 지역적으로 구성할 수 있는 공통된 특징을 보임으로써 많은 기본적 어플리케이션에서 구성기 또는 호스트의 필요성을 제거하였다.

2. Fieldbus의 작동 방식

시스템 구조는 두 가지 주요 부분 즉, 상호연결 및 어플리케이션으로 나누어진다. 상호연결은 필드 장치, 오퍼레이터 콘솔 또는 configurator 등 모든 장치 간의 데이터 전달을 의미하며 Fieldbus의 통신 프로토콜 부분을 이룬다. 어플리케이션은 시스템이 수행하는 자동화 기능으로써 Fieldbus는 어플리케이션 부분을 표준화함으로써 기타 어떤 통신 표준보다 진보된 형태이며 이러한 규칙을 준수하는 제품간의 상호 운영성을 보장한다.

개요

Fieldbus 어플리케이션 구조는 네트워크를 통해 상호 연결된 필드 장치로 자동화 작업을 분산시킨다. 장치가 수행하는 가장 기본적 기능은 블록으로 모델화되고 서로 연결되어 있는 이 블록들은 상호 작용하여 장치 간의 전달 파라미터 및 오퍼레이터를 지원한다.

Fieldbus 상호 연결 구조는 ISO (International Organization for Standardization)에 의해 개발된 OSI(open systems interconnect) 참조 모델의 3 계층 하위 집합 구조에 기반하고 있다. OSI 어플리케이션 및 시스템 관리 및 Fieldbus 어플리케이션 구조 모델도 OOP (Object Oriented Programming) 개념에 기반한다. OSI 및 OOP 모두 기능을 쉽게 이해하도록 모델을 사용한다. Fieldbus에 대한 이해를 돕기 위해 이 두 가지를 다음과 같이 간단하게 소개해 보았다.

OSI 모델

OSI 참조 모델은 개방형 네트워크를 바탕으로 한 네트워크 구조에 대한 표준으로서 국제적으로 공히 사용되고 있다. 이 표준은 모든 수준의 원격 통신에 대한 모델로 개발되었다. 네트워크의 모든 기능(주소 지정, 에러 점검 및 인코딩/디코딩 등)은 계층이라는 모두 7개의 논리 집합으로 그룹화된다. 각 계층은 위로 쌓아 올려지며 이 적층을 프로토콜 스택(protocol stack)이라 부른다. 어떤 계층은 스택에서 직접 접하고 있는 위, 아래 계층과만 인터페이스 하고 스택은 위로 어플리케이션과 인터페이스하며 아래로 전송 매체와 인터페이스 한다. 전체 어플리케이션 중 시스템이 장치에서 수행하는 이 부분을 *AP (application process)*라 부르는데 AP는 두 부분 즉, 기능을 의미하는 사용자 부분과 통신 부분으로 이루어진다. Fieldbus에서 사용자 부분은 측정, 제어(함수 블록) 또는 사용

자 인터페이스와 같은 실제 장치 기능이다.

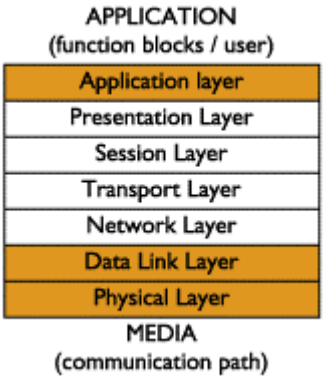

그림 2.1 : OSI 프로토콜 스택

각 계층은 상위 계층에 대한 서비스를 제공하며 다른 스테이션의 대응 계층(*피어*)과 통신하는데 이를 *피어 투 피어 통신(peer-to-peer communication)*이라 하며 계층이 상위 계층에 제공하는 기능 집합을 *서비스*라 부른다.

한 어플리케이션에서 다른 어플리케이션으로 데이터를 전송하는 과정을 살펴보면, 데이터는 물리 계층인 *프레임*을 얻는 각 계층이 처리하여 스택의 꼭대기에서 바닥으로 전달하고 이 프레임이 매체를 통해 전달되며 다른 종단부에서 데이터를 수신하여 스택의 바닥에서 꼭대기로 전달한다. 수신 종단에서는 대응하는 전송 종단 계층이 수행한 처리 과정을 반대로 수행하여 원래 어플리케이션 데이터를 얻는다.

Fieldbus(그리고 대부분의 기타 LAN)는 이러한 계층의 목적인 네트워크 간의 상호 연결을 갖지 않기 때문에 계층 3에서 6은 사용하지 않는다. 이러한 단순성으로 인해 필드 장비와 같이 제한된 처리 능력을 갖는 장치에 Fieldbus를 쉽게 구현할 수 있으며 속도를 더욱 높일 수 있는것이다.

나머지 3 계층과 제공 기능 및 절차상의 특징은 다음과 같다.

1. *물리 계층(Physical Layer)* : 통신을 위한 비트 흐름을 투명하게 전달하는 물리 연결을 미디어에 상관 없이 활성화, 유지 및 비활성화한다. 이는 개별 비트만을 인식하며 문자 또는 다중 문자 프레임은 인식하지 않는다. 표준에서는 매체와 신호 형태, 전송 속도 및 노드 수를 포함한 토폴로지 그리고 장치 전원(Fieldbus의 경우만)을 정의한다.

2. *데이터 연결 계층(DLL)* : 네트워크 개체 간에 데이터를 전송한다. 데이터 링크 연결의 활성화, 유지 및 비활성화, 비트를 문자 및 메시지 프레임으로 그룹화, 문자 및 프레임 동기화, 에러 제어, 매체 액세스 제어 및 흐름 제어(여러 장치가 네트워크를 공유하도록 함)의 역할을 한다. 표준에서는 미디어 액세스 제어 유형, 프레임 포맷, 에러 점검 및 스테이션 주소 지정* 등을 정의한다.

7. *애플리케이션 계층(Application Layer)* : 분산 시스템을 지원하기 위해 일련의 로컬 및 통신 서비스에 대한 액세스를 제공한다(여러 AP와 사용자 간의 상호 연결). 표준에서는 AP가 사용할 수 있는 메시지 포맷과 서비스를 정의한다.

**참고: 실제로, 주소 지정은 Fieldbus에서 정의되지 않는 네트워크 계층 부분이며 DLL에서 대신 수행한다.*

계층 3에서 6의 일부 기능은 Fieldbus 프로토콜 내에 존재하며 어플리케이션 계층에서 구현하고 있다.

OSI 네트워크 관리는 OSI 계층을 확장한 것이며 모든 계층에 영향을 미친다. 시스템 관리는 네트워크 자원의 운영 상태를 감시하고 제어하며 시스템 관리의 기능별 영역으로 세분된다. Fieldbus의 경우, 가장 중요한 기능별 영역은 네트워크 구성으로써 시스템 관리는 한 스테이션에서 다른 스테이션으로 접속된다. 이는 관리자 역할을 수행하는 관리 시스템, MIB(Management Information Base)를 갖으며 에이전트 역할을 수행하는 관리 대상 시스템으로 모델화된다. MIB는 네트워크 관리를 지원하기 위한 정보 및 자원의 논리적 저장소이다.

최종 사용자는 물리적 연결 및 어플리케이션에 주로 관심을 갖고 있다. 이전에 언급했듯이 물리 계층은 이미 표준화되었으며 무선기와 같은 기타 매체를 포함시키기 위해 그 내용이 확장될 것이 확실하지만 변경되지는 않을 것이다. 제기된 모든 Fieldbus 프로토콜이 사용자 어플리케이션에 대해 거의 동일한 솔루션을 제안하기 때문에, 사용자는 Fieldbus를 "즉시" 사용할 수 있다. 따라서, 사용자는 시간 낭비 없이 Fieldbus의 설치하고 어플리케이션 및 작동 측면을 배울 수 있다. 어플리케이션, 데이터 연결 계층 및 관리 기능이 불확실한 상태이지만 사용자에게는 신뢰성을 주며 제품 개발자에게는 관심의 대상이다.

Fieldbus의 OSI

Fieldbus에서는 계층 1, 2 및 7만 사용되며 어플리케이션 기능은 함수 블록이 제공한다.

Fieldbus 장치에는 3 가지 AP 즉, 함수 블록 어플리케이션, 네트워크 관리 및 시스템 관리가 존재한다.

Fieldbus 물리 계층

4-20 mA 시스템과 유사한 기능을 수행하여 폐쇄 루프를 제어하고 변속 드라이브를 사용하는 경우 고속 데이터 전송이 필요한데 높은 속도를 얻으려면 전력 소비가 크기 때문에 일부 어플리케이션에서 자체 안전성 요건과 상반되다. 따라서 적합한 통신 속도를 선택하였으며 자체 안전성이 없는 고속 옵션도 사용 가능하게 하여 모든 어플리케이션의 필요사항을 만족시켰다. 이 시스템은 저속 옵션에서도 제어 요건을 만족시키기 위해 통신 오버헤드를 최소화하도록 설계하였다.

물리 계층은 여러 가지를 조합 사용할 수 있으며 각각 상대적인 장점을 갖는다. 버스 상의 모든 장치에 대한 매체, 연결 및 전송 속도는 동일한 옵션을 사용해야 한다. 그러나, 장치의 자체 안전성 여부에 관계없이 버스를 탑재 또는 탑재하지 않은 장치를 혼합 사용할 수 있다.

물리 매체 옵션:

- 와이어
- 광섬유(pending)
- 무선(pending)

전송 속도 옵션:

- 31.25 kbit/s
- 1 Mbit/s
- 2.5 Mbit/s

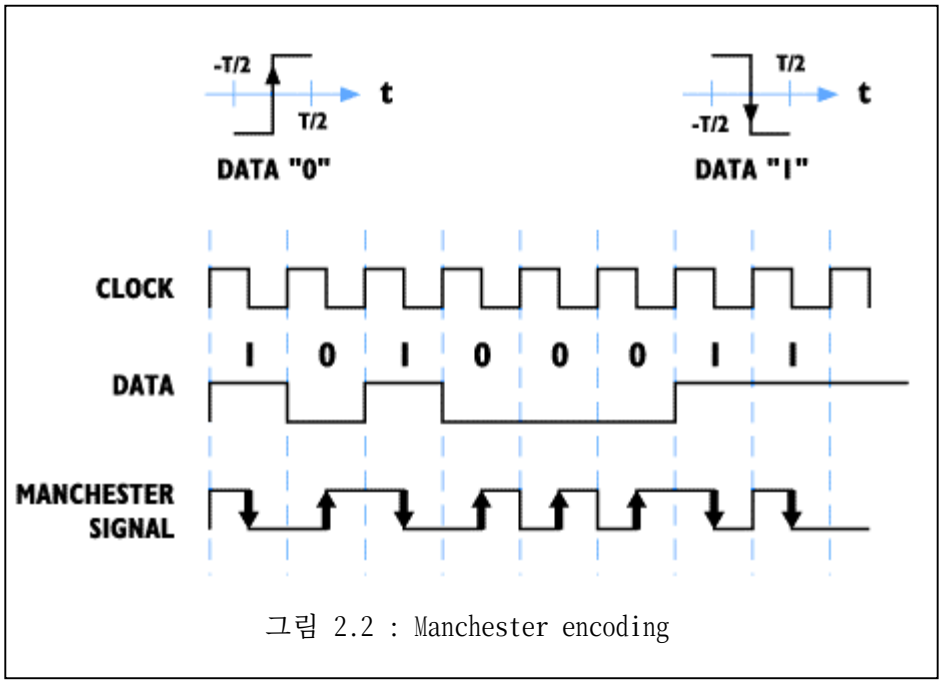

그림 2.2 : Manchester encoding

에서는 클럭과 데이터가 결합되어 있으며 rising edge가 논리 0 데이터를 falling edge가 논리 1 데이터를 나타낸다. 전송 시 전화의 "벨" 신호에 해당하는 preamble(추가 정보)이 우선적으로 전송되어 기타 장치의 수신기를 동기 시킨다. 메시지의 시작과 끝은 각각 시작 및 끝 구분 기호로 나타낸다. 구분 기호는 Manchester 인

