KOROSKOR
OROSKOR
ROSKOR
OSKOR
SKOR
KOR
OR

KOROS

# 서비스 로봇을 위한 모듈 – X-1 부 : 모듈을 위한 공통 정보 모델

KOROS XXXX : YYYY

한국로봇산업협회

**YYYY년 MM월 DD일 제정**

**http://www.KOROS.or.kr**

KOROS XXXX : YYYY

**심 의 : 지능형로봇표준포럼 운영위원회**

| | 성 명 | 근 무 처 | 직 위 |
|---|---|---|---|
| (운영위원장) | 문 승 빈 | 세종대학교 | 교 수 |
| (운 영 위 원) | 이 순 걸 | 경희대학교 | 교 수 |
| | 우 종 운 | 한국로봇산업진흥원 | 센 터 장 |
| | 곽 관 웅 | 세종대학교 | 교 수 |
| | 조 영 조 | 한국전자통신연구원 | 책임연구원 |
| | 김 동 한 | 경희대학교 | 교 수 |
| | 정 영 숙 | 한국전자통신연구원 | 책임연구원 |
| | 전 진 우 | 한국로봇산업진흥원 | 실 장 |
| | 임 성 수 | 경희대학교 | 교 수 |
| | 문 인 혁 | 동의대학교 | 교 수 |
| | 권 용 관 | 대림대학교 | 교 수 |
| | 김 규 로 | 국가기술표준원 | 국 장 |
| | 김 석 중 | 하이젠모터 | 이 사 |
| | 김 승 훈 | 전자부품연구원 | 책 임 |
| | 김 혜 지 | 국가기술표준원 | 주 무 관 |
| | 단 병 주 | LG전자 | 수석연구원 |
| | 류 영 선 | 라스테크 | 대 표 |
| | 문 전 일 | 한국로봇산업진흥원 | 원 장 |
| | 박 종 환 | 널판 | 대 표 |
| | 박 홍 성 | 강원대학교 | 교 수 |
| | 서 준 호 | 한국로봇산업협회 | 팀 장 |
| | 서 태 원 | 로보테크 | 부 장 |
| | 심 재 홍 | 한국산업기술대학교 | 교 수 |
| | 유 원 필 | 한국전자통신연구원 | 책임연구원 |
| | 이 병 우 | 한국산업기술시험원 | 전문위원 |
| | 이 용 국 | 현대로보틱스 | 수석연구원 |
| | 임 우 철 | 두산로보틱스 | 선 임 |
| | 장 민 | 유진로봇 | 이 사 |
| | 정 병 찬 | 한화정밀기계 | 수석연구원 |
| | 지 수 영 | 한국전자통신연구원 | 책임연구원 |
| (간 사) | 유 기 은 | 한국로봇산업협회 | 대 리 |

**표준작성 기여자**

| | 성 명 | 근 무 처 | 직 위 |
|---|---|---|---|
| (과제제안자) | O O O | OOOOOOOOOOOOO | O |
| (작성참여자) | O O O | OOOOOOOOOOOOO | O |
| | O O O | OOOOOOOOOOOOO | O |
| | O O O | OOOOOOOOOOOOO | O |
| | O O O | OOOOOOOOOOOOO | O |
| | O O O | OOOOOOOOOOOOO | O |

# 목 차

# 서 문

## 표준의 목적

본 표준은 서비스 로봇에 사용되는 모듈의 공통 특성을 명시하는 추상화된 모델인 공통 정보 모델(CIM)을 설명한다. CIM은 다양한 제조업체와 디자이너가 개발하고 만든 로봇 모듈의 통합을 용이하게 한다. 모듈 간 관계와 API를 포함하는 속성과 메소드를 가진 클래스가 CIM을 구성한다. 모듈을 하드웨어 특성만을 가진 하드웨어 모듈, 소프트웨어 특성만으로 구성된 소프트웨어 모듈, 하드웨어 특징과 소프트웨어 특성을 모두 가진 모듈(HWSW 모듈로 칭함)로 구분하고, 이들 모듈들의 공통적인 클래스인 CIM의 구성요소(클래스 포함)를 제공하며, 구성 요소로써 각 클래스의 메소드와 속성을 정의하고, 이들 메소드와 속성 값을 제공하는 XML 형식의 명시 방법을 제시한다.

본 표준에서는 필수 항목은 “~하여야 한다,” 형태로, 권고사항은 “~하는 것이 좋다.” 혹은 “~하여야 할 것이다.” 형태로, 허용 항목은 “~해도 된다.” 형태로, 마지막으로 가능성 항목은 “~ 할 수 있다.” 형태로 문장을 서술한다. 이외에 대등한 문장은 KS A 001 Annex H를 참조하여 작성한다.

## 타 표준과의 관련성

### 타 국내표준과의 관련성

KOROS 1067-3:2018 개방형 로봇 소프트웨어 플랫폼 제3부: 프로파일
KOROS 1106-2:2019 서비스 로봇용 기계적·전기적 모듈화-제2부: 하드웨어 속성

### 타 국제표준과의 관련성

ISO FDIS 22166-1 Modualrity for service robots – Part1 : General requirements
IEC 62443-4-2, Security for industrial automation and control systems – Part 4-2: Technical security requirements for IACS components

## 지적재산권 인지 관련사항

해당사항 없음

## 적합인증 관련사항

해당사항 없음

## 표준의 이력

본 표준은 YYYY년 MM월 DD일에 개최된 제OO차 지능형로봇표준포럼 운영위원회에서 제정되었다.
본 표준은 YYYY년 MM월 DD일에 개최된 제OO차 지능형로봇표준포럼 운영위원회에서 KOROS XXXX:YYYY를 개정하였다.

**KOROS XXXX : YYYY**

# 서비스 로봇을 위한 모듈 – x-1 부 – 모듈을 위한 공통 정보 모델

Modularity for service robots – Part x-1 – Common Information Model for Modules

## 1 적용범위

이 표준은 서비스 로봇에 사용되는 모듈의 공통 특성을 명시하는 추상화된 모델인 공통 정보 모델(CIM)을 설명한다. CIM의 목적은 다양한 로봇 제조업체 및 개발자가 설계 및 개발하고 만든 로봇 모듈의 통합을 용이하게 하는 것이다. 해당 목적을 위해, CIM은 속성과 함수를 가진 객체 클래스를 제공하는데 모듈 아이디와 종류, 속성, 입력, 출력, 변수를 위한 클래스 및 상태, 건전성을 위한 클래스 그리고 함수 클래스, 인프라 클래스와 안전과 보안 레벨을 위한 클래스와 모델링 클래스 들로 구성된다.

.

이 표준의 대상 고객은 로봇 제조업체에 납품하는 모듈 제조업체/통합자설계자/개발자이다. 이 표준은 로봇 도메인에서 통합 및 재사용을 가능하게 하는 일종의 도구로 생각할 수 있으며, 특정 구현 또는 서비스 로봇 유형에 관계없이 모듈 간 상호 운용성 및 호환성을 촉진한다

## 2 인용표준

다음 문서는 전체 또는 일부가 이 문서에서 규범적으로 참조되며 해당 응용 프로그램에 반드시 필요하다. 날짜가 기입된 참고 문헌에 대해서는 인용된 판만 적용된다. 기한이 지난 참조의 경우 참조 문서의 최신 버전 (모든 개정 포함)이 적용된다.

ISO DIS 22166-1, Modurlarity for service robots – part 1 — General requirements

IEC 62443-4-2, Security for industrial automation and control systems – Part 4-2: Technical security requirements for IACS components

## 3 용어와 정의

이 표준의 목적을 위하여 다음의 용어와 정의를 적용한다.

### 3.1 정보모델 (Information model)

**IM**

동작 환경, 속성 및 함수 그리고 서로 간의 관계를 가진 객체의 추상화와 표현

### 3.2 공통 정보 모델 (Common Information model)

**CIM**

모든 모듈에 공통으로 적용되는 정보 모델

### 3.3 통합모델링언어 (Unified Modelling Language)

**UML**

디자인의 시각화에 의한 시스템 디자인을 돕는 모델링 언어

### 3.4 모듈 (module)

시스템 디자인과 통합 그리고 상호운용성 및 재사용성을 도모하기 위하여 속성 프로파일에 수반되어 정의된 인터페이스로 복합 또는 결합한 컴포넌트

### 3.5 소프트웨어 모듈 (software module)

**SW 모듈**

프로그램된 알고리즘으로만 구성된 모듈

### 3.6 하드웨어 모듈 (hardware module)

**HW 모듈**

메카니컬 부품과 전기적 회로와 같은 하드웨어 특성을 가지고 있거나 소프트웨어 특성을 동시에 가지고 있지만 외부와의 통신 인터페이스를 가지고 않고 메카니컬 인터페이스에 의해 연계 가능한 독립적으로 동작하는 모듈

예: 메카니컬 프레임, 통신기능이 없는 전원공급장치

### 3.7 하드웨어 및 소프트웨어 특성을 모두 가진 모듈 (module with HW aspects and SW aspects)

**HWSW 모듈**

메카니컬 부품과 전기적 회로와 같은 물리적 부품과 소프트웨어 특징을 가지고 있으면서 동시에 외부의 모듈과 데이터 교환을 할 수 있는 통신 인터페이스를 가진 모듈

예: 센서 모듈, 모터 모듈, 컴퓨팅 모듈

### 3.8 모듈 관리자 (Module manager)

모듈들을 로딩하고 두개 이상의 동일한 모듈 타입에 고유한 모듈 아이디를 할당하는 모듈

### 3.9 인스턴스 (Instance)

동작하거나 사용되는 프로세스 혹은 객체

## 4 모듈의 공통 정보 속성

### 4.1 개요

모듈의 공통 정보 모델이 지원하는 속성은 표4-1과 같다. 표4-1에 있는 공통 정보 처리를 위한 공통 정보 모델은 그림 5-3이 나타낸다. 표 4-1에 있는 'M' 과 'O'의 기호는 필수항목과 선택항목을 의미한다. 표 4.1의 공통정보 속성은 5장에서 자세히 설명한다.

**표 4.1. 공통 정보 속성과 관련 그룹화**

| 번호 | 속성 종류 | 필수(M) 또는 선택(O) | | | 관련 그룹화/태그 이름 |
|---|---|---|---|---|---|
| | | HWSW 모듈 | HW 모듈 | SW 모듈 | |
| 1 | Module Name | M | M | M | GenInfo (or General Information) |
| 2 | Description | O | O | O | |
| 3 | Manufactures | M | M | M | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | Examples | | O | O | O | |
| 5 | Module ID | | M | M | M | IDnType |
| 6 | Module type | | M | M | M | |
| 7 | Hardware Aspects | Basic | - | - | - | |
| | | Composite | M | M | - | |
| 8 | Software Aspects | Basic | - | - | - | |
| | | Composite | M | - | M | |
| 9 | Module properties | | M | M | M | Properties |
| 10 | Inputs | | M | M | M | IOVariable |
| 11 | Outputs | | M | M | M | |
| 12 | Function(capabilities) | | M | M | M | Services |
| 13 | Infrastructure | | M | O | O | Infra |
| 14 | Safety/Security | | M | M | M | SafeSecure |
| 15 | Modelling | | O | O | O | Modelling |

'Module Name'은 모듈의 이름을 명시하는 항목이다. 'Descrption' 은 모듈의 어떠한 부분인지, 무엇을 하는지, 어떻게 사용하는지 등의 모듈의 개괄적 설명을 명시한다. 'Manufactures' 는 제조사의 연락 정보를 제공한다. 'Examples'는 모듈의 전형적인 유즈케이스를 명시한다.
'Module ID'는 다음의 두 부분, 모듈 식별자와 모듈 객체 아이디로 구성된다: 전자는 모듈 제조사가 제공하는 고유의 식별자이며, 후자는 하나 이상의 모듈이 동일한 모듈 식별자를 가질 때, 모듈 통합자가 할당하거나 자동프로그램으로 할당된다. 'Module ID'는 하나의 서비스 로봇에서는 중복되는 것이 없어야 한다. 'Module Type'은 표 4.2의 구성타입(confType)과 표4.3의 기능타입(funcType)을 결합한 것이다. 'Module Type'의 형식은 confType과 funcType을 '_'을 사용하여 결합한 것이며, 전자의 것이 먼저 명시되어야 한다.

예 1: Basic module Type 으로 감지하는 모듈의 경우는 'BTS-SENS'라고 나타낼 수 있다.

**표 4.2 모듈구성 종류**

| 종류 | 설명 | 약자 |
|---|---|---|
| Composite Module Type 1 | 하나 이상의 모듈로 구성된 모듈로 HWSW 모듈, HW 모듈, SW 모듈의 조합으로 구성된 모듈 | CTALL |
| Composite Module Type 2 | 하드웨어 모듈로만 구성된 복합모듈 | CTH |
| Composite module Type 3 | 소프트웨어 모듈로 구성된 복합모듈 | CTS |
| Basic module Type 1 | 하드웨어 모듈 | BTH |
| Basic module Type 2 | 소프트웨어 모듈 | BTS |

**표 4.3 모듈기능 종류**

| 종류 | 설명(예) | 약자 |
|---|---|---|

| | | |
|---|---|---|
| 감지(Sensing) | 거리, 1D/2D/3D 이미지, 원점, 위치 등 | SENS |
| 구동(Actuating) | 서보모터, PWM 모터, 등 | ACT |
| 전원(Power) | 전기, 공압, 유압 | PWR |
| 인지/인식 (Perception/recognition) | 사물/사람/장애물 인식, 사람 식별, 얼굴인식, 등 | RECG |
| 제어/관리 (Control/management) | 제어 모듈, 감독, 관리SW, 안전/보안관리, 등 | CNTR |
| 통신(Communication) | 통신관련모듈 독립모듈. 통신의 예: 이더넷, 이더켓, CAN 등등 | COMM |
| 이동(Mobility) | SW모듈을 포함한 이동 관련 모듈 | MOBI |
| 조작(Manipulation) | SW모듈을 포함한 조작 관련 모듈 | MANI |
| 주행(Navigation) | SW모듈을 포함한 주행 관련 모듈 | NAVI |

만일 'Module Type' 이 'Basic' 이라면, 'Hardware Aspects' 나 'Software Aspects'에 해당하는 항목이 없어야 한다. 만일 'Module Type' 이 Composite 이라면, 'Hardware Aspects' 및/혹은 'Software Aspects'에 관련된 모듈의 식별자를 포함하여야 한다.

예 1: 만일 복합모듈인 주행 모듈이 라이다 모듈, 라이다 감지 SW 모듈, 카메라 모듈, 이미지 감지 모듈, 위치 인식 모듈로 구성되어 있다고 가정하자. 'Module Type'은 복합모듈타입1 과 주행 으로 CTALL_NAVI 이다. 'Hardware Aspects'는 라이다 모듈과 카메라 모듈이며, 'Software Aspects'는 라이다 감지 모듈, 이미지 감지 모듈 그리고 위치 인식 모듈이다.
'Module Properties'는 모듈의 초기화에 사용되는 값들이다. 만일 명시된 값이 시스템 에러를 일으키지 않는다면 동작시에도 사용될 수 있다. 'Module Properties'는 필수항목과 선택항목으로 나뉜다.

주 1: 환경적 제약도 속성으로 간주될 수 있다. 해당하는 예는 동작 온도, 동작 습도, 또는 허용되는 최대 충격량 등이 있다.

예 2: PID제어 알고리즘에 사용되는 계수는 초기에 한번 사용되거나, 값이 변경되어 연관된 소프트웨어 모듈의 실행 중에 여러 번 사용될 수 있다.

'Inputs'와 'Outputs'는 해당 모듈로 입력되거나 해당 모듈에서 출력되는 값 혹은 변수들을 말한다. 'Inputs'와 'Outputs'는 필수항목과 선택항목으로 나누어 명시할 수 있다. 상태는 모듈의 'Outputs'으로 제공되어야 한다.

예 3: 카메라 모듈의 입력값은 이미지 데이터이다. 엔코더를 가진 복합 서보 모듈의 입력값은 모터 제어 값이고 출력값은 엔코더의 값이다. 서보 모터 제어 소프트웨어 모듈의 입력값은 엔코더 값이고 출력값은 모터 제어 값이다.

주 2: 속성과 입력값은 모듈 관점에서 입력되는 것들이지만 입력값은 모듈의 환경에서 입력되는 내용이고 속성은 모듈 내부의 파라미터로 구분되어야 할 것이다. 예를 들어 서보모터 제어 소프트웨어 모듈의 입력값은 엔코더 값과 명령값이고, 속성은 P, I, D 계수들이다.

‘Functions’는 모듈이 제공하는 서비스 혹은 함수들이다. 구성 항목은 함수 이름, 함수의 전달 변수, 반환 값과 설명이다. 응용 프로그램의 인터페이스 형식과 유사하게 정의한다. 전달 변수는 없거나 하나 이상의 데이터 타입과 그 이름으로 구성된다. 오버로딩이 허용된다. ‘Functions’는 필수항목과 선택항목이 있다.