공통적인 매체 특징

데이터는 직렬 *반이중(half-duplex)* 신호 형태로 동기적으로 교환된다. 장치는 동일 매체에서 데이터를 송수신하지만 동시적이지는 않다. 신호는 Manchester(Biphase L로도 불림) 인코딩을 사용하여 자체적으로 클럭을 전달한다. 전송은 동기 방식이기 때문에 시작 및 중지 비트가 필요하지 않다. Manchester 코딩

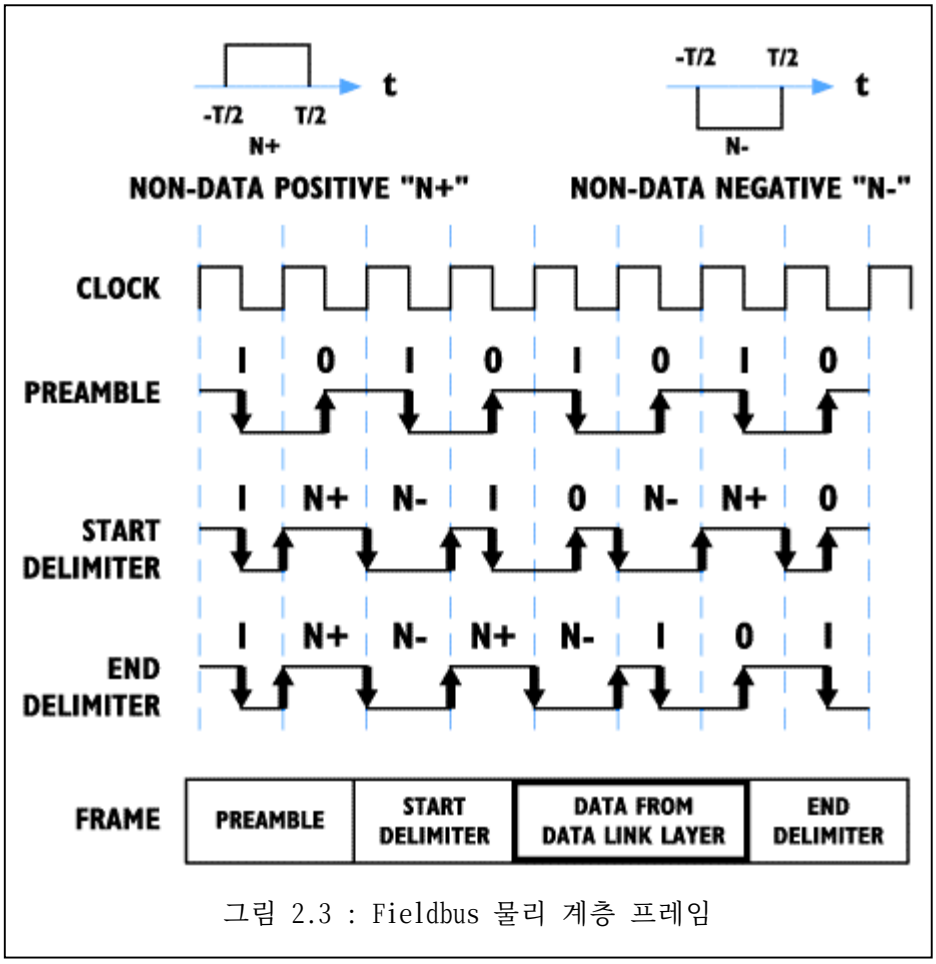

그림 2.3 : Fieldbus 물리 계층 프레임

코딩에 포함되지 않기 때문에(데이터만 인코딩) 고유하게 식별할 수 있다. 구분 기호의 인코딩되지 않은 비트를 N+ (Nondata positive) 및 N- (Nondata negative)로 부른다. 전송 장치의 물리계층에 추가되는 preamble과 구분 기호는 수신 장치의 물리 계층에서 제거된다.

- 1 Mbit/s (voltage mode): 750m
- 1 Mbit/s (current mode): 750m
- 2.5 Mbit/s: 500m

장치가 멀티드롭된 경우, 통신 하드웨어인 *MAU(Media Attachment Unit)*를 접지에서 분리하여 전기 접지 루프를 방지해야 한다.

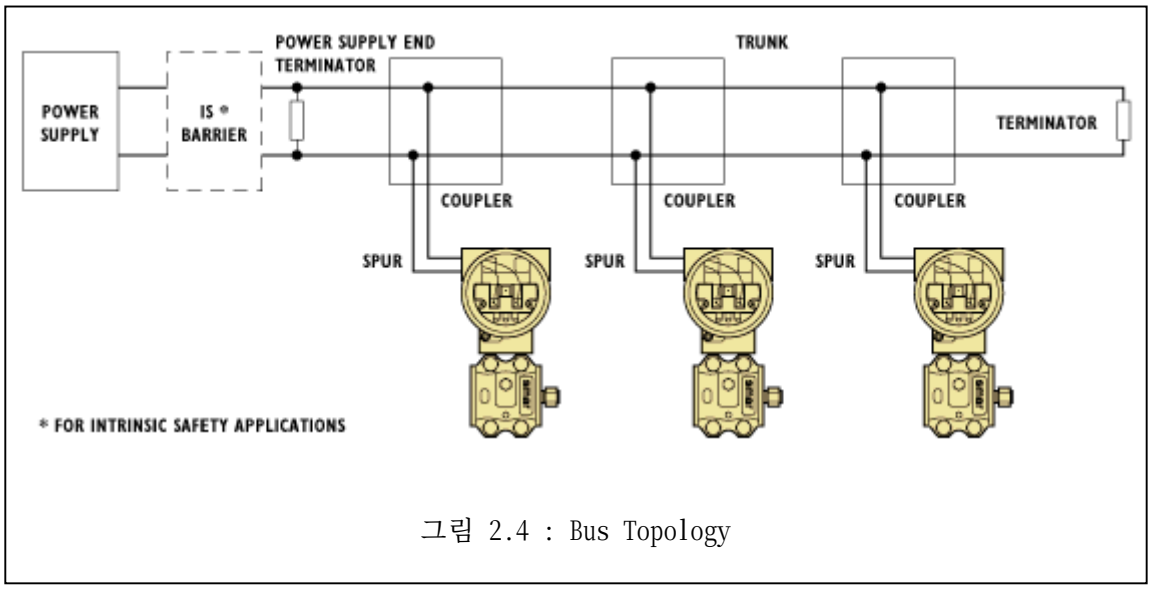

그림 2.4 : Bus Topology

이중화는 물리 계층 및 매체를 복제하여 얻을 수 있다. Voting(시간 초과와 유효성에 기반)은 장치가 두 매체 중 어느 것을 사용할 지 제어한다.

Wire Media 특성

Wire Media는 표준 트위스트 페어선(twisted pair wire)을 통한 전기 신호를 사용하며 이미 1992년 IEC/ISA 표준으로 승인되었다.

와이어 매체를 사용하는 두 장치간에 허용된 최대 길이는 선택한 전송 속도에 따라 다르다.

- 31.25 kbit/s: 1,900 m

Bus Topology(그림 2.4), Tree Topology(그림 2.5) 및 지점간(point-to-point) Topology를 지원한다. Tree Topology 는 저속 버전에서만 지원된. 버스에는 두개의 터미네이터가 부착된 트렁크 케이블이 있고 장치는 분기선(spur)을 통해 트렁크에 연결된다. 분기선은 장치에 통합시킬 수 있으며 이 때 분기선의 길이는 0이 된다. 분기선은 길이에 따라 장치를 하나 이상 연결할 수 있으며 분기선 길이를 확장하기 위해 Active 커플러를 사용할 수 있다. 트렁크 길이를 확장하는 경우에는 Active repeater를 사용한다.

터미네이터는 전송 주파수 근처에서 각 100Ω의 임피던스를 갖도록 설계되었다. 장치는

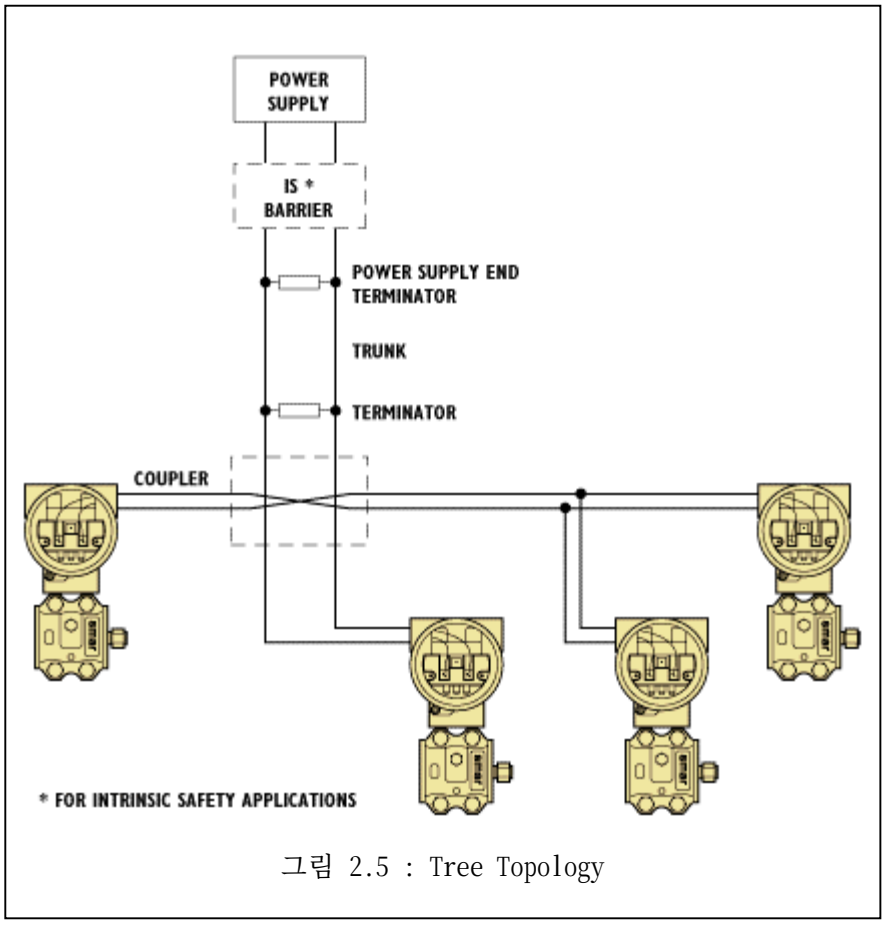

그림 2.5 : Tree Topology

Manchester 인코드된 신호에 따라 네트워크 전류를 변조하여 전송을 수행한다. 수신 장치는 전류 변조에 따라 두 터미네이터 간에 발생하는 전압 강하를 감지한다. 저속 버전의 경우 변조 전류의 피크 간 범위는 수신기 감도 150mV에서 15~20 mA 이다.