예 4: 기능의 형식은 표 4.4에 보여주고 있다. 해당 예는 int16 과 unint8은 표 4.7에 명시된 데이터 타입이다.

**표 4.4 함수의 형식 예**

| 이름 | 인자 내용 | 반환 데이터 형 | 설명 |
|---|---|---|---|
| Initialize | int16 val1, float64 val2 | uint 8 | 2 개 전달변수를 이용한 초기화<br>반환 값:<br>(0: 성공, 음수값: 에러타입) |
| | int16 val1, float64 val2, int32 val3 | uint 8 | 3 개 전달변수를 이용한 초기화<br>반환 값:<br>(0: 성공, 음수값: 에러타입) |

모듈의 ‘Capabilities’는 ‘Inputs’, ‘Outputs’, ‘Functions’를 사용하여 정의한다. ‘Description’ 항목에 모듈의 역량을 제안할 수 있다.
‘Infrastructure’은 모듈이 사용하고 있는 동력의 종류, 미들웨어 종류 등과 관련된다. 동력 종류는 동력 공급 장치 종류를 명시하고, 모듈의 동력 소비량에 관한 상세한 설명을 제공한다. 미들웨어 종류는 모듈이 사용하고 있는 미들웨어를 표시하는데, 그 예로는 ROS, openRTM, OPRoS 등이 있다.

주 3: 공통 정보는 채택된 미들웨어에서 적절하게 변환되어야 할 것이다.

주 4: 모듈의 전원 종류는 전기적, 공압적, 유압적 유형으로 구분할 수 있다. 예로 전기적 동력 관련 상세 설명은 (사용 전압, 소비 전류) 쌍으로 하나 이상이 표시된다.

‘Safety/Security’은 안전 관련 성능 레벨과 모듈이 제공하는 보안 정보를 제공한다.
‘Modeling’은 모듈의 시뮬레이션을 위해 3D 또는 유사 모델을 제공한다.

## 4.2 정보 모델 표현

### 4.2.1 일반 사항

공통 정보는 XML형식을 따른다

### 4.2.2 모듈의 GenInfo에 대한 정보

GenInfo 그룹을 위한 정보는 그림 4.1과 같이 제공해야 한다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<GenInfo>
  <ModuleName> Module Name </ModuleName>          <!-- string -->
```

```
  <Description> Description of module   </Description>      <!-- string -->
  <Manufactures> manufacturer </Manufactures>              <!-- string -->
  <Examples> list of use case </Examples>                   <!-- string -->
</GenInfo>
```

**그림 4.1 XML 형식의 GenInfo를 위한 정보**

모듈 이름은 다음과 같은 포맷을 사용하여야 하며, 기본모듈유형과 상세모듈유형으로 구성되며, 상세모듈유형은 상세 규격에서 정의하며, 이 2 문자열은 '_'로 연결된다. 만일 복합 모듈인 경우에는 가장 중요한 기능으로 기본모듈유형을 정한다.

기본모듈유형_상세모듈유형

기본 모듈유형은 표 4.5와 같이 정의된다.

**표 4.5 기본 모듈 유형의 값**

| 모듈 종류 | 기본모듈유형 | 설명 |
|---|---|---|
| 감지 모듈 | SENS | 감지 하는 모듈 |
| 구동 모듈 | ACT | 전기, 유압, 공압으로 구동하는 기기. 모터 등을 말함 |
| 파워 모듈 | PWR | 전기, 유압, 공압을 제공하는 모듈 |
| 관절 모듈 | JNT | 관절 관련 기구와 구동 모듈이 결합된 모듈 |
| 인지/인식 모듈 | PCT | 감지 모듈과 인지/인식 관련한 SW 모듈이 결합된 모듈 |
| 제어/관리 모듈 | CONT | HW 및 SW 들을 관리 및 제어하는 모듈 |
| 통신 모듈 | COMM | 통신에 관련된 모듈 |
| 이동 플랫폼 모듈 | MOBI | 이동 플랫폼 관련 복합 모듈 |
| 머니퓰레이터 모듈 | MANU | 머니퓰레이터 관련 복합 모듈 |
| 주행 모듈 | NAVI | 주행 관련 복합 모듈 |

### 4.2.3 모듈 ID와 모듈 유형 정보

IDnType 그룹을 위한 정보는 그림 4.2와 같이 제공해야 한다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<IDnType>
  <ID>
     <mid> module id     </mid>                  <!-- string provided by manufacturer -->
     <InstanceID>
        <GenMethod> AUTO </GenMethod>       <!-- AUTO or Manual -->
        <IID>   iid   </IID>                        <!-- Instance ID provided manually -->
     </InstanceID >
  </ID>
  <ModuleType>
    <Type>   conf_type_funcType   </Type>         <!-- see Table 2 for values -->
    <HWAspects> list of 3-tuple (ModuleName, mid, IID) </HWAspects> <!-- see examples -->
    <SWAspects> list of 3-tuple (ModuleName, mid, IID) </SWAspects> <!-- see examples -->
  </ModuleType>
```

```
</IDnType >
```

**그림 4.2 XML 형식의 IDnType을 위한 정보**

인스턴스 ID를 위한 GenMethod 태그 값은 표 4.6에 명시된 값이어야 한다.

**표 4.6 'GenMethod' 태그의 값**

| 값 | 설명 |
|---|---|
| AUTO | 감독관 또는 연관 프로그램에 의해 자동으로 생성된 객체 식별자 |
| MANUAL | 시스템 통합자나 모듈 개발자가 수동으로 할당하는 객체 식별자 |

인스턴스ID(IID)와 모듈 ID(mid)를 결합한 ID는 고유한 값이어야 한다. 만일 인스턴스 ID가 자동으로 생성되거나 GenMethod가 *Auto*이면 IID의 값이 필요하지 않다. HWAspects 태그에는 HWSW 모듈이나 하드웨어 모듈을 포함해야 한다. SWAspects는 소프트웨어 모듈만 포함해야 한다. HWAspects와 SWAspects는 3개의 값 (ModuleName, mid, IID)으로 표시되어야 하며, 리스트로 구성된다. 모듈이 수행되기 전에는 관리자 모듈이나 유사 모듈이 리스트에 존재하는지, 즉 중복되는지를 확인해야 한다. 만일 리스트에 열거된 모듈의 하나가 존재하지 않는다면 모듈은 실행되어서는 안 된다. 추가로 리스트에 동일한 모듈 ID가 있다면 인스턴스 ID는 모듈의 실행 전에 고유한 값으로 설정되어야 한다.

주 : 리스트에 동일한 모듈이 존재하지 않는다면 IID는 필요하지 않다

예 : **주행** 모듈의 예를 고려하자. 이 모듈은 복합 모듈로 구성 모듈은 카메라 모듈 (mid: CAM-234-VB), 라이다 모듈 (mid: Ld-YG2020-12), 이미지 감지 SW 모듈 (mid:IM-SW-XYCol-23), 거리 감지 SW 모듈 (mid: Dis-SW-vvXX-76), 위치 인식 SW 모듈 (mid:LO-SW-RXTv-396), 장애물 인식 모듈 (mid:OBS-SW-23-XNvT)이다. HWAspects는 다음의 2개 값을 가진 리스트를 가지고 있다: (SENS-Camera, CAM-234-VB, -) 와 (SENS-Lidar, Ld-YG2020-12,- ). 여기서 '-' 는 공백이거나 널(null) 값임. SWApects는 다음의 4개 값을 가진 리스트를 가지고 있다: (SENS-image sensing, IM-SW-XYCol-23, -), (SENS-distance sensing, Dis-SW-vvXX-76, -), (NAVI-localization, LO-SW-RXTv-396, -), (NAVI-obstacle recognition, OBS-SW-23-XNvT, -). 이 예를 XML로 명시하면 그림 4.3과 같다.

```
….
  <ModuleType>
    <Type>  CTALL_NAVI  </Type>        <!-- see Table 2 for values -->
    <HWAspects>
      (SENS-Camera, CAM-234-VB, ) (SENS-Lidar, Ld-YG2020-12, )</HWAspects>
    <SWAspects>
      (SENS-image sensing, IM-SW-XYCol-23,  )
      (SENS-distance sensing, Dis-SW-vvXX-76, )
      (NAVI-localization, LO-SW-RXTv-396,  )
      (NAVI-obstacle recognition, OBS-SW-23-XNvT, )
    </SWAspects>
  </ModuleType>
….
```

**그림 4.3 복합 모듈로써 주행 모듈의 모듈 타입 정보**

#### 4.2.4 ‘Properties‘ ‘Inputs‘ ‘Outputs‘ 정보

Properties 그룹과 IOVariables 그룹에 속한 ‘Properties‘ ,‘Inputs‘ ,‘Outputs‘은 모듈의 실행과 관련이 있고, SW 특징에 속한 파라미터와 변수로 사용된다. 이들의 속성들이 표 4.7에 표시되어 있다. 특히 변수가 입력과 출력을 동시에 할 수 있다면 ‘input/output‘로 표시한다. 필요하다면 동작 환경 또는 조건은 ‘Properties‘의 항목으로 제공되어야 한다.

**표 4.7 Property, Inputs and Outputs 의 속성**

| 속성이름 | 속성 | 입력 | 출력 | 입출력 | 설명 |
|---|---|---|---|---|---|
| Name | M | M | M | M | 예 : encoder, ratedCurrent , MaxCurrent, P_Coeff |
| Data type | M | M | M | M | 예 : int16, float64, int32, int8, uint16 |
| Unit | M | M | M | M | 예 : Ampere, meter, celsius, no unit, enum |
| Description | O | O | O | O | 설명 |
| value | O | O | O | O | 초기화를 위한 값 |

‘Name‘이라는 속성은 모듈에 사용되는 속성, 입력, 출력의 이름을 의미한다. ‘Data Type‘ 과 ‘Unit‘은 ‘Properties‘ ,‘Inputs‘ ,‘Outputs‘이 가지는 데이터 타입과 단위을 의미한다. 데이터 유형은 표4.8에서 하나를 선택해야 한다.

**표 4.8 속성 ‘Data Type’ 유형**

| 타입 | 크기(바이트) | 설명 |
|---|---|---|
| boolean | 1 | True(1) 또는 False(0) |
| int8 | 1 | 1 바이트 Signed integer |
| int16 | 2 | 2 바이트 Signed integer |
| int32 | 4 | 4 바이트 Signed integer |
| int64 | 8 | 8 바이트 Signed integer |
| uint8 | 1 | 1 바이트 unsigned integer |
| uint16 | 2 | 2 바이트 unsigned integer |
| uint32 | 4 | 4 바이트 unsigned integer |
| uint64 | 8 | 8 바이트 unsigned integer |
| float32 | 4 | 4 바이트 floating point |
| float64 | 8 | 8 바이트 floating point |
| enum | 4 | 열거형 타입 |

‘Data Type’의 속성이 ‘enum‘이라면, 정의하여 사용해야 한다. 속성 ‘Unit‘은 물리적 양의 단위를 의미하며 그 예로는 meter, rpm, sec, bps (bit per sec), ampere, volt, watt, and Celsius이다. 특히 물리적 양이 없을 경우는 ‘Unit‘에 ‘none‘이라 명시한다.

none : no data type
'OS Type'은 표 4.9에 나열된 열거 값을 가져야 한다. 만일 OS Type이 'OTHER' 값을 가지면, 관련된 XML파일 제공자는 그림 4.4 와 같은 추가된 정보를 제공해야 한다.

**표 4.9 OS type을 위한 열거형 값**

| 열거형 값 | 설명 |
|---|---|
| WIN32 | 32 비트 윈도우 OS |
| WIN64 | 64 비트 윈도우 OS |
| UBU32 | 32 비트 우분투 OS |
| UBU64 | 64 비트 우분투 OS |
| RED32 | 32 비트 레드햇 OS |
| RED64 | 64 비트 레드햇 OS |
| DEB32 | 32 비트 데비안 OS |
| DEB64 | 64 비트 데비안 OS |
| VxWorks | 실시간 OS |
| XENOMAI | 실시간 OS |
| PREEMPT | 실시간 OS |
| OTHER | 그 이외의 OS 타입 |

'Compiler Type'은 표 4.10에 나열된 열거 값을 가져야 한다. 만일 Compiler Type이 'OTHER' 값을 가지면, 관련된 XML파일 제공자는 그림 4.5 와 같은 추가된 정보를 제공해야 한다.

**표 4.10 컴파일러 타입을 위한 열거형 값**

| 열거형 값 | 설명 |
|---|---|
| VCC32 | 32 비트 비주얼 C++ 컴파일러 |
| VCC64 | 64 비트 비주얼 C++ 컴파일러 |
| GCC32 | 32 비트 GNU 리눅스 컴파일러 |
| GCC64 | 64 비트 GNU 리눅스 컴파일러 |
| GPP32 | 32 비트 GNU C++ 리눅스 컴파일러 |
| GPP64 | 64 비트 GNU C++ 리눅스 컴파일러 |
| MGWGPP32 | 32 비트 GNU C++ 윈도우 컴파일러 |
| MGWGPP64 | 64 비트 GNU C++ 윈도우 컴파일러 |
| OTHER | 그 외 컴파일러 타입 |

CPUbit의 태크의 값은 {8, 16, 32, 64}와 같은 열거 값을 가져야 한다:
Property와 IOVariable 클래스를 위한 정보는 그림 4.4와 같이 제공되어야 한다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Properties>
  <Property name=”maxRatedCurrent” type=”float32” unit=”ampere” description = “maximum of rated
```

```
current for motor”   value =15 />
  <Property name=”maxRatedVoltage” type=”float32” unit=”volt” description = “maximum of rated voltage for motor” >
     <value> 5 </value>
 </Property>
 <OperatingEnv>      <!-- operating environment or condition is provided -->
   <Temperature>   <!-- pair of maximum and minimum values (HW aspect) -->
      maxValue, minValue      <!-- float type -->
   </Temperature>
   <Humidity>   <!-- pair of maximum and minimum values (HW aspect)-->
      maxValue, minValue      <!-- float type -->
   </ Humidity>
   <OStype> Enumeration Value </OStype>   <!-- OS type, Table 4.8 -->
   <Compilertype> Enumeration Value </OStype>   <!-- compiler type to be used for module -->
   <CPUbits> Enumeration Value </CPUbits> <!-- the number of CPU bits to be used for module -->
 </OperatingEnv>
</Properties>
<IOVariables>
  <Inputs>
     <input name=”controlValue” type=”float32” unit=ampere description = “current control command to motor” >
        <value/>
     </input>
  </Inputs>
  <Outputs>
     <output name=”encoder” type =”uint32” unit = “none” description = “value of absolute encoder” value = 0 />
  </Outputs>
  <InOutputs>   <!-- input and output -->
     <inout name=”state” type=”uint8” unit = “none” description = “ reset or read status of motor”>
        <value />
      </inout>
  </InOutputs>
</IOVariables>
```

**그림 4.4 XML 형식의 Property와 IOVariable을 위한 정보**

```
   <OStype> OTHER </OStype>   <!-- OS type including the number of bits -->
   <OtherOS> OS X   </OtherOS>   <!-- this tag exist if OStype has the value ‘OTHER’. -->
```

**그림 4.5 XML 형식의 OSType이 OTHER인 경우 OtherOS을 위한 정보**

### 4.2.5 기능(역량)을 위한 정보

Service 그룹에 사용되는 기능(역량)을 위한 정보는 그림 4.6과 같이 제공되어야 한다. 정보는 C/C++, 자바, 파이선과 같은 프로그래밍 언어로 작성된 프로그램에서 사용된다. 그래서 전달 인자들의 순서를 결정하는 것과 인자들을 통하여 추가적 반환 값을 받는 것이 중요하다. 추가되는 반환 값으로써 사용되는 인자의 데이터 타입은 C/C++에서는 포인트를, 자바와 파이선에서는 참조 타입을 사용해야 한다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Services>
  <Function>
     <name>   initialize </name>          <!-- service/function name      -->
     <TypeBorO> BASIC   </TypeBorO>    <!-- enumeration : BASIC, OPTIONAL -->
     <arguments>
        <1st>
```

```
            <Aname> val1 </Aname>          <!-- argument name     -->
            <type> int16 </type>           <!-- data type of argument     -->
          </1st>
          <2nd>
            <Aname> val2 </Aname>
            <type> float64 </type>
         </2nd>
      <arguments>
      <ReturnValue type="uint8" />     <!--   data type of return value   -->
      <Description>
          Initialization using 2 arguments Return value: (0: success, negative value: error type)
      </Description>
  </Function>
  <Function>
      <name>   initialize </name>
     <TypeBorO> OPTIONAL   </TypeBorO>     <!-- enumeration : BASIC, OPTIONAL -->
      <arguments>
          <1st>
            <Aname> val1 </Aname>
            <type> int16 </type>
          </1st>
          <2nd>
            <Aname> val2 </Aname>
            <type> float64 </type>
         </2nd>
          <3rd>
            <Aname> val3 </Aname>
            <type> int32 </type>
         </3rd>
      <arguments>
      <ReturnValue type="uint8" />
      <Description>
          Initialization using 3 arguments Return value: (0: success, negative value: error type)
      </Description>
  </Function>
  <Function>
      <name>   read </name>
     <TypeBorO> BASIC   </TypeBorO>     <!-- enumeration : BASIC, OPTIONAL -->
      <arguments>
          <1st>
            <Aname> fd </Aname>      <!-- file descriptor -->
            <type> int </type>           <!-- depending on OS and CPU -->
          </1st>
          <2nd>
            <Aname> buf </Aname>
            <type> voidp </type>         <!-- void pointer type -->
          </2nd>
          <3rd>
            <Aname> nbytes </Aname>     <!-- number of bytes to read   -->
            <type> uint </type>          <!-- depending on OS and CPU -->
         </3nd>
      <arguments>
      <ReturnValue type="uint8" />
      <Description>
          Initialization using 2 arguments Return value: (0: success, negative value: error type)
      </Description>
  </Function>
</Services>
```