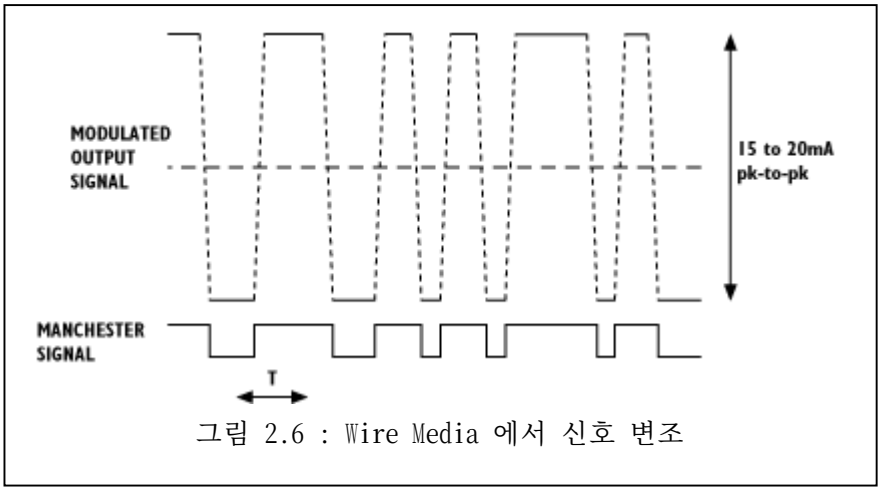

그림 2.6 : Wire Media 에서 신호 변조

31.25 kbit/s 와이어 매체 특성

최저속 와이어 매체 옵션인 31.25 kbit/s는 가장 다용도로 사용할 수 있으며 가장 널리 사용될 것으로 예측된다. 이 옵션에서는 자체 안전성(intrinsic safety)버전과 버스에 의한 전원 공급을 제공한다. 장치 수는 다음 선택에 따라 제한된다.

- intrinsic safety / nonintrinsic safety
- 버스에 의한 전원 공급/ 별도 전원 공급

일반적인 장치의 수를 표 2.1에 나타내었다. 실제 수는 장치 종류에 따라 다르다.

자체 안전성 시스템에서 안전 장벽(safety barrier)은 전원과 전원 종단 터미네이터 사이에 위치해야 한다.

장치는 버스로 구동되기 때문에 전원 공급과 통신을 위한 두 개의 와이어가 필요하고 모든 장치에 공통적인 단상 전원은 트렁크 종단에서 네트워크에 연결되며 전압 범위는

9~32VDC 이다. 전원의 임피던스는 통신 신호를 단락 시키지 않도록 전송 주파수 근처에서 최소 3kΩ 이상이어야 한다. Manchester 코딩 신호의 듀티 주기(duty-cycle)는 정확히 50% 이고 AC 신호여야 한다. 따라서 장치의 DC 전원 소비는 일정하다.

	intrinsically safe	non intrinsically safe
bus powered	2-6*	2-12**
separately powered		2-32

* 1-4 in the hazardous area
** 1 at power supply end

표 2.1 : 최대 네트워크 노드 수

Fieldbus DLL -- FDL

Fieldbus DLL은 두 개의 하위 계층으로 이루어지며 아래 부분은 FMAC(Fieldbus Media Access Control)이며 위 부분은 FDLC (Fieldbus Data Link Control) 이다.

Fieldbus 네트워크 상에 있는 장치는 다음 두 스테이션 중 하나에 속한다.

- 마스터 스테이션
- 슬레이브 스테이션

마스터 스테이션은 매체에 대한 액세스 권한(통신 초기화)을 부여받으며 슬레이브는 마스터의 요청에 대한 응답 권한만을 갖는다.

FMAC(Fieldbus Media Access Control)

Fieldbus 매체 액세스는 토큰 전달 및 폴링 원리를 결합한 방식으로써 하나의 네트워크 상에서 여러 장치가 마스터일 수 있으며 토큰을 소유한 스테이션만이 통신을 초기화할 수 있다. 마스터는 토큰을 보유한 상태에서 슬레이브 장치에 폴링(마스터가 요청하고 슬레이브가 응답)을 수행한다.

토큰은 특수 프레임을 통해 다음 마스터로 전달된다.

장치에는 스테이션 주소가 개별적으로 부여되고 모든 프레임에는 메시지를 위한 목적 주소(Destination Address) 및 원시 주소(Source Address)가 포함된다.

Fieldbus는 사용자가 주소를 할당하고 추적해야 하는 노고를 덜어준다.

데이터의 신뢰성이 제어의 신뢰성을 결정하기에 모든 프레임 데이터에 대한 2바이트의 FCS(Frame Check Sequence)가 전송 장치의 다항식을 사용하여 계산되며 프레임에 추가된다. 수신 장치는 동일한 계산을 수행하고 그 결과를 FCS와 비교함으로써 에러를 탐지한다. FCS는 패리티 비트 및 비동기 프로토콜의 주기적 이중화 점검에 상당하는 기능이다.

위 계층이 FDL에 메시지 전달을 요청하는 경우 메시지 우선권이 이와 함께 전달된다. 우선권에는 높음(예, 경보) 및 낮음(구성 및 진단 데이터)의 두 가지 종류가 있다. FDL은 높은 우선권 메시지를 우선적으로 전송한다.

FDLC(Fieldbus Data Link Control)

FDLC는 어플리케이션 계층이 다른 스테이션에

데이터를 보낼 수 있는 다양한 수단을 제공한다.

시스템에서 확인할 수 있는 메시지 유형에는 다음의 두 가지 종류가 있다.
- Operational(작동)
- Background(배경)
작동 소통량은 통신 방식의 일환으로 장치 간에 전달되는 데이터(예, 공정 변수)이며 작은 볼륨, 시간 민감 및 주기성을 특징으로 한다. 배경 소통량은 장치와 오퍼레이터 인터페이스 간에 전달되는 데이터(예, 구성 및 진단)로서 작동 소통량과 반대로 큰 볼륨, 시간 민감 및 비주기적(간헐적)인 특징을 갖는다.

객체 지향적 설계

Fieldbus 어플리케이션과 같은 복잡한 시스템을 운영할 때는 시스템의 전체적 작동 방식을 이해하기 힘들 수 있다. 계층 구조의 더욱 원시적 요소를 적합한 추출 수준으로 분해하면 (본질적 특질을 이해하여) 복잡한 상호 작용 부분들을 정렬시키고 시스템을 더 쉽게 이해할 수 있다. 이는 단지 몇 가지 단순한 부분들만 이해하면 되므로 가능하다. OOD(Object Oriented Design)이라 불리는 기술은 어플리케이션 계층 및 함수 블록 어플리케이션 프로세스의 설계에 사용되었다.

OOD에는 많은 "키워드"가 존재하지만 Fieldbus의 경우에는 객체 및 클래스로 충분하다. 객체는 명확한 작동 기능을 갖는 개체이기에 시스템은 시스템의 구성 요소로 생각할 수 있는 객체들로 분류된다. OOD에서 소프트웨어는 메시지를 받거나 작업 지시를 받았을 때 작동하거나 변경되는 객체에 기반한다. 따라서, OOD는 알고리즘(수행 단계)에 기반하지는 않는다. 객체는 종종 실세계의 개체(예, 파일)를 의미한다. 객체는 기능 및 기타 공통적 특성에 따라 분류할 수 있다. 클래스는 다양한 "종류의" 객체를 정의하며 객체의 독특한 특성은 클래스의 *인스턴스*를 정의한다.

추출 요소의 순서를 지정하고 우선권을 부여하기 위해 객체 및 클래스는 모두 계층(추출 또는 복잡성의 수준)으로 구성된다. 어떤 수준은 다른 수준 위에 구축되기 때문에 각 수준을 그 자체로 이해할 수 있다. 상속(inheritance)은 클래스 계층이기에 (하위 클래스는 상위 클래스의 구조와 동작을 공유한다) 다중 상속도 가능하며 많이 사용된다. 집계는 객체 계층이며 객체는 하위 객체로부터 구축된다.

예: 대규모 제어 시스템에는 관리 시스템, 감독 시스템 및 필드 장비 등 제어 시스템의 모든 부분이 포함된다. 필드 장치에는 센서, 전자 장치 및 외장 등의 하위 부분이 포함될 수 있다. 어떤 시스템에는 LD302가 있을 수 있으며 이 LD302는 압력 전송기, 전송기, 필드 장치일 수 있다. LD302는 압력 전송기 클래스의 특성과 동작을 상속한다. 따라서 압력 전송기에 익숙한 오퍼레이터는 수분 안에 LD302를 작동시킬 수 있기 때문에 LD302의 독특한 특성 이외에는 알 필요가 없다.

OOD는 공통 메커니즘, 집계 및 상속을 재사용하여 시스템 규모를 축소시킨다.

모델을 사용하면 추출, 분해 및 계층 정렬 작업을 쉽게 수행할 수 있기 때문에 Fieldbus 및 모든 엔지니어링에서 광범위하게 사용된다.

Fieldbus의 OOD

Fieldbus 제어 시스템은 적합한 추출 수준인 이른바 *단순 변수*로 분해되며 부동 소수점, 정수 및 문자열이 그 사례이다. 단순 변수는 그 자체로 사용되지만 또한 함수 블록 I/O 파라미터 및 함수 블록 연결과 같은 *데이터 구조*의 일부로도 사용된다. 이러한 데이터 구조는 또한 데이터 구조이기도 한 함수 블록 데이터 유형의 일부이다.

함수 블록은 함수 블록 어플리케이션 프로세스의 일부이며 이는 다시 필드 장치의 일부이고 이것은 마지막으로 시스템의 일부가 된다.

변수는 다음과 같이 다양한 방식으로 분류할 수 있다.

- 부동 소수점, 정수 및 문자열
- 정적 또는 동적
- 읽기 또는 쓰기
- 기타

변수는 Fieldbus에서 정의되는 많은 유형의 객체 중 하나이지만 가장 중요하며 AP에 존재하기 때문에 *APO(Application Process Object)*라 부른다.

Fieldbus 어플리케이션 계층

시스템의 분산 어플리케이션 프로세스에는 통신이 필요합니다. Fieldbus는 어플리케이션 프로세스 간의 논리적 통신 경로(채널)로서의 역할을 수행한다. 다양한 통신 요구를 만족시키기 위해 다양한 연결 형태를 조합 사용하여 다양한 특성을 얻을 수 있다. 동시에 여러 개를 연결할 수 있으므로 다중 변수 액세스가 가능하다.

Fieldbus 연결은 두 가지 방식으로 모델화된다.

- 클라이언트-서버 모델
- publisher-subcriber 모델

클라이언트-서버 모델은 비주기적 데이터 전송을 설명하는데 사용한다. 통신 측면에서 *클라이언트*는 원격 AP의 기능을 사용하는 AP이고 이 원격 AP를 서버라 부른다. 예를 들어, 오퍼레이터 콘솔에서 필드에 있는 컨트롤러의 튜닝 파라미터를 읽으려는 경우, 콘솔의 AP는 클라이언트이며 컨트롤러의 AP는 서버이다.

publisher-subcriber 모델은 주기적 데이터 전송을 설명하는데 사용한다. 통신 측면에서 *가입자*는 원격 AP의 기능을 사용하는 AP이다. 이 원격AP를 publisher 라 부른다. publisher는 실제로 데이터를 생산(발행)하며 가입자는 그 데이터를 소비(가입)한다. 예를 들어, 전송기는 컨트롤러가 소비하는 공정 변수를 발행한다. 컨트롤러는 엑츄에이터가 소비하는 출력을 publishing 한다. 전송은 publisher에게 데이터 발행을 요청하는 제 3자인 *요청자(requestor)*가 제어한다.
참고: publisher-subcriber 모델은 더 일반적인 생산자-소비자 모델에서 유래하고 있다.

OOD를 설명하면서 언급하였듯이 Fieldbus 시스템은 변수로 분류된다. 일련의 서비스가 존재하여 어떤 AP가 또 다른 장치의 AP 기능, 예를 들어 변수 값을 얻는 등의 기능을 사용하도록 하거나 객체를 조작할 수 있도록 한다.

Fieldbus의 일차적 목적은 함수 블록을 사용하여 어플리케이션을 구축하는 것이며 이는 FBAP(Function Block Application Process)에서 수행된다. 그러나 Fieldbus 장치 내에서는 다른 유형의 AP 즉, 아직 그 정의가 마련되지 않았지만 사다리 논리 또는 구조적 텍스트를 사용할 수 있다.

Fieldbus의 관점에서 보면 장치는 하드웨어의 일부가 아니다. 예를 들어, 압력 전송기는 압력 센서, 전자 장치 및 외장을 조립한 것이 아니라 파라미터를 포함하는 네트워크 노드인 것이다. 이러한 네트워크 관점을 VFD(Virtual Field Device)라 부른다. 장치(스테이션)는 단지 하나의 FBAP만 포함한다. FBAP는 장치의 어플리케이션을 개별 루프로 세분하여 오퍼레이터가 쉽게 검토할 수 있도록 몇 가지의 VFD를 포함한다.

VFD는 프로토콜 스택 및 함수 블록 AP 간의 인터페이스로써 변수 및 블록 등과 같은 통신 객체 즉, 네트워크를 통해 볼 수 있고 액세스할 수 있는 실제 어플리케이션의 일부이다.

장치가 또 다른 장치의 통신 객체(예, 변수)에 액세스하려면 우선 사용 가능한 객체와 그 구조를 알아야 하는데 응답의 해석 방법을 모르면 부동 소수점 또는 정수의 경우 변수를 요청할 수 없기 때문에 구조의 파악이 중요하다. 이러한 정보는 사전 구성되거나 통신 동반자로부터 얻을 수 있다. 여기에는 두 가지 종류의 서비스 즉, 객체를 조작하기 위한 작동 서비스 및 그 설명적 속성을 조작하기 위한 서비스가 있다.

모든 객체(변수 등)에는 쉽게 참조할 수 있도록 인덱스가 존재한다. 시스템의 각 파라미터는 연결 및 인덱스를 통해 고유하게 구별된다. 이러한 방법은 Fieldbus 시스템의 설치 및 가동을 시작하고 데이터를 요청할 때 사용한다. 인덱스 및 주소 추적은 네트워크가 수행하며 사용자 인터페이스의 종류에 따라 사용자에게 완전히 투명하기 때문에 사용자는 이에 대해 걱정할 필요가 없다.

휴대용 터미널과 같은 MMI(man machine interface)는 통신 목적 이상으로 객체에 대한 정보를 필요로 한다. 예를 들어, 변수 작성 전에 연관된 특정 프로시저가 존재하는 경우, 정보를 동적으로 업데이트할 때 이를 사용자에게 표시(예, 메뉴로)하는 방법이 있어야 한다. 이 정보는 예를 들어, 장치에 저장하거나 자기 매체에 별도로 공급할 수 있다.

서비스의 예

- 연결 해제
- 장치 상태 읽기
- 장치 제조 업체, 형태 및 버전 읽기
- 구성의 전체 또는 일부 읽기
- 변수 읽기
- 변수 쓰기
- 이벤트 통지
- 블록 생성
- 파라미터에 대한 OD 인덱스 검색
- 블록 삭제

시스템 및 네트워크 관리

네트워크 관리의 목적은 Fieldbus 시스템의 유지 관리 및 시작과 같이 네트워크 프로토콜 스택을 중앙에서 구성하고 제어하기 위한 서

비스를 제공하는 것이다. 예를들어 네트워크 관리는 연결(connection)을 관리하는 것이다. 시스템 관리는 제어 어플리케이션이 기반할 수 있는 기본적 기능을 제공하는 커널 그리고 작동을 최적화하고 문제 진단을 제공하는 부분으로 구분된다. 이는 모든 계층에서 기능을 조정하고 전체 장치의 작동과 시작을 제어한다.

시스템 관리 커널의 목적은 다음에 대한 기능을 제공하는 것이다.

- 장치 태그 할당
- 스테이션 주소 할당
- 클럭 동기화
- 분산 AP 예약
- 함수 블록 결합

어떤 시스템에서 위의 각 기능은 한 장치만이 관리할 수 있으며(한 장치가 여러 기능의 처리 능력이 있지만), 기타 기능은 에이전트 역할을 수행한다. 관리 장애가 있는 경우, 에이전트 중 하나가 관리자 역할을 담당하고 시스템 관리를 위해서는 네트워크 상의 다른 스테이션에 있는 시스템 관리와 협동하여야 한다. 단순 장치는 시스템 관리 기능의 단지 일부만 수행한다.

물리 장치 태그 할당

장치를 네트워크에 연결하기 전에 사용자는 우선 장치에 물리 장치 태그를 할당해야 한다(즉 오프라인으로 수행). 태그는 예를 들어 PT-10270과 같은 표준 사용 관례를 준수하여 최대 10 문자로 이루어진다.

스테이션 주소 할당

네트워크의 각 장치가 고유 주소를 갖도록 자동 할당한다. 구성 장치 태그가 이중으로 존재하는지 여부를 점검한 후 새 장치를 탐지하고 스테이션 주소를 할당하여 장치를 "준비 완료" 상태로 만든다. 휴대용 구성기와 같은 임시 장치는 네트워크에 소통량이 없는 경우 자신의 주소를 스스로 선택한다.

함수 블록 결합

네트워크는 주어진 함수 블록에 대한 장치(스테이션 주소)를 자동을 검색한 다음 태그가 여러 개 존재하는지 점검한다. 이 기능은 블록 입/출력 간의 연결(블록의 태그 및 파라미터 이름으로 구별)을 짧은 주소 및 인덱스 참조로 바꿀 때 사용한다.

클럭 동기화(Clock synchronization)

Fieldbus 시스템이 경보 및 이벤트의 타임 스탬프와 같은 예약 및 기타 시간 관련 기능을 수행하도록 각 장치에는 분산된 시간 베이스(클럭)가 존재하여 모든 장치 간에 공통 시간 표준 즉, "시스템 시간"을 제공한다. 시스템 관리는 “마스터 클록”을 통해 올바른 시간을 제공함으로써 각 장치의 시간 동기를 위한 메커니즘을 제공한다.

예약(Scheduling)

예약의 목적은 통신 중 발생하는 지연을 최소화하는 것이다. 지연시간이 발생하면 이는 곧 제어를 어렵게 만드는 순전히 비생산적인 시간이 되기에 예약을 통하여 지연이 일정하도

록 함수 블록을 정확히 주기적으로 실행하고 변수를 표본 추출할 수 있다. 지연에 변화가 생기는 경우 튜닝을 다시 해야 하기 때문에 반드시 지연을 일정하게 유지해야 한다. 이런 식으로 배경 소통량을 위한 충분한 시간을 확보한 상태에서 폐쇄 루프를 기밀하게 제어할 수 있다.

예약에는 다음 3 가지 기능이 있습니다.

- Background Traffic
- Operational Traffic
- 함수 블록 실행(Function Block Excution)

동기화는 시스템 시간에 바탕을 두고있다. 정수 개의 단계(*클록 단위*)로 세분되는 블록 수행 주기가 *거시 주기(macro cycle)*를 이룬다. 거시 주기 동안 모든 함수 블록이 실행되고 모든 작동 소통량이 전달된다(그림 2.7 참고). 함수 블록 실행(Function Block Execution)은 적합한 순간에 FBAP에 알려줌으로써 단계의 초기에 함수 블록의 실행을 시작한다. 변환기 블록은 견본이 사용 가능한 경우 특수 감지 기술 또는 측정을 가능하게 하도록 예약되지 않는 다는 것을 참고바란다. 사용자는 파라미터 전파로 인한 지연을 최소화하기 위해 블록의 실행 순서를 결정할 수 있다. 배경 소통량은 함수 블록 실행 또는 작동 소통량에 사용되지 않는 단계 중 전달되도록 예약된다.

그림 2.7의 예에서, 단순 제어 루프는 입력(A1), 제어(PID) 및 출력(AO)으로 이루어지며 A1 및 PID는 동일 물리 장치에 존재한다. 공정 변수에 대해서는 장치 간 연결이 필요하지 않다. 본 예에서 AO 블록에 대한 실행 주기는 블록 실행 시간이 블록에 의존한다는 것을 나타내기 위해 더 짧게 표시하였다.

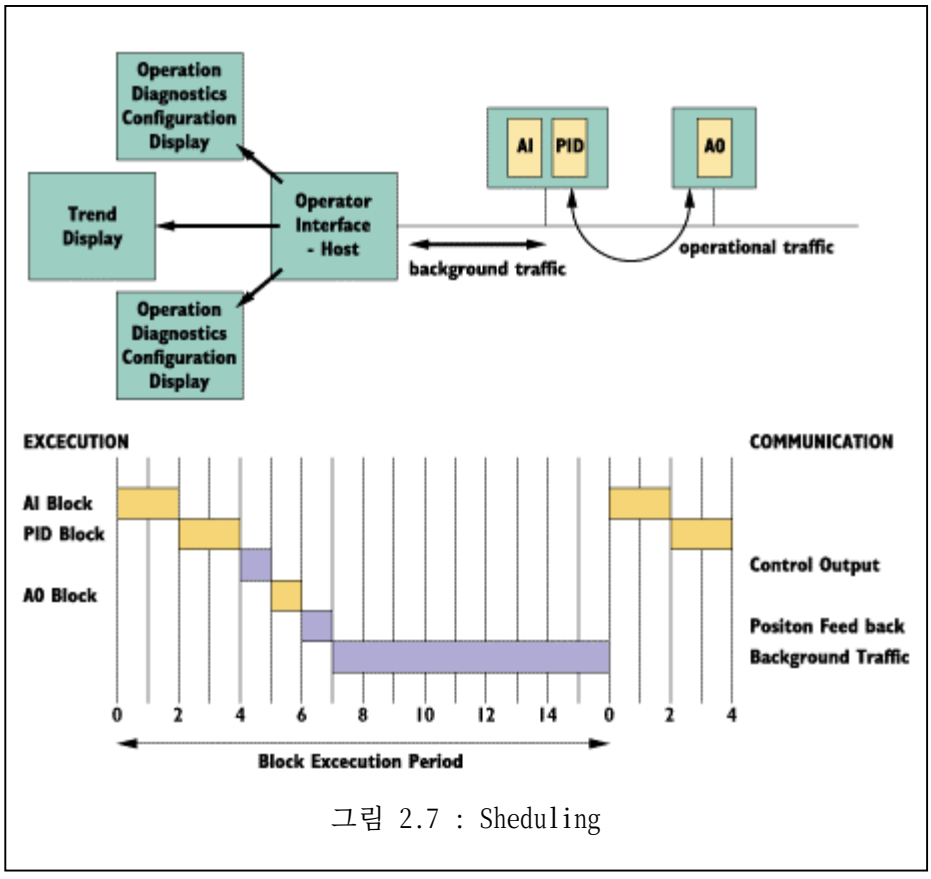

그림 2.7 : Sheduling

통신은 소통량을 제어하고 통신을 요청하는 마스터 장치에서 예약되며 함수 블록은 개별 장치에서 예약된다.