**그림 4.6 XML 형식의 Service 를 위한 정보**

### 4.2.6 인프라스트럭처에 대한 정보

Infra 그룹에서 제공하는 정보는 인프라 스트럭처의 타입 및 환경 보호에 관련된 정보이며, 동력 소스, 데이터 버스, 방수방진 등급(ingress protection,IP) 관련 정보이다. 동력 소스는 동력을 소비하는 모듈에 공급하는 동력 유형과 그 모듈이 공급하는 동력 유형(존재하는 경우)과 관련된다. 데이터 버스는 연결되는 모듈의 통신 유형과 관련되는데, 예로는 Ethernet, EtherCAT, CAN, USB 와 RS422 이다. IP는 모듈이 IEC 60529을 따라 제공되는 IP 코드 값과 관련이 있다. 정보는 그림 4.7 과 같이 제공되어야 한다.

데이터 버스용 정보는 다음을 제공하여야 한다: 프로토콜 유형, 전송 속도, 지원되는 API. 프로코톨 유형의 예는 Ethernet, EtherCAT, CAN, USB, RS422이며 지원되는 API의 예들은 initialize(), open(), read(), write(), close(), and control() 이다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Infra>
  <Powers>
    <PowerIn>
        <PowerType> Electric   </PowerType>   <!-- Electric, Pneumatic, Hydraulic   -->
        <consumedPower> 23   </consumedPower>   <!-- total power, unit: Watt-->
        <PowerType>
          <type Volt = 5 Ampere = 5 />     <!-- unit V, A -->
          <type Volt = 12 Ampere = 0.5 />     <!-- unit V, A -->
          <type Volt = 24 Ampere = 0.5 />     <!-- unit V, A -->
        </PowerType>
    <PowerIn>
    <PowerOut>     <!--   If there is no power from module, remove this item.   -->
      <PowerType> Electric   </PowerType>   <!-- Electric, Pneumatic, Hydraulic   -->
      <consumedPower> 9.2   </consumedPower>   <!-- total power, unit: Watt-->
      <PowerType>
        <type Volt = 5 Ampere = 0.4 />     <!-- unit V, A -->
        <type Volt = 12 Ampere = 0.2 />     <!-- unit V, A -->
        <type Volt = 24 Ampere = 0.2 />     <!-- unit V, A -->
     </PowerType>
   <PowerOut>
  </Powers>

  <DataBuses>   <!-- list of data buses used in Module -->
   <Databus>
     <Type> CAN </Type>
     <Speed> 1000 </Speed>     <!-- 1 Mbps, unit Kbps -->
     <API>
        <initialization> initialize() </Initialization>
        <open> open()   </open>
        <close> close()   </close>
        <readBlock> read()   </readBlock>
        <readChar> getc()   </readChar>
        <writeBlock> write()   </writeBlock>
        <writeChar> putc()   </writeChar>
        <control> control() </control>
    </API>
   </Databus>
```

```
    <Databus>
        <Type> EtherCAT </Type>
      ….
    </Databus>

    <Databus>
      <Type> RS422 </Type>
      …..
    </Databus>

  </DataBuses>
  <IP>    <!--   Ingress Protection IEC 60529-->
   <IPCode> IP23CH </IPCode>
  </IP>
</Infra>
```

**그림 4.7 XML 형식의 인프라 클래스를 위한 정보**

### 4.2.7 안전과 보안에 관한 정보

안전과 보안에 관한 정보는 모듈이 제공하는 안전 성능 레벨과 보안 관련 정보를 제공해야 한다. 그 정보는 그림 4.8과 같이 제공되어야 한다.

태그 PL은 나열값 [n, a, b, c, d, e]을 가져야 한다. 여기서 n은 안전 성능 레벨이 없는 것을 의미하여, a~e는 ISO 13849-1에서 정한 PLa~PLe를 의미한다. 보안 혹은 사이버보안은 나열 값 [0, 1, 2, 3, 4 ]을 가져야 한다. 여기서 0은 어떠한 보안 대응책이 없는 것을 의미하며, 1~4 는 IEC 62443-4-2에서 정한 보안 수준(SL) 1 ~ SL 4를 의미한다. 사이버 보안에 사용되는 대응책이 다양하기 때문에 모듈에서 제공되는 보안 대응책의 유형은 표 4.11에서 표시된 태그를 사용하여 표시하여야 한다. 물리적 보안은 ISO 22166-1의 5.6에서 정의된 내용을 사용한다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<SafeSecure>
  <Safety>
    <PL> a </PL>     <!-- choose one in a, b, c, d, e, and none.    -->
  </Safety>
  <Security>
    <PhysicalSecurity>
     <!-- None, LatchSensor, LockwithKey, LockwithActuator , see sub-clause 5.6 in ISO 22166-1 -->
     None
    </PhysicalSecurity>
    <CyberSecurity>     <!-- list security functions provided by module -->
      <TagType 1>     </TagType 1>   <!--    see Table 4.10 -->
        …
      <TagType N>     </TagType N>   <!--    see Table 4.10 -->
    </CyberSecurity>
  </Security>
</SafeSecure>
```

**그림 4.8 XML 형식의 class SafeSecure 를 위한 정보**

**표 4.11사이버 보안에 사용되는 태그 리스크**

| 보안 대응책 유형(IEC 62443-4-2) | 태그 유형 |
|---|---|
| Human user identification and authentication | HU_IA |
| SW process and device identification and | SD_IA |

| | |
|---|---|
| authentication | |
| Account management | ACNT_MGT |
| Identifier management | ID_MGT |
| Authenticator management | AUTH_MGT |
| Wireless access management | WIRELEE_MGT |
| Strength of password-based authentication | PW_AUTH |
| Public key infrastructure certification | PK_CERT |
| Strength of public key-based authentication | STR_PK_AUTH |
| Unsuccessful login attempts | LOGIN_NO |
| Access via untrusted networks | ACC_UNTRUST_NET |
| Authorization enforcement | AUTHORIZE |
| Wireless use control | WIRELESS_USE |
| Session lock | SESS_LOCK |
| Remote session termination | SESS_TERM |
| Concurrent session control | SECC_CNTR |
| Auditable events | AUDT_EVT |
| Timestamps | TIMESTM |
| Non-repudiation | NON_REP |
| Communication integrity | COMM_INTG |
| Protection from malicious code | PROT_MALI_CODE |
| Security functionality verification | SECUR_VERIFY |
| Software and information integrity | SW_INTGT |
| Input validation | INPUT_VALD |
| Deterministic output | DET_OUT |
| Error handling | ERR_HNDL |
| Session integrity | SESS_ INTGT |
| Information confidentiality | INFO_CONFI |
| Information persistence | INFO_PERS |
| Use of cryptography | CRYTO |
| Restricted data flow | RSTIC_FLOW |
| Denial of service protection | DoS |
| Resource management | RESOU_MGT |
| Control system recovery and reconstitution | CNTR_RECOV_RECON |