식 2.1 설명 : 실행 주기/시간, 여기서 x는 -128 ~ 127의 단일 정수. 예) x = -3의 결과는 125 ms 이다.

$$Time = 2^{x} \text{ seconds}$$

Equation 2.1

Execution period/time, where x is a signed integer: -128 to 127.
For example. x = -3 results in 125 ms

식 2-1

는다.

Fieldbus 장치의 기능은 객체로 모델화된다. 블록 객체는 3 가지 클래스로 이루어지며 이는 다시 하위 클래스로 이루어지고 하위 클래스 하에서 다양한 블록이 그룹화된다.

- 블록 객체(Block Object)

함수 블록 객체

인력 함수 블록

함수 블록 애플리케이션 프로세스(FBAP)

FBAP(Function Block AP)는 사용자가 측정 및 제어 어플리케이션을 구성하는 위치이며 그 일부가 필드의 다양한 장치에 분산된다. 이는 DCS에서처럼 단일 제어 카드에서 실행되지 않

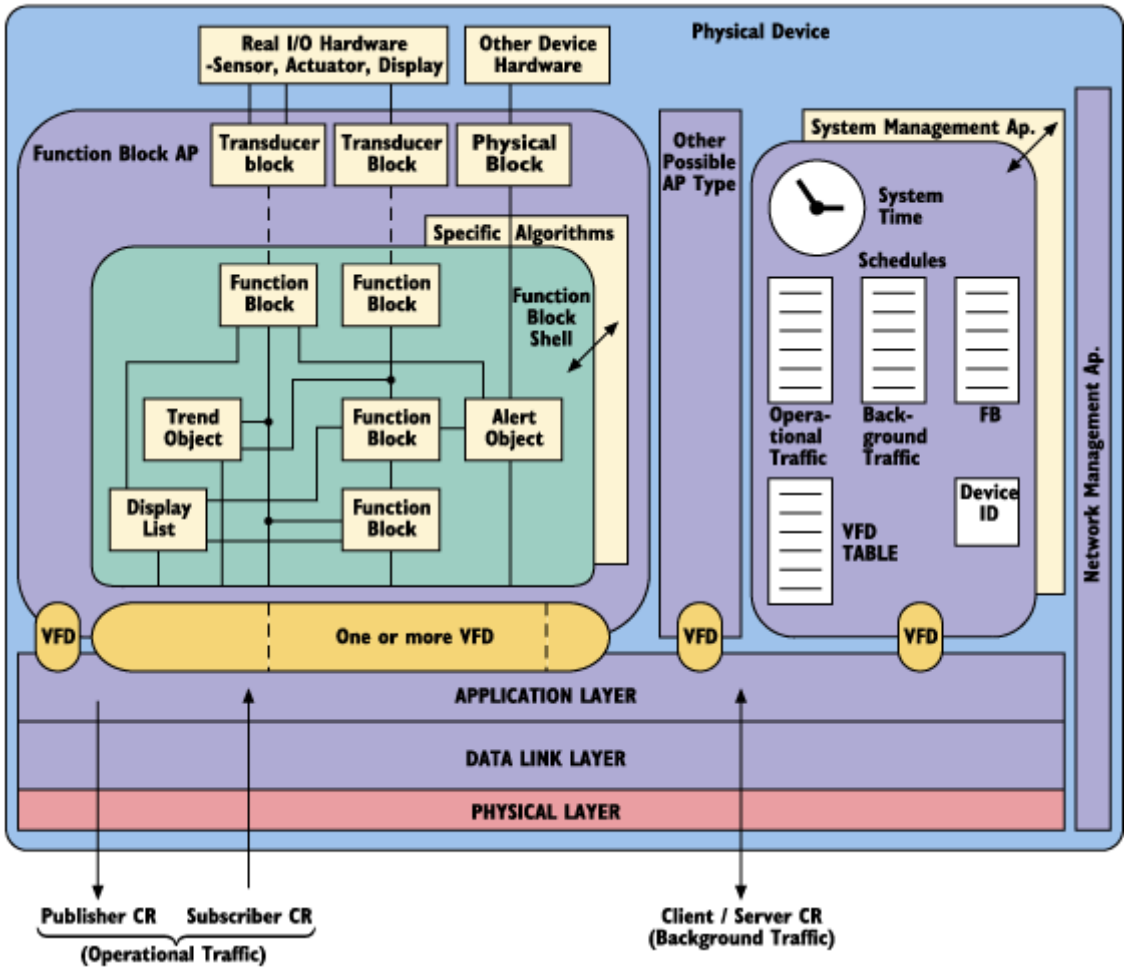

그림 2.8 : FBAP 구조

출력 함수 블록
제어 함수 블록
함수 블록 계산
변환기 블록 객체
입력 변환기 블록
출력 변환기 블록
디스플레이 변환기 블록
물리 블록 객체
- 경보 객체
- 이벤트 객체
- 경향 객체
- 디스플레이 목록

Fieldbus로 표준화되는 FPAP 부분을 *함수 블록 셸(function block shell)*이라 부른다. 표준화되지 않는 부분의 예로 블록 알고리즘이 있는데 각 블록에 대해 일단의 파라미터가 존재하여 특정 범위까지 어떤 블록이 갖을 최소 기능이 정의되지만 제조 업체는 독자적인 방식으로 이러한 블록을 구현할 수 있다. 예를 들어, PID 제어 블록에는 GAIN 파라미터가 있어야 하며 따라서 제조 업체는 이 파라미터를 GAIN 또는 비례 대역으로 사용할 수 있다.

함수 블록

블록은 전체 어플리케이션 중 사용자가 구성할 수 있는 부분을 모델화한다. 이전에는 보통 이러한 기능을 개별 물리 장치에서 사용할 수 있었으나 현재는 몇 가지가 소프트웨어 블록의 형태로 단일 장치에 포함된다. Fieldbus 시스템의 이러한 모든 상이한 함수 블록 형태는 대부분의 제어 시스템에 필요한 모든 기능을 제공한다. 사용자는 이러한 함수 블록을 연결하여 해당 어플리케이션에 적합한 제어 방안을 구축할 수 있다.

일반적으로 함수 블록은 알고리즘과 포함된 파라미터를 사용하여 입력 파라미터 및 그 결과로 발생하는 출력 파라미터를 처리한다고 볼 수 있다. 다시 한번 언급하지만 블록은 소프트웨어 및 데이터를 추출한 것에 불과하다. 장치 내부에는 블록을 찾을 수 없기 때문이다. 함수 블록의 개념은 PID 블록을 중심으로 설계되었는데 그 이유는 이 블록이 가장 복잡한 블록이기 때문이다. 로컬/원격 세트포인트, 자동/수동 출력, Cascaed(원격 세트포인트) 및 알고리즘의 개념이 다른 블록으로 전달되는데 처음에는 이것이 이상하게 생각되었을 것이다. 특정하게 선택한 세트포인트 및 출력을 블록 모드라 부른다. 알고리즘은 PID 블록의 PID 알고리즘만을 이르는 말이 아니라 일반적으로 모든 블록의 처리 함수를 지칭한다.

각 블록은 사용자가 할당한 태그를 통해 시스템에서 식별할 수 있기 때문에 이 태그는 Fieldbus 시스템에서 고유해야 한다. 블록의 각 파라미터는 이름을 가지고 있으며 변경할 수 없다. 시스템의 모든 파라미터는 블록 태그와 파라미터 이름으로 고유하게 정의된다.

다른 블록의 입력은 비동기적으로 도달되며 블록이 실행되면 따라서 입력의 "Snapshot"을 얻는다. 이는 또한 블록 실행 중 입력 데이터가 변경되는 것을 방지한다. 블록 알고리즘 실행 후 그 출력은 네트워크에서 업데이트 및 Broadcast되며 이 정보를 이용하여 블록의 입력 준비가 완료된다 이러한 식으로, 출력이 많은 입력에 연결되어 있는 경우에도 오직 한 차례만 통신을 수행하여야 한다 블록에 대한 실행 시간도 식 2.1의 실행 주기와 마찬가지로 표현된다.

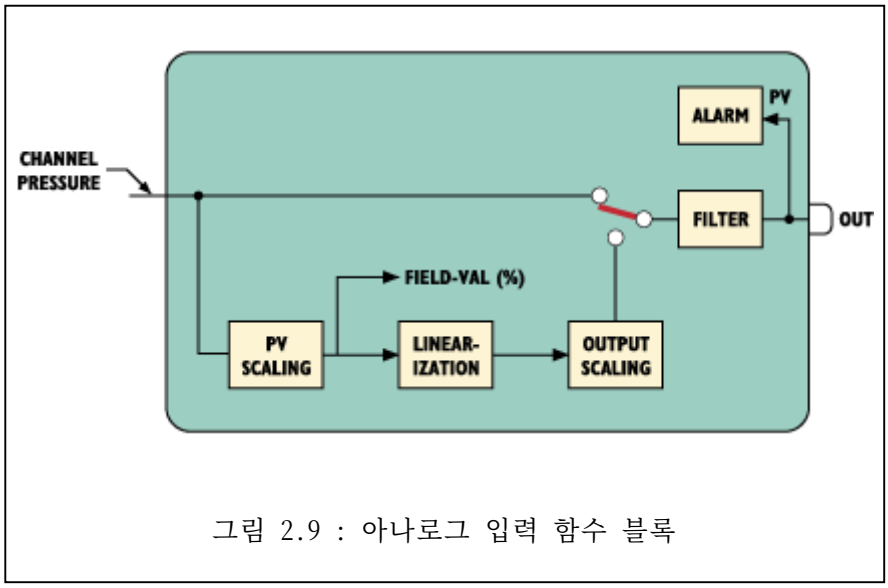

그림 2.9 : 아나로그 입력 함수 블록

또는 수준에 대해 필드에 분산되어 있는 전송기 또는 밸브의 컨트롤러로 작동할 수 있다. 이는 또한 공정 변수 및 이탈 경보, UR 및 A/M 스테이션, 오버라이드 및 전방향 피드(feed-forward)를 제공한다.

아날로그 출력 블록밸브와 같은 액츄에이터의 기능을 제공하며 계산된 제어 출력을 액츄에이터 하드웨어가 사용할 수 있도록 한. 이는 또한 신호 역전환 및 실제 위치에 대한 피드백을 옵션으로 적용한다.

구성은 기본적으로 태그를 할당하는 작업이며, 블록을 선택하여 제어 방안을 구축(설치)하고 이를 연결하며 포함 요소를 조절하는 것은 단순한 휴대용 구성기를 사용하거나 그래픽 사용자 인터페이스 기능을 제공하는 컴퓨터를 사용하여 수행할 수 있다. 이를 통하여 사용자는 구성을 제어 다이어그램으로 도식화할 수 있다. 구성 작업은 미리 또는 작동 중 수행할 수 있다.

아날로그 입력 블록

전송기의 기능을 제공하며 장치가 수행한 측정 결과를 Fieldbus 시스템이 사용할 수 있도록 한다 측정한 차변 압력의 제곱근과 같은 교정, 감쇄(damping) 및 전송 기능도 옵션으로 적용하여 차변 압력 전송기의 흐름과 같은 적외선 측정을 가능하게 한. 이는 또한 공정 변수 경보를 제공한.

PID 제어 블록

PID 컨트롤러의 기능을 제공한. 이를 통하여 장치는 통상적으로 자신이 측정하거나 액츄에이터 역할을 하는 압력, 흐름

함수 블록 연결

함수 블록 출력을 다른 함수 블록의 입력에 연결함으로써 제어 방안을 구축할 수 있다. 이러한 연결을 구성하는 경우 입력이 출력에서 값을 "추출"함으로써 그 값을 얻을 수 있다. 연결은 동일 장치 또는 상이한 장치의 함수 블록 간에 구성할 수 있다(그림 2.10 참고). 출력은 많은 입력에 연결할 수 있다. 이러한 연결은 순전히 소프트웨어 방식이며 물리적인 와이어를 통한 연결 길이에는 기본적으로 제한이 없다. 포함 변수로는 연결을 구성할 수 없다. 아날로그 값은 공학 단위의 부동 소수점으로 전달되지만 백분율로 스케일을 조정하여(예, PID 제어 블록에서) 무차원 튜닝 파라미터를 얻을 수 있다. 디지털 값은 불리언(Boolean), 0 또는 255로 전달됩니다. 아날로그 스케일 조정 정보는 오퍼레이터 인터페이스에서 사용하여 막대 그래프를 생성할

수도 있다.

출력 값은 예를 들어 센서로부터 받은(순방향 경로) 값이 제어하기에 적합한지 알려주는 상태를 항상 동반하며 출력이 최종 제어 요소를 이동시키는지 여부를 판단할 수 있는 피드백으로 역할을 수행한다. 이러한 상태는 소스로부터 판단한다. 추출 시스템은 역방향 경로에도 마찬가지로 사용된다는 것을 참고하라. 이러한 방식을 통해 수신 함수 블록이 해당 동작을 취할 수 있다.

연결은 출력 파라미터의 이름과 그 출원인 함수 블록의 태그에 의해 고유하게 정의되기 때문에 사용자가 아주 쉽게 식별할 수 있다. 시스템 관리는 블록 태그+파라미터 이름 구성을 짧은 참조 주소+인덱스로 변환하여 더욱 빠른 통신을 가능하게 한다. 주소 및 인덱스를 사용하여 연결을 사전에 직접 구성할 수도 있다. 연결은 전원 공급 시 연결관리(Link Management)가 자동으로 구성한다.

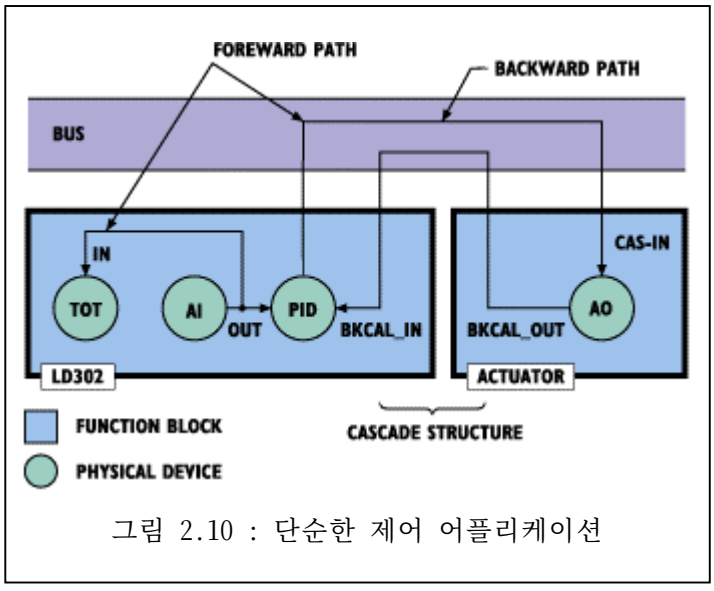

그림 2.10 : 단순한 제어 어플리케이션

모든 함수 블록 링크는 publisher-subcriber 연결을 구현한다. 함수 블록은 통신이 단절된 경우(예, publishing manager failure -발행 관리자 장애) 동작을 취하도록 정의된다.

변환기 블록

변환기 블록은 함수 블록 및 장치 I/O 하드웨어 간의 인터페이스를 담당한다. 센서, I/O 터미널 또는 디스플레이와 같은 각 하드웨어 지점에 하나의 변환기 블록이 존재한다. 표준 자체는 변환기 블록에 대해 어떤 파라미터도 지정하지 않지만 사용자 그룹은 다양한 장치 유형에 대한 파라미터를 식별한다. 또한 변환기 블록은 예약되지 않기 때문에 제조 업체는 변환기의 실행을 제어하여 예를 들면 함수 블록 실행보다 빠른 자체 감지 기술을 실현할 수 있다. 변환기 블록은 함수 블록(입력 도는 출력 클래스)에 대한 인터페이스를 장치에 상관 없이 구성한다.

변환기 블록은 그 파라미터 모두가 내부에 포함되기 때문에 함수 블록 링크를 사용하여 링크할 수 없다. 변환기 블록은 열거된 하드웨어 채널을 통해 그리고 동일 물리 장치의 블록을 통해서만 함수 블록에 인터페이스하며 해당 함수 블록이 하드웨어 채널을 지정한다.

입력 변환기 블록:
온도 특성화 및 트림과 같은 센서 신호의 처리를 담당한다.
출력 변환기 블록: 트림과 같은 액츄에이터 및 피드백 신호의 처리를 담당한다.
디스플레이 변환기 블록: 로컬 디스플레이 및 키보드 등을 담당한다.

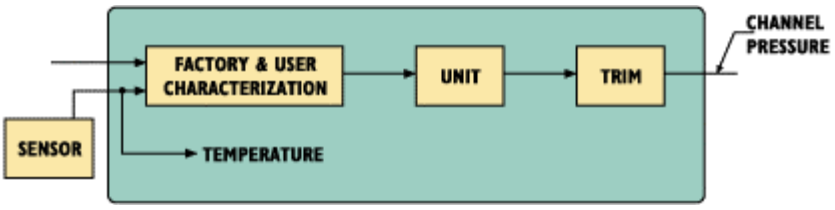

그림 2.11 : 아날로그 입력 변환기 블록(압력)

물리 블록(Physical Block)

장치에는 하나의 물리 블록만이 존재하여 자가 진단과 같이 전체 장치의 작동을 감시하는 역할을 한다. 여기에는 또한 최종 조립 번호 및 재료와 같은 장치 정보가 포함된다. 물리 블록은 함수 블록 링크를 사용하여 연결할 수 없으며 모든 파라미터는 내부에 유지된다. 물리 블록에는 또한 장치의 모든 블록이 사용할 수 있는 "전역" 파라미터가 포함될 수 있다. 예를 들어, 여러 아날로그 입력 블록이 하나의 선형화 데이터 집합을 사용할 수 있다.

함수 블록은 예약되지 않기 때문에 제조 업체는 실행을 제어하여 장치 필요성을 만족시킬 수 있다.

블록 파라미터

OOD 부분에서 언급하였듯이, 구조는 단수 변수로 분해되며 변수에 대해 두 가지 매체 유형이 존재한다.

- 단순 변수(Simple variable)
- 레코드(데이터 구조)

각 메타 유형에 대해 많은 하위 유형이 존재한다. 예를 들어, 부동 소수점, 정수 또는 문자열 등이 단순 변수가 될 수 있다. 데이터 구조는 관련된 단순 변수 파라미터를 논리적으로 그룹화한 것이다. 예를 들어, 모드 데이터 구조는 대상, 실제 및 표준 모드를 모아놓은 것이며 경보 데이터 구조는 상태/우선권, 원인, 원인 값 및 시간 스탬프를 모아놓은 것이다. 블록도 일종의 데이터 구조이다. 단순 변수는 자체적으로 사용하거나 데이터 구조의 요소로 사용할 수 있다. 데이터 구조의 장점은 각 요소에 대해 반복적으로 요청하지 않고 한번의 요청으로 필요한 파라미터에 한꺼번에 액세스할 수 있다는 것이다.

블록에서는 많은 파라미터 분류를 사용하며 어떤 파라미터는 여러 분류에 속할 수 있다.

- 사용(Use)
- 저장(Storage)
- 계층(Hierarchy)
- 액세스(Access)

블록에 파라미터의 사용에 따라 다음 3 가지 분류가 존재한다.

- 입력 -- 함수 블록의 출력 값을 얻기 위해 여기에 연결할 수 있으며 PID 블록의 공정 변수가 그 예이다. 연결하는 경우 오퍼레이터가 입력을 설정할 수 있다. 입력 파라미터는 수치 및 상태로 이루어진다.
- 출력 -- 값을 전파하기 위해 함수 블록 입력에 연결할 수 있다. 이는 블록이 계산하거나 특정 모드에서 오퍼레이터가 설정할 수 있

으며 PID 블록의 조작 변수(출력)가 그 예이다. 출력 파라미터는 값과 상태로 이루어진다.
- 포함 요소 -- 연결할 수 없으나 구성, 작동 및 진단과 같은 사용자 인터페이스에 대해 사용할 수 있다. 포함 파라미터는 블록의 작동을 제어한다. 이는 블록 모드에 따라 블록이 계산하거나 오퍼레이터가 설정하며 PID 블록의 조정 파라미터가 그 예이다.

변환기 블록과 물리 블록에는 포함 파라미터만 존재한다.

운터가 증가한다. 이 값은 전원을 차단하여도 보존되며 경보 제한이 그 예이다.
- 비 휘발성(Non-volatile) -- 이 변수에 작성하면 정적 수정 카운터가 증가하지 않는다. 이 값은 전원을 차단하여도 보존되며 PID 블록의 세트포인트가 그 예이다.
- 동적 -- 이 값은 함수 블록이 계산하거나 블록 입력으로 수신된다. 사용자는 이 변수를 작성하지 않는다. 이 값은 정적 수정 카운터를 증가시키지 않으며 전원 차단 시 지워진다. 블록 입력 및 출력이 그 예이다.