#### 4.2.8 모델링에 관한 정보

모델링에 관한 정보는 Modelling 그룹에서 정의되며, 모듈이 제공하는 시뮬레이션을 위한 정보를 제공해야 한다. 이러한 정보의 예는 3D모델과 Universal Robotic Description Format (URDF)파일이다. 이러한 정보는 그림 4.9와 같이 제공되어야 한다. 만일 URDF파일이나 3Dmodel파일이 없다면, 해당 사항에 'NA' (not avaiable)라 명시해야 한다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Modelling>
    <Simulator>  <!--list of Simulation programs used for 3D graphic model and model description -->
       SimulatorName1 ….  SimulatorNameN  <-- e.g. gazebo or commercial simulation prog. -->
    </Simulator>
    <URDF>      <!-- model description file name for model of module -->
      ../../moduleDescrtionFileName // the path of model description file or NA e.g. urdf file, sdf file
    </URDF>
    <3Dmodel>  <!-- 3D graphic file name for model of module -->
      ../../module3DGraphicFileName // the path of 3Dgraphic file model file or NA
    </3Dmodel>
</Modelling>
```

**그림 4.9 XML 형식의 모델링을 위한 정보**

# 5 모듈을 위한 공통 정보 모델

## 5.1 개요

CIM은 일종의 객체 지향 모델링 기술인 UML 표기법을 사용하여 규정한다.
모듈을 위한 정보 모델은 모든 유형의 모듈에 대한 공통 정보 모델을 그룹화해야 하는데, 모듈의 유형으로는 HWSW 모듈, 소프트웨어 모듈, 하드웨어 모듈이 있다. HWSW 모듈은 소프트웨어 특성과 HW 특성을 모두 가진 모듈의 조합이다. 하드웨어 특성만 가진 모듈의 예는 기구적 프레임이나 커버 또는  조인트 기구물이다. 하나의 모듈로써 서비스 로봇의 정보 모델을 표현하면 그림 5.1과 같다. 그림 5.2는 그림 5.1에 기반한 이동 로봇용 정보 모델을 나타낸다.

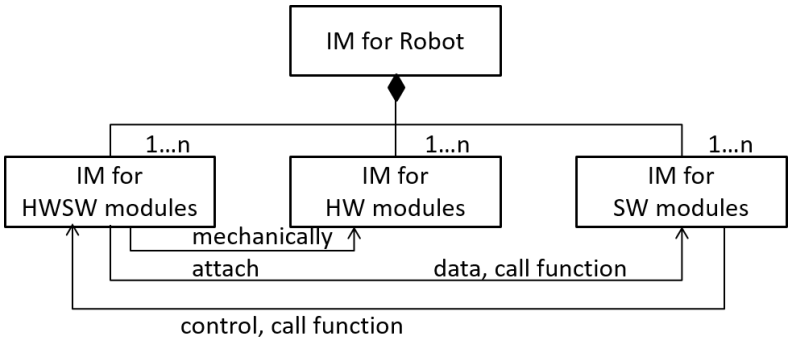


**그림 5.1 서비스 로봇용 정보 모델**

예 : 이동 로봇을 위한 정보 모델의 예가 다음 그림과 같다.

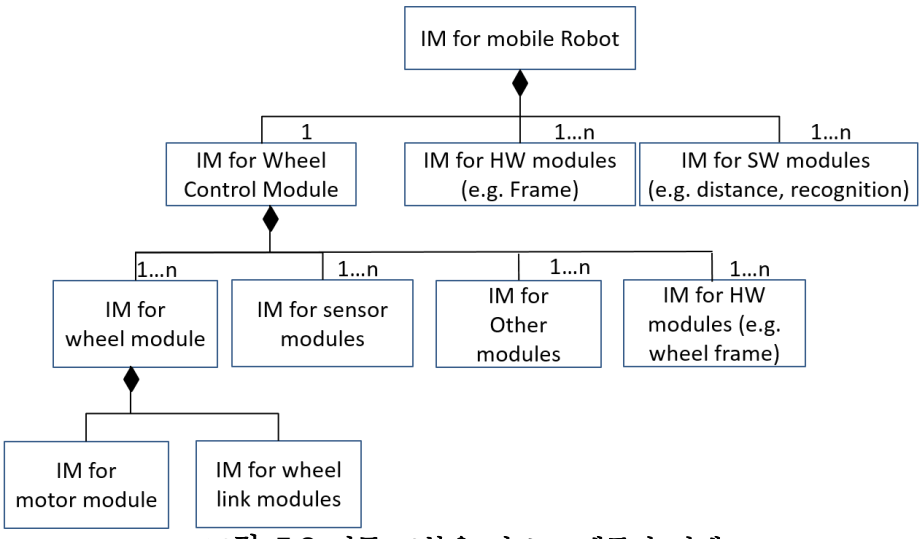


**그림 5.2 이동 로봇용 정보 모델들간 관계**

## 5.2 공통 정보 모델을 위한 클래스

### 5.2.1 개요

공통 정보 모델은 그림 5.3과 같이 9개의 클래스로 구분해야 한다: 표 4.1의 8개 클래스와 추가된 클래스는 모델의 상태/건전성 관련 정보이다. 후자 클래스는 모듈의 실행 중에만 사용된다.

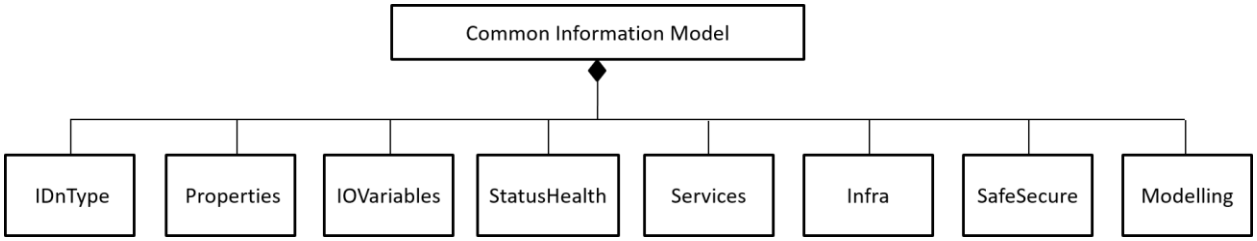


**그림 5.3 공통 정보 모델 클래스**

공통 정보 모델 (CIM) 클래스는 그림 5.4의 속성을 가져야 하고, 해당 속성은 ISO 22166-1을 기반으로 한다. 모듈 이름, 설명, 제조사, 예제와 같은 속성의 값은 그림 4.2를 통해서 제공한다. 정보 모델 버전은 모듈이 명세될 때, 사용되는 정보 모델의 버전 번호이다. 버전 번호는 표준이 수정될 때마다 업데이트된다. oXXXX 기호는 객체 변수를 의미한다. 그림 5.4의 IDnType, Propertiees, IOVariables, StatusHealth, Services, Infra, SafeSecure, and Modelling 는 그림 5.4 - 그림 5.11 에 설명된 클래스들이다.

주1 : 현재 버전은 1.0이다.

주2 : 속성은 다음 표식 중 하나를 사용한다: private (-), protected (*), public (+). 속성의 이름이 먼저 나오고, 속성의 데이터 타입을 그다음에 표시한다. 속성의 이름과 속성의 데이터 타입의 구분자는 ':' 이다. 만일 속성이 public 멤버로 선언 된 경우 해당 속성에 접근하기 위한 함수를 정의할 필요가 없다.

| Class CIM |
|---|
| - ModuleName : String<br>- Description : String<br>- Manufacturer : String<br>- Examples : String<br>- InformationModelVersion : String<br>+ oIDnType : IDnType<br>+ oProperties : Properties<br>+ oIOVariables : IOVariables<br>+ oStatusHealth : StatusHealth<br>+ oServices : Services<br>+ oInfra : Infra<br>+ oSafeSecure : SafeSecure<br>+ oModelling : Modelling |
| + getModuleName() : String<br>+ putModuleName() : Boolean<br>+ getDescription() : String<br>+ putDescription() : Boolean<br>+ getManufacturer() : String<br>+ putManufacturer() : Boolean<br>+ getExamples() : String<br>+ put Examples() : Boolean |

**그림 5.4 CIM 클래스 다이어그램**

### 5.2.2 모듈 ID와 모듈 유형 클래스

모듈 ID와 모듈 유형 정보는 IDnType의 클래스로 명시해야 한다. IDnType 클래스는 그림 5.5에 명시

된 속성을 가져야 한다. 그림 5.5의 속성 값은 그림 4.3을 통해서 제공된다. 모듈 내에 동일한 타입의 모듈이 사용된다면 모듈은 개별적으로 구별되어야 한다. 이러한 목적으로 Instance ID가 필요하며, 모듈 관리자에 의해서 고유하게 생성되어야 한다. 만일 IIDGen의 속성 값이 AUTO이면, 속성 IID는 genInstanceID()를 이용하여 값을 갖게 된다. 만일 IIDGen 의 속성 값이 Manual 이면, 속성 IID는 setInstanceID()를 이용하여 값을 갖게 된다. 값들이 정해지면 IID는 고유한 값을 가진다. 따라서 genInstanceID()와 setInstanceID()는 사용되는 IID가 고유한지 확인해야 한다. getInstanceID()는 IID를 가져오는 데 사용된다.

| Class IDnType |
|---|
| + moduleID : String<br>+ IIDGen : Enumeration // AUTO or Manual<br>- IID : int32 // Instnace ID<br>+ ModuleType: String<br>+ oHWAspects: Vector of 3-tuple ( ModuleType, moduleID, IID)<br>+ oSWAspects: Vector of 3-tuple ( ModuleType, moduleID, IID) |
| + getInstanceID() : int32<br>+ setInstanceID(ID: int32 ) : Boolean<br>+ genInstanceID() : int32 |

**그림 5.5 IDnType 클래스 다이어그램**

### 5.2.3 Property 클래스

Property 정보는 Property 클래스에 명시되어야 한다. Property 클래스는 그림 5.6같이 모듈의 실행과 동작 환경 및 조건을 위해서 필요한 정보를 포함한 모듈의 속성을 제공해야 한다. 모듈의 실행에 필요한 정보로는 모듈의 초기화를 위한 값들과 관련이 된다. 상세한 함수 이름과 관련된 데이터 타입과 같은 속성은 그림 4.4와 같이 Property 태그의 값을 사용하여 정의해야 한다.

예 : 그림 4.4의 예는 다음과 같다:

```
<Property name="maxRatedCurrent" type="float32" unit="ampere" description = "maximum of rated current for motor" value =15 />
```

해당 태그로부터 변수와 데이터 유형의 요소가 생성된다.

+ maxRatedCurrent = 15 : float32 or public float32 maxRatedCurrent=15

다른 모듈들이 모듈의 속성 값 또는 변수로써 정의된 속성 값을 이용하기 위해서는 다음과 같은 형식을 가져야 한다: 4개의 영역이 '_'로 연결된다.

<ModuleType>_ <ModuleID>_<InstanceID>. <Name defined in module>

Module ID, instance ID, Module Type은 그림 4.3과 그림 5.5에 제공된다. 변수로써의 속성 이름은 그림 4.4와 그림 5.6에 제공된다.

실행 환경과 조건은 하드웨어 특성의 모듈이 동작하는 조건이고 소프트웨어 모듈이 실행되는 환경이다. 전자의 예는 온도, 습도, 전압, 전류 등의 동작 범위 등이다. 후자의 예는 OS types, compiler type, and CPU type (or 8/16/32/64 bit ). 이러한 변수는 그림 4.4를 사용하여 제공해야 한다. 4개의 변수 maxTemp, minTemp, maxHumidity, minHumidity는 그림 4.4의 값 태크를 이용하여 생성되야한다.

그림 5.6에는 2 종류의 속성들이 제공되어 있는데, 속성 이름이 정의되어 있지 않는 형태와 속성 이름이 정의되어 있는 형태이다. 속성 이름이 정의되어 있는 내용에서도 하드웨어적인 속성과 소프트웨어적인 속성으로 구분되어 있어 해당 모듈에 있어서 알맞은 값을 해당 속성에 넣어야 한다. 속성 이름이 정의되어 있지 않는 경우는 해당 속성들은 그림 4.4를 사용하여 제공하며, 이를 위하여 다음

과 같은 클라스를 사용하여 정의하여 이를 클라스 Properties에 적용한다.

| Class Property |
| --- |
| PropertyName : String // attributes provided in the tag Property<br>PropertyName = initialValue : data type // if initial value is provided, initialValue is given.<br>// In thid field, PropertyName is a kind of variable name. |
| |

| Class Properties |
| --- |
| - properties[] : Property // list of attributes generated according to the tag Property<br>- maxTemp : float32 // maximum value of temperature in operating environment<br>- minTemp : float32 // minimum value of temperature in operating environment<br>- maxHumidity : float32 // maximum value of humidity in operating environment<br>- minHumidity : float32 // minimum value of humidity in operating environment<br>- OpOS : Enumeration // OS type in operation. Table 4.8<br>- UsedCompilerType : Enumeration // compiler type to be used, Table 4.9<br>- OpCPUBit : Enumeration // the number of CPU bit to be used. 8, 16, 32, 64 |
| + putProperty(propertyName: String, value: data type) : int16<br>+ getProperty(propertyName: String): data type<br>+ putMaxTemp(value: float32) : int16<br>+ getMaxTemp(): float32<br>+ putMinTemp(value: float32) : int16<br>+ getMinTemp(): float32<br>+ putMaxHumidity(value: float32) : int16<br>+ getMaxHumidity(): float32<br>+ putMinHumidity(value: float32) : int16<br>+ getMinHumidity(): float32<br>+ putOpOS(value: Enumeration) : int16<br>+ getOpOS():Enumeration<br>+ put UsedCompilerType(value: Enumeration) : int16<br>+ get UsedCompilerType():Enumeration<br>+ put OpCPUBit(value: Enumeration) : int16<br>+ get OpCPUBit():Enumeration |

**그림 5.6 Properties 클래스 다이어그램**

### 5.2.4 모듈의 입력, 출력 및 공유변수를 위한 클래스

모듈의 입력, 출력, 공유변수를 위한 정보는 그림 5.7과 같이 IOVarables 클래스에 정의되야 한다. IOVarables 클래스는 입력, 출력, 모듈에 사용되는 모듈의 실행에 필요한 변수들을 제공해야 한다. 이름과 데이터 타입에 관련된 요소와 같은 상세한 속성은 그림 4.4의 IOVarables 태그의 값을 이용하여 정의해야 한다.

다른 모듈들이 모듈의 속성값 또는 변수로써 정의되어진 속성값을 이용하기 위해서는 다음과 같은 형식을 가져야 한다: 4개의 영역이 '_'으로 연결된다.

<ModuleType>_ <ModuleID>_<InstanceID>. <Name defined in module>

Module ID, instance ID, Module Type는 그림 4.3과 그림 5.5에 제공된다. 변수로써의 속성이름은 그림 4.4와 그림 5.7에 제공된다.

예 : 그림 4.4에서 제공된 예를 보면 다음과 같다.

```
<IOVariables>
  <Inputs>
    <input name=”controlValue” type=”float32” unit=ampere description=“current control command to motor” >
        <value/>
    </input>
  </Inputs>
  <Outputs>
    <output name=”encoder” type =”uint32” unit = “none” description = “value of absolute encoder” value= 0 />
  </Outputs>
  <InOutputs>  <!-- input and output -->
    <inout name=”state” type=”uint8” unit = “none” description = “ reset or read status of motor”>
        <value />
     </inout>
  </InOutputs>
</IOVariables>
```

위의 태그로부터 다음과 같은 변수와 데이터 유형이 생성되고, 이러한 생성된 데이터가 그림 5.7에 사용된다.

+ controlValue : float32  or public float32 controlValue // current control command to motor
+ encoder=0 : float32  or public float32 encoder=0  // value of absolute encoder
+ state : uint8  or public uint8 state  // reset or read status of motor

| Class IOVarables |
|---|
| + InputName1 : data type1  // attributes generated according to the tag Input<br>…<br>+ InputNameN=initialVaule : data typeN  // attributes generated according to the tag Input<br>+ OutputName1 : data type1  // attributes generated according to the tag Output<br>…<br>+ OutputNameN=initialValue : data typeN  // attributes generated according to the tag Output<br>+ IOName1 : data type1  // attributes generated according to the tag InOutput as variable<br>…<br>+ IONameN=initialValue : data typeN  // attributes generated according to the tag InOutput as variable |
| |

**그림 5.7 IOVarables 클래스 다이어그램**

### 5.2.5 모듈의 Status, Health codition을 위한 클래스

모듈의 Status, Health codition를 위한 정보는 그림 5.8과 같이 StatusHealth 클래스에 정의되어야 한다.

StatusHealth 클래스는 모듈의 상태나 건전성 조건을 제공한다. 만일 하드웨어 정보 모델이 존재한다면, 건전성 조건을 말하며, 소프트웨어 정보 모델이 존재한다면, 통신 상태나 동작중 일어나는 에러 타입을 말한다. setExecStatus() 함수는 실행 수명 사이클을 관리해야 할 것이다. 함수 setErrorRecovery()는 안전 관리자에 의해서 실행되고, 만일 함수가 성공적으로 실행되면 입력된 inStatus로 상태 전이를 한다. 변수 ExecutionStatus와 inStatus는 DIS 22166-1의 그림 6에 나와 있는 상태의 하나이어야 한다.

| Class StatusHealth |
|---|

| |
|---|
| - HealthCond : Enumeration<br>- ExcutionStatus : Enumeration<br>- ErrorType : int32 |
| + resetStaus() : Boolean<br>+ resetHealthCond(); Boolean<br>+ resetExecutionStatus(); Boolean<br>+ resetCommStatus(); Boolean<br>+ getHealthCond() : Enumeration value of Health condition or status of module<br>+ setHealthCond(inStatus: Enumeration value) : Boolean<br>+ getExecStatus() : Enumeration value of Execution life cycle status<br>+ setExecStatus(inStatus: Enumeration value) : Boolean<br>+ setErrorRecovery(inStatus: Enumeration value) : Boolean<br>+ getErrorType() : int32<br>+ resetErrorType() : Boolean |

**그림 5.8 StatusHealth 클래스 다이어그램**

### 5.2.6 모듈의 서비스를 위한 클래스

모듈의 서비스는 서비스 클래스 안에 정의해야 한다. 서비스 클래스는 모듈이 지원하는 함수를 그림 5.9와 같이 제공한다. 모듈의 함수 속성은 그림 4.6의 Function 태그의 값을 사용하여 정의해야 한다.

| Class Services |
|---|
| + NoOfBasicService : int8 // Number of Basic services provided<br>+ NoOfOptionalService : int8 // Number of Optional services provided<br>+ BasicServDes[] : String // Descriptions of Basic services provided<br>+ OptionalServDes[] : String // Descriptions of Basic services provided |
| // basic(or Mandatory) Services<br>+ serviceName_b1(ArgName1: dataType1, .., ArgNameN: dataTypeN) : type of return data<br>..<br>+ serviceName_bM(ArgName1: dataType1, .., ArgNameN: dataTypeN) : type of return data<br>// optional(or Special) Services<br>+ serviceName_s1(ArgName1: dataType1, .., ArgNameN: dataTypeN) : type of return data<br>..<br>+ serviceName_sM(ArgName1: dataType1, .., ArgNameN: dataTypeN) : type of return data |

**그림 5.9 Services 클래스 다이어그램**

예 : 그림 4.6에서 제공되는 것을 고려하면 다음과 같다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Services>
  <Function>
     <name>   initialize </name>          <!-- service/function name      -->
     <TypeBorO> BASIC   </TypeBorO>    <!-- enumeration : BASIC, OPTIONAL -->
     <arguments>
        <1st>
           <Aname> val1 </Aname>        <!-- argument name     -->
           <type> int16 </type>          <!-- data type of argument     -->
         </1st>
         <2nd>
           <Aname> val2 </Aname>
           <type> float64 </type>
        </2nd>
```

```
      <arguments>
      <ReturnValue type="uint8" />    <!--   data type of return value   -->
      <Description>
          Initialization using 2 arguments Return value: (0: success, negative value: error type)
      </Description>
  </Function>
  <Function>
      <name>   initialize </name>
     <TypeBorO> OPTIONAL   </TypeBorO>    <!-- enumeration : BASIC, OPTIONAL -->
      <arguments>
          <1st>
              <Aname> val1 </Aname>
              <type> int16 </type>
           </1st>
           <2nd>
              <Aname> val2 </Aname>
              <type> float64 </type>
          </2nd>
           <3rd>
              <Aname> val3 </Aname>
              <type> int32 </type>
          </3rd>
       <arguments>
      <ReturnValue type="uint8" />
      <Description>
          Initialization using 3 arguments Return value: (0: success, negative value: error type)
      </Description>
  </Function>
</Services>
```

위의 태그로부터 아래와 같은 변수와 함수들이 생성된다.

```
+ NoOfBasicService =1 : int8              // Number of Basic services provided
+ NoOfOptionalService=1 : int8          // Number of Optional services provided
+ BasicServDes[] : String               // Descriptions of Basic services provided
+ OptionalServDes[] : String
// basic(or Mandatory) Services
+ initialize(val1: int16, val2 : float64) : uint8
// optional(or Special) Services
+ initialize(val1: int16, val2 : float64, val3:int32) : uint8
```

#### 5.2.7 모듈의 Infrastructure를 위한 클래스

모듈의 Infrastructure를 위한 정보는 Infrastructure 클래스에 정의해야 한다. Infrastructure 클래스는 그림 5.11과 같이 모듈이 지원하는 Infrastrucrue 타입을 제공해야 할 것이다. 모듈의 시제품 함수는 그림 4.7의 Function 태그의 값을 이용하여 정의해야 한다. 속성 powerIn와 powerOut은 모듈에 인가되는 전원 입력과 모듈로부터 출력되는 전원 출력이다. 후자의 것은 존재한다면 명시해야 한다.

서비스 클래스에 사용되는 ElecPWR와 DataBus는 그림 5.10에 나타나 있다.

```
class ElecPwr {
   int16 count ; // the number of supplied powers
   class ElecComp {
      float32 volt;       // voltage
      float32 current;
   } pwr[count] ;
```

```
    void ElecPWR(int16 count)       // constructor
}
Class DataBus {
    String type;     // Physical/MAC protocol type used in module : CAN2.0, EtherCAT, Ethernet, RS485
    float32 speed; // transmission speed, unit : Kbps
    int16 initialize();   // overriding is needed.
    int16 open();   // overriding is needed.
    int16 read();   // overriding is needed.
    int16 write();// overriding is needed.
    int16 conclude();// overriding is needed.
    int16 control();   // overriding is needed.
}
```

**그림 5.10 ElecPWR 및 DataBus 클래스 다이어그램**

| Class Infrastructure |
|---|
| // Electric power type<br>+ powerIn : ElecPWR // Input electric Power into a module.<br>+ powerOut : ElecPWR // Output electric Power from a module, if any<br>// Pneumatic and Hydraulic types will be defined<br>// dataBuses<br>+ noBuses : int8 // the number of databuses used in module. Normal value is 1.<br>+ dataBus[noBuses] : DataBus<br>// types of DataBase mgt system<br>+ dbType : String // DBMS type used in module. e.g oracle, sql, …<br>// IP code<br>+ IPcode : String // Ingress Protection IEC 60529 |
| |

**그림 5.11 Infrastructure 클래스 다이어그램**

### 5.2.8 모듈의 Safety 와 Security 정보를 위한 클래스

SafeSecure 클래스는 그림 5.13과 같이 모듈이 제공하는 안전 레벨과 보안 레벨을 명시해야 한다. 사이버 보안 대응 방법은 보안 종류와 해당 종류에 대한 보안 수준(0~5)을 명시하여야 하고, 모듈은 0개 이상의 보안 대응 방법을 가질 수 있다. 사이버 보안 대응 방법은 그림 5.12와 같은 클라스를 사용하여 정의하여야 한다.

주 : 안전 관련 기능 및/또는 보안 관련 기능은 안전/보안 관리에 사용할 수있는 서비스 클래스에서 제공된다.

| Class CyberSecurity |
|---|
| - securityType : String // Tag Name in Table 4.10<br>- securityLevel : Enumeration // Cyber Security 0,1,2,3,4 |
| |

**그림 5.12 CyberSecurity 클래스 다이어그램**

| Class SafeSecure |
|---|
| - SafetyLevel : Enumeration // SIL<br>- PhySecurityLevel : Enumeration // Physical Security |

| - CybSecurityLevel[] : CyberSecurity |
|---|
| + getSafetyLevel() : Enumeration<br>+ setSafetyLevel(inSL: Enumeration) :Boolean<br>+ getPhySecurityLevel() : Enumeration<br>+ setPhySecurityLevel(inSL: Enumeration) :Boolean<br>+ getCybSecurityLevel(securityType:String) : Enumeration // if securityType is found, return<br>// the corresspoding SL. Othewrwise return 0.<br>+ setCybSecurityLevel(cybSec: CyberSecurity) :Boolean |

**그림 5.13 SafetySecurity 클래스 다이어그램**

### 5.2.9 모듈의 Modelling을 위한 클래스

Modelling 클래스는 그림 5.14과 같이 시뮬레이션을 위한 모듈이 지원하는 모델링 관련 정보를 제공해야 한다.

모델을 위한 정보는 모듈이 시뮬레이션을 위해 제공하는 정보를 제공해야 하는데, 그 예로는 3D모델과 Universal Robotic Description Format (URDF) 파일이 있다. 이러한 정보는 그림 4.9와 같이 제공되야 한다. URDF파일이나 3D파일이 없다면 해당 항목에 “NA” (not available)라고 명시해야한다.

| Class Modelling |
|---|
| + simulator[] : String // list of Simulation programs<br>+ mdf : String // path of model description file or “NA”<br>+ 3Dmodel: String // path of 3Dgraphic file model file or “NA” |
| |

**그림 5.14 Modelling 클래스 다이어그램**

지능형로봇표준포럼

# 표준서의 서식 및 작성방법

**Rules for the drafting and presentation of Korea Robot Standards**

**KOROS XXXX : YYYY**

제 정 자 : 지능형로봇표준포럼 의장　　제정 : YYYY년 MM월 DD일

지능형로봇표준포럼 사무국
서울시 용산구 한강대로 31 금영빌딩 8 층한국로봇산업협회
전화 : (02) 780-3060