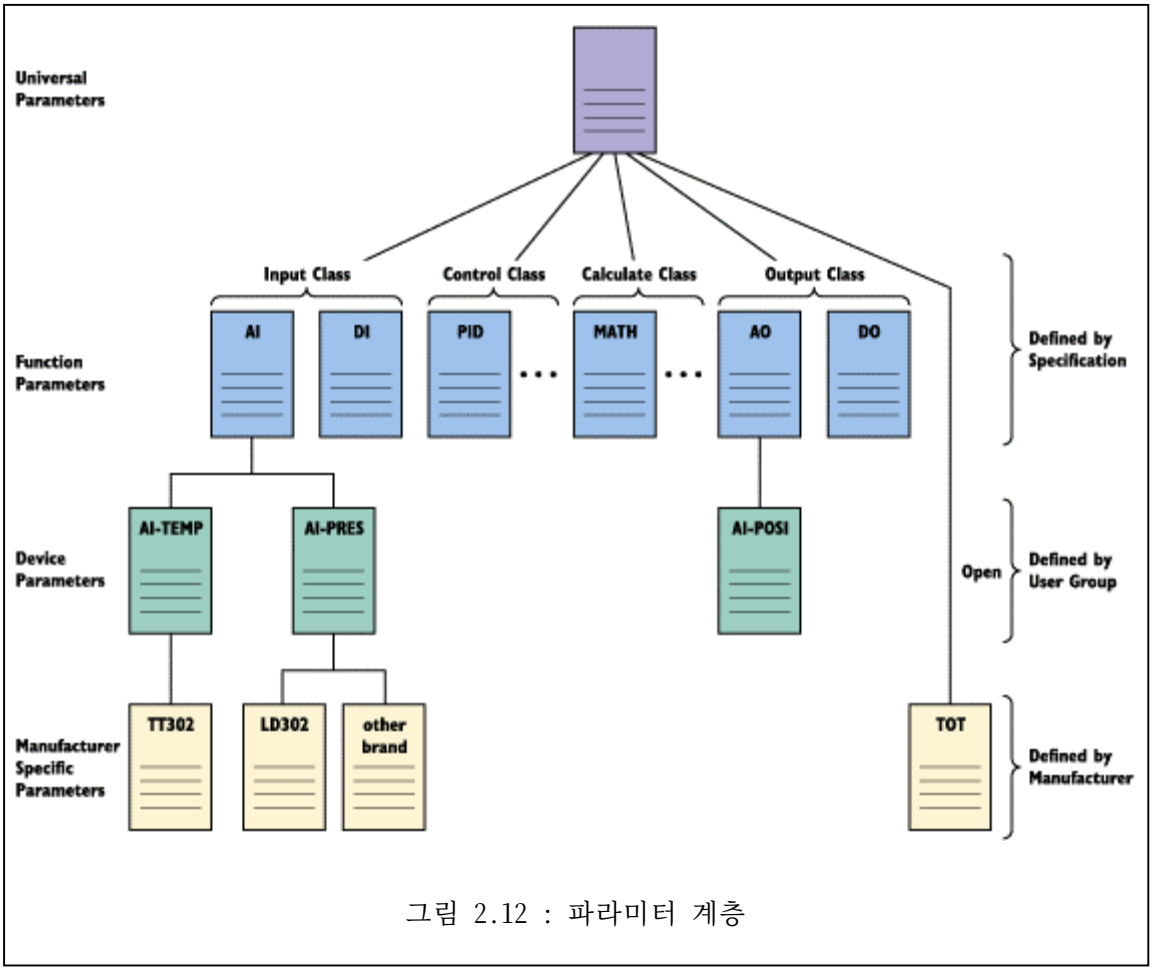

그림 2.12 : 파라미터 계층

파라미터의 저장 방식에 따라 다음 3 가지 분류가 존재한다.

- 정적 -- 이 변수에 작성하면 정적 수정 카

파라미터에 대한 액세스 방식에 따라 다음 2 가지 분류가 존재한다.

- 쓰기 -- 값을 읽거나 쓸 수 있다.

- 읽기 전용 -- 값을 읽을 수만 있으며 쓸 수 없다.

파라미터의 계층 수준에 따라 다음 4 가지 분류가 존재한다.

- 범용 -- 모든 함수 블록에 필수적이다. 예를 들어, 모든 블록에는 태그 및 모드 파라미터가 존재한다.
- 함수 표준에서의 정의에 따라 특정 함수 블록에서 필수적이다. 예를 들어, 아날로그 입력 블록에는 감쇄가 있어야 하며 PID 제어 블록에는 세트포인트가 있어야 한다.
- 장치 -- 사용자 그룹이 동의하는 특정 장치 유형에서 필수적이다. 예를 들어, 모든 압력 전송기에는 프로세스 연결에 대한 파라미터가 있어야 하며 모든 온도 전송기에는 센서 유형에 대한 파라미터가 있어야 한다.
- 제조 업체 -- 제조 업체의 정의에 따라 장치에 특정하거나 선택적이다. 제조 업체가 추가 파라미터를 도입할 수 있도록 하여 자사 장치의 고유 특징을 적용할 수 있으며 조절형 제곱근 자르기가 그 예이다.

파라미터의 상속은 상호 운용성에 중요한 역할을 한다. 상속 준수 장치는 장치 파라미터로 정의되는 파라미터 및 관련 기능을 최소 조건으로 포함한다.

경고 객체

다수의 함수 블록에는 경고 블록이 내장되어 있어 높고 낮은 공정 변수 및 이탈 경보를 탐지한다. 경보 및 기타 중요한 이벤트가 발생하는 경우, 경고 객체가 자동으로 사용자에게 이를 알려준다. 따라서, 오퍼레이터 인터페이스는 경보 조건의 존재 여부를 결정하기 위해 주기적 폴링을 수행할 필요가 없다. 물리 및 변환기 블록은 하드웨어 및 전체 작동 상태의 장애를 탐지한다. 경고 객체는 블록이 경고를 처리해야 하는 부담을 덜어줌으로써 실행에 영향을 미치지 않도록 한다. 또한 경고 객체가 오퍼레이터의 고지 수신 여부를 판단할 수 있도록 인지 메커니즘이 제공된다. 지정 시간 내에 응답이 수신되지 않는 경우, 경고 고지가 반복되며 경보 조건이 제거되는 경우에도 사용자에게 이를 고지한다.

이벤트의 예는 다음과 같다.

- 어떤 이유로 모드가 강제되는 경우
- 블록 태그가 변경되는 경우
- 잠긴 출력/안전 조건
- 피드백이 원하는 출력과 일치하지 않는다.

경보에 대해 사용자는 이동 수준 및 우선권 수준 및 불필요한 대역(dead-band)을 구성할 수 있다. 콘솔에 전달되는 경고 고지에는 시간 스탬프, 원인, 우선권, 현재 상태(경보가 이미 제거된 경우 포함) 및 이동 값이 포함된다.

구성이 변경되는 경우 우선권, 구성의 수정 수준, 변경된 파라미터 및 시간 스탬프를 포함하는 경고 고지가 발행된다.

모든 경고는 또한 경보를 발생시킨 장치 및 블록, 오퍼레이터에게 열거되는 메시지 식별용 유형 코드 및 공장 부서별 분류를 위한 경고 키를 알려준다. 메시지는 표준 메시지이거나 제조 업체가 기타 메시지를 지정할 수 있다.

각 블록에 대해 최대 16개의 경고에 대한 경보 요약이 존재하여 현재 상태, 경보의 수용 여부, 오퍼레이터의 보고 수신 여부 또는 실행 해제 상태인지 여부를 요약한다.

경향 객체

경향 분석은 장치가 경향 객체를 사용하여 자체적으로 수행할 수 있으며 이를 통해 시간에 민감한 주기적 통신의 필요성이 제거된다. 데이터는 20개의 표본으로부터 수집되며 단일 통신으로 액세스하므로 통신 및 네트워크 오버헤드를 감소시킬 수 있어 시간에 민감한 전송을 위해 더 많은 시간을 할당할 수 있다.

디스플레이 목록 객체

원격 오퍼레이터 인터페이스는 공정 변수 및 세트포인트와 같은 변수의 감시 및 액츄에이션을 제공한다. 이러한 인터페이스에는 구성 및 진단을 위한 액세스가 필요하다. 이러한 변수는 사용에 따라 4개의 그룹으로 묶을 수 있으며 여러 개의 개별 통신이 아니라 단일 통신으로 액세스하기 때문에 액세스의 빈도를 줄임으로써 네트워크 오버헤드를 감소시킨다. 모든 오퍼레이터 인터페이스 데이터는 배경 소통량으로 예약된다.

- 동적 작동 데이터 -- 예, 공정 변수
- 정적 작동 데이터 -- 예, 허용 모드
- 모든 동적 데이터 -- 모든 입력 및 출력
- 기타 정적 데이터 -- 예, 모든 경보 구성

액세스 권한

콘솔에 위치한 오퍼레이터는 블록에 있는 4개 집합의 파라미터에 대한 액세스를 로컬 인터페이스 또는 더 높은 수준의 장치에 허용 또는 거부할 수 있다. 콘솔에서 로컬로 또는 배치 프로그램으로 수행한 조절의 결과는 함수 블록에 동일하게 나타난다.

4개의 그룹은 다음과 같다.

높은 수준의 장치의 경우
- 프로그램(Program) -- 모드, 세트포인트 및 출력
- 튜닝(Tune) 튜닝 파라미터
- 경보(Alarm) -- 경보 파라미터

휴대용 터미널 또는 로컬 인터페이스의 경우
- 로컬 -- 모드, 세트포인트 및 출력

최적화

저속 Fieldbus 버전은 자체 안전 요건의 충족을 위해 처리 능력을 제한하려는 목적으로 선택되었다. 낮은 통신 대역폭에 의해 제한되는 기밀한 폐쇄 루프의 제어 필요성을 충족시키기 위해 위에서 이미 설명한 방식으로 네트워크 사용을 최적화하였으며 다음과 같다.

- 예약
- 짧은 참조
- 표준 파라미터 정의
- 구성 변경 경고
- 디스플레이 목록 객체
- 경보 객체
- 경향 객체

예약을 통하여 네트워크가 다른 파라미터의 도달 동안 대기하는 유휴 상태를 제거한다.

사용자 친화적이지만 길이가 긴 블록 태그 및 파라미터 이름은 주소 및 인덱스로 변환된다. 파라미터는 표준화되며 여러 방식으로 "디코딩"할 필요가 없다. 필드 장치가 변경되는 경우, 호스트는 변경이 발생한 위치에 대한 정보를 자동으로 고지 받음으로써 해당 파라미터만을 업데이트하면 되며 구성 변경 여부를 계속해서 점검할 필요가 없다. 경향 객체 및 디스플레이 목록 객체는 데이터 액세스에 필요한 통신 빈도를 줄임으로써 시간에 민감한 작업을 위해 더 많은 시간을 할애할 수 있다. 경고 객체는 경보 상태 변경 시 이를 자동으로 고지하여 호스트가 경보 상태에 대한 폴링을 자주 수행하여 네트워크에 로드를 가하지 않도록 한다.

3. Fieldbus 사용

일반 사용자에게 있어 함수 블록의 동작에 대한 이해가 가장 중요하다. 예를 들어 그림 2.10과 같은 제어 루프에는 통상적으로 적어도 A1 블록(전송기), PID 블록(컨트롤러) 및 AO 블록(밸브)이 포함되는데 이러한 3 가지 요소가 개별 장치에 존재하는 경우, 블록 간의 상호 운용성에 많은 조건이 발생하게 된다. 예를 들어, 오퍼레이터가 밸브의 제어를 인수하는 경우, 컨트롤러의 불안정한 상태를 방지하고 오퍼레이터가 밸브의 제어를 반환할 때 부드러운 전송이 가능하도록 컨트롤러에 이를 알려주어야 한다. 마찬가지로, 컨트롤러는 전송기로부터의 측정이 불량하거나 존재하지 않는 경우 통합 작업을 중단해야 한다.

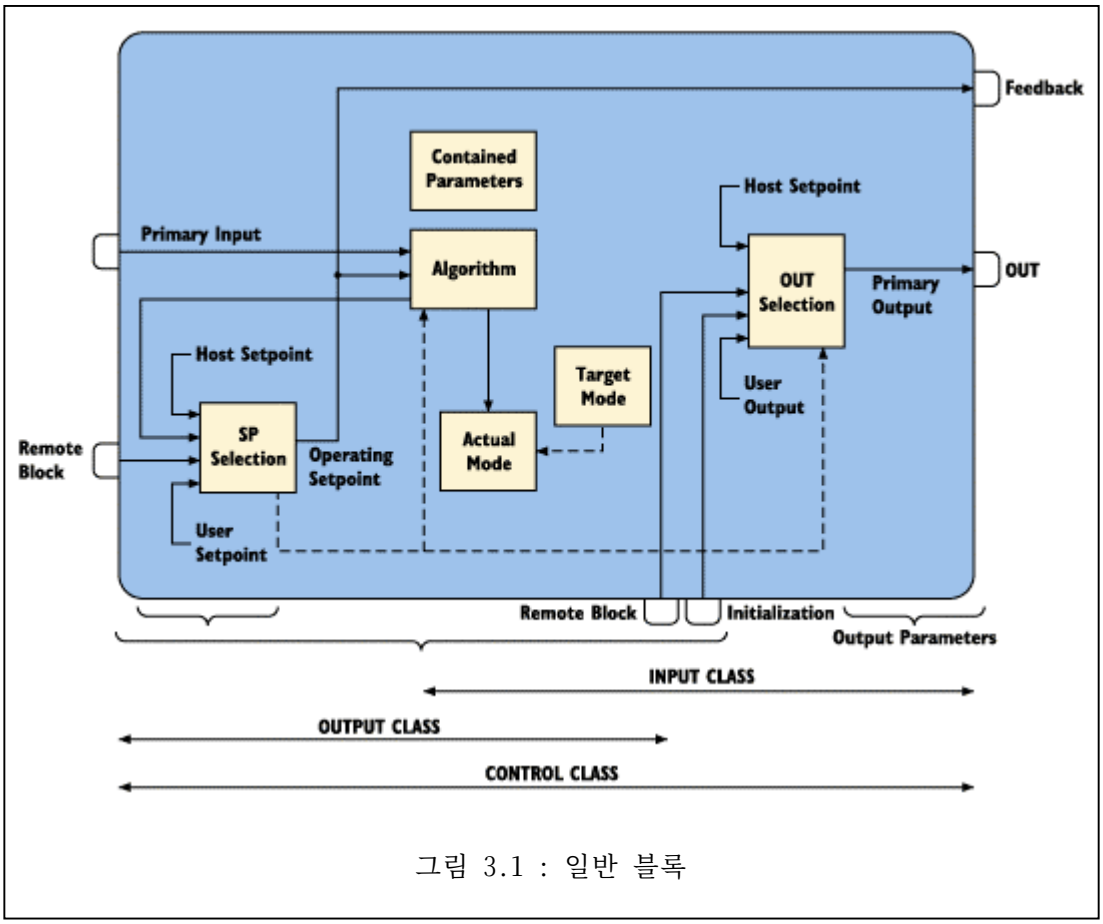

그림 3.1 : 일반 블록

모드

모드는 세트포인트 선택 및 출력 선택의 두 가지 함수를 갖으며 이는 전통적 제어 모드 즉, 각각 로컬/원격 및 자동/수동과 비교할 수 있다. "서비스 중단" 및 "로컬 오버라이드"(안전성)에 대한 모드도 존재한다. 장치에 구현된 모드의 수는 블록에 따라 다르다. 그림 3.1에 나타낸 일반적 블록에는 입력 블록에 세트포인트 선택이 없고 출력 블록에 출력 선택이 없는 것을 볼 수 있다.

운영 모드의 의미를 일반화하는 방법:

출력 선택:

자동: 블록 알고리즘은 입력을 사용하여 출력을 계산한다. 이는 세트포인트가 사용되는 출력 선택 그룹에서 유일한 모드이다.

수동: 오퍼레이터가 출력을 설정한다.

로컬 오버라이드: 출력은 다른 블록이 제공하는 추적 파라미터(예, 안전값)를 따른다. 이 모드는 특수 블록 입력으로만 설정된다.

초기화: 블록이 그 출력을 조정한다. 즉, 하향 블록이 제공하는 피드백 값을 따른다.

원격 출력: 호스트 컴퓨터가 블록 출력을 계산 및 제공한다.

세트포인트 선택:

로컬: 블록이 입력과 오퍼레이터의 세트포인트를 사용하여 출력을 계산한다.

케스케이드: 블록이 입력과 원격 블록의 세트포인트를 사용하여 출력을 계산한다.

원격 케스케이드: 호스트 컴퓨터가 세트포인트를 계산하고 제공한다. 출력은 블록 자체가 계산한다.

세트포인트 추적: 수동 출력 선택의 경에만 유효하다. 세트포인트는 공정 변수를 따른다.

블록에는 몇 가지 상이한 모드 파라미터가 존재하며 각각 특정 함수를 갖는다.

대상 모드: 오퍼레이터가 요청하는 모드. 허용 모드가 허락한 모드만 선택할 수 있다. 블록은 이 모드의 획득을 시도하며 다양한 블록 및 장치 상태에 따라 성공 여부가 결정된다.

실제 모드: 블록의 지배적 모드. 이는 현재 상태에 따라 대상 모드와 동일하거나 다를 수 있다. 사용자의 요청뿐만 아니라 기타 이벤트에 의해 변경될 수 있다.

허용: 오퍼레이터가 사용을 허용 할 수 있는 모드 정의. 이것은 작동 전에 공정 엔지니어가 구성한다.

블록이 구체화되고 작동이 시작되면 초기 모드의 서비스가 중단되며 작동 상태에서는 전원 공급이 중단되어도 마지막 모드가 보존된다. 블록이 실행되는 경우, 모드는 피드백 근원, 출력 및 다양한 세트포인트의 상태 비트 그리고 전체 장치의 상태로부터 결정된다.

상태 비트

마이크로프로세서 기반 장비는 자신의 하드웨어에서 에러를 탐지할 수 있다. 이 정보는 전달된 변수의 질을 알려주는데 사용하여 제어 블록의 완전한 중단을 방지하고 블록의 안전 모드 전환을 위한 메커니즘을 제공한다. 모든 함수 블록의 입력 및 출력에는 상태가 동반되며 다음에 그 예를 나타내었다.

- 질: 양호, 불안정, 불량 또는 서비스 중단은 통신 두절, 하드웨어 장애 등에 기인할 수 있다.
- 요청 초기화: 상향 블록을 초기화하여 부드러운 전송 제공
- 하향 블록 제한기를 통한 상한
- 하향 블록 제한기를 통한 하한

다른 블록으로부터 제어의 불연속을 나타내는 입력으로 받은 상태 정보를 통해 블록이 모드를 자동으로 전환한다. 이러한 동작을 "흘리기"라고 한다. 흘리기의 원인이 되는 상태가 제거되면 모드는 보통 이전 모드로 복귀한다. 사용자는 블록이 흘려지는 모드를 구성할 수 있으며 블록 출력 변수의 상태는 블록 모드에 따라 다르다. 따라서 한 블록에서 장애가 발생하면 모드 흘리기의 연쇄 효과가 발생하여 모든 블록을 적합한 모드로 유지시킨다.

케스케이드 구조

케스케이드 구조는 제어 방안을 구축할 때 중요한 개념이다. 더 넓은 의미로 케스케이드는 한 블록(통상적으로 PID 블록)의 출력이 또 다른 블록(반드시 PID 블록일 필요는 없으며 보통 출력 블록)의 케스케이드 세트포인트에 연결됨을 의미한다. 출력을 제공하는 블록을 신호 경로에서 "하향"이라 하고 케스케이드 세트포인트를 받는 블록은 "하향"이라 한다. 따라서 더 넓은 의미로 케스케이드 세트포인트는 상향 블록에서 출원하며 수신 블록은 그에 따라 출력을 조작한다.

피드백 경로도 케스케이드 구조에서 제공한다. 케스케이드 구조의 하향 블록이 상향 블록의 케스케이드 세트포인트를 언제나 수용할 수 있는 것은 아니다. 예를 들어, 케스케이드 세트포인트가 블럭의 모드로 인해 사용되지 않는 경우, 블럭에서 설정한 한계를 초과하는 경우, 블럭이 액츄에이터를 이동할 수 없는 경우 등이 해당된다. 상향 블럭이 출력을 더 이상 이동할 수 없다는 고지를 받지 못하는 경우, PID 컨트롤러의 통합 동작이 중단되거나 다른 블록의 경우, 실제 그렇지 않은데도 액츄에이션 작동이 실행중인 것으로 잘못 판단한다. 피드백 값의 상태를 이용하여 하향 블록이 상향 블록에 현재 상태를 알려줄 수 있다. PID 블록도 피드백 값을 사용하여 출력을 조정할 수 있다.

어플리케이션

지금까지 많은 함수 블록을 정의하였다. 이들을 결합하면 단일 아날로그 입력 블록만을 사용하여 가장 단순한 측정을 수행하고 단일 루프, 케스케이드, 할당, 교차 한계(cross limit) 등과 같은 전통적 제어 방안 또는 그림 3.2 ~ 3.4에 일부 나타낸 더욱 복잡한 구성을 구축할 수 있다.

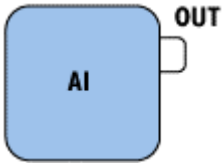

그림 3.2 : 측정

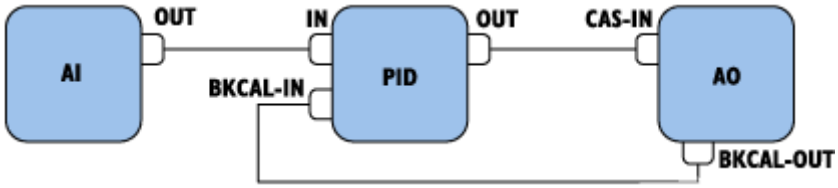

그림 3.3 : 단일 루프

해 사용할 선택 세트포인트의 복사본

RCAS_IN: 호스트로부터의 세트포인트에 대한 입력, 원격 케스케이드 모드에서 선택.

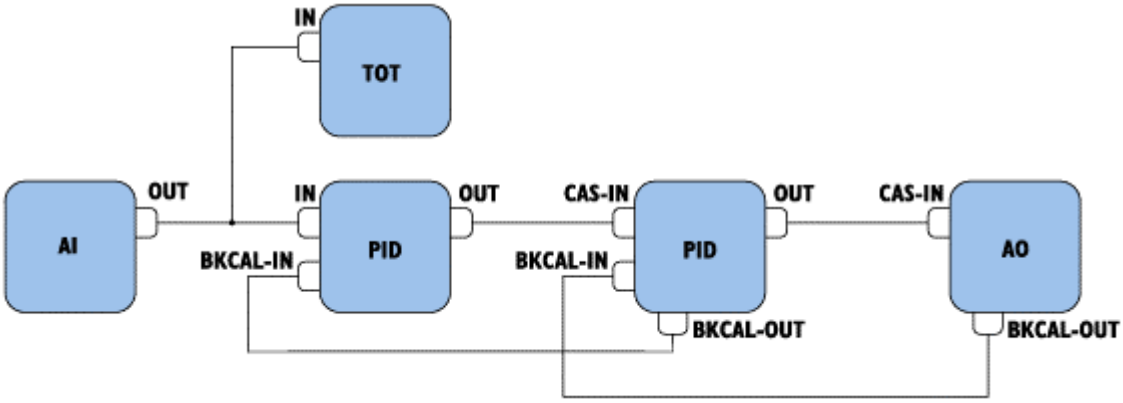

그림 3.4 : 케스케이드 제어

다음에는 가장 기본적인 입력 및 출력 파라미터를 설명한다.

IN: 공정 변수 입력

OUT: 블록의 1차 출력

CAS_IN: 다른 블록의 케스케이드 세트포인트로부터의 원격 세트포인트에 대한 입력

BKCAL_IN: 하향 블록으로부터의 피드백 입력. 로컬 모드로 복귀할 때 하향 블록에서 부드러운 세트포인트 전송을 보장하기 위해 블록 출력의 초기화, 조정에 사용하는 값

BKCAL_OUT: 상향 블록이 출력의 초기화를 위

RCAS_OUT: 선택 세트포인트의 복사본, BKCAL_OUT와 동일한 값이지만 상태는 RCAS_IN 통신에 기반.

ROUT_IN: 호스트로부터의 원격 출력에 대한 입력, 원격 출력 모드에서 선택.

ROUT_OUT: 선택한 출력, OUT의 복사본, 그러나 상태는 ROUT_IN 통신에 기반.

SP (포함): 로컬 모드에서 오퍼레이터가 설정한 세트포인트 또는 다른 블록이나 호스트에서 받은 세트포인트 제한 후의 세트포인트