碩士學位論文

# ALE 미들웨어 기반
# 다중 RFID 코드 변환 처리

濟州大學校 大學院

컴퓨터工學科

邊 持 熊

2008年 12月

# ALE 미들웨어 기반
# 다중 RFID 코드 변환 처리

指導敎授 邊 暎 哲

邊 持 熊

이 論文을 工學 碩士學位 論文으로 提出함

2008年 12月

邊持熊의 工學 碩士學位 論文은 認准함

審査委員長 _____ 印

委 員 _____ 印

委 員 _____ 印

濟州大學校 大學院

2008年 12月

# Multi-RFID code conversion process based on ALE

Ji-Woong Byun
(Supervised by professor Yung-Cheol Byun)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Computer Engineering

2008. 12.

This thesis has been examined and approved.

Thesis director, _____

Thesis director, _____

Thesis director, _____

December 2008

Department of Computer Engineering
Graduate School
Cheju National University

# 감사의 글

2007년에 대학원 생활을 시작하기 이후부터 어느새 2년이라는 시간이 쏜살같이 흘러가게 되었습니다. 영어영문학 전공을 하였고 컴퓨터공학에서 대학원 생활을 하였던 제가 이제는 논문에 담을 감사의 글을 쓰게 되었습니다.

먼저 제가 대학원에 들어올 수 있도록 격려와 용기를 주셨던 저의 지도교수님인 변영철 교수님에게 감사의 말을 전하고 싶습니다. 또한 저의 논문 지도에 애써주신 김장형 교수임과 이상준 교수님께도 감사합니다

그리고 논문이 완성되기까지 세심한 지도를 해주셨던 안기중 교수님, 변상용 교수님, 곽호영 교수님, 송왕철 교수님, 김도현 교수님께 감사합니다.

항상 대학원 연구실 생활 내내 저의 논문을 도와주고 같이 연구 생활을 하였던 노영식씨와, 저와 같이 졸업하면서 연구원 생활을 같이하였던 양문석씨, 연구에 도움을 주었던 차지윤, 그리고 요종이와 가영이, 혜영이, 대오도 감사드립니다.

그리고 대학원 생활에 대한 조언을 아끼지 않았던 한경복 선배님, 권훈 선배님, 김정희 선배님께 감사드리고, 저의 연구실 선배인 홍연미 선배님과, 실질적인 대학원 생활을 도와주셨던 정은경 선생님과 이정화 선생님께 감사하다는 말을 전하고 싶습니다. 그리고 대학원 생활동안 같이 수업을 받았던 김용우와 이철식 선생님에게 감사하다는 말을 전하고 싶습니다. 그 외에 대학원 생활에서 많은 도움을 주었던 선후배님들에게 감사하다는 말을 전합니다.

끝으로 고생했던 대학원 생활동안 뒤에서 응원을 해주었던 대학교 친구들인 정환이, 형진이, 제현이, 지은이 등에게 감사하고 초등학교 때부터 쭈욱 같이 지내면서 울고 웃고 떠든 경수, 영재, 상현에게도 감사하고 마지막으로 감사한다는 말을 천번 만번 넘게 말하여도 언제나 부족한 저의 부모님께 감사하다는 말을 남깁니다.

# 목 차

# 그림 목차

# 표 목차

## 국문초록

# ALE 미들웨어 기반 다중 RFID 코드 변환 처리

컴퓨터공학과 변지웅
지 도 교 수 변영철

최근 유비쿼터스 시대의 도래로 인하여 RFID가 핵심적인 기술로 관심 받고 있다. 특히, RFID 기술 중 하나인 RFID 미들웨어는 EPCglobal에서 제안한 ALE 스펙으로 인하여 사실상 표준으로 제정되었다. 그러나 국내 RFID 사업에서는 다양한 RFID 코드 체계들을 RFID 미들웨어가 처리하지 못하는 문제점과 EPC 코드 체계의 비싼 비용으로 인하여 자체 정의하는 코드를 사용하는 문제 때문에 RFID 사업의 영속성 및 상호 운용성을 이루지 못하고 있어서 국내 RFID 사업의 네트워크화를 하는 데에 어려운 점이 많다. 본 논문은 ALE 미들웨어 기반 다중 RFID 코드 변환 처리 방법을 제안한다. 본 논문의 연구를 통하여 현재 다양한 표준 RFID 코드 체계를 식별하여 ALE 미들웨어가 처리 가능한 URN 코드로 변환할 수 있다. EPC 코드 체계 대신에 상대적으로 비용이 싼 ISO 코드 체계를 사용하여도 ALE 미들웨어에서 처리 가능하여 RFID 시스템을 구현할 수 있다. 또한, 다양한 RFID 코드 체계를 ALE 미들웨어에서 처리할 수 있기 때문에 국내 RFID 사업의 영속성과 상호운용성이 가능하여 RFID 산업의 네트워크화를 이룰 수 있다. 또한 새로운 RFID 코드 체계가 제정되더라도 미들웨어의 수정이나 변경 없이 규칙에 맞는 RFID 코드 변환 정보를 작성 및 추가하여 ALE 미들웨어가 새로운 RFID 코드 체계를 변환할 수 있어서 ALE 미들웨어는 범용성을 가질 수 있다.

# ABSTRACT

## Multi-RFID code conversion process based on ALE

Byun, Ji-Woong
Department of Computer Engineering
Graduate School
Cheju National University

RFID technologies are concerning to one of the core technologies for ubiquitous computing by many research activities currently. RFID middleware which be led to high quality of technologies is suggested by ALE specification of de-facto international group's EPCglobal Inc. But, it is difficult for the RFID business networks in domestic RFID business because of various RFID code system and the expensive fee of EPC code system. In this paper, we propose a multiple RFID code conversion process based on ALE. By this paper, we can identify to various RFID code systems, converse to the URN, and process it in ALE middleware. Also, we can build RFID systems with the cheap fees for ISO code systems in instead of EPC code systems. Using the proposed method, they will accomplish perpetuity and networks of domestic RFID business. According to process the domestic and foreign RFID business in a RFID middleware, the ALE middleware system can be maximize effectively. And even if new RFID code systems is established, ALE middleware can converse and process it without middleware's update.

# 약어표

| | |
|---|---|
| RFID | Radio Frequency IDentification |
| EPC | Electronic Product Code |
| ISO | International Organization for Standardization |
| IEC | International Electrotechnical Commission |
| uID | ubiquitous IDentification |
| IATA | International Air Transport Association |
| IC | Integrated Circuit |
| ALE | Application Level Events |
| UII | Unique Item Identifier |
| TID | Tag Identifier |
| RFU | Reserved For User |
| PC | Protocol Control |
| UMI | User Memory Indicator |
| XI | Extended PC Indicator |
| NSI | Numbering System Identifier |
| AFI | Application Family identifier |
| EAN | European Article Number |
| UCC | Uniform Code Council |
| URN | Uniform Resource Name |
| API | Application Programming Interface |
| DSFID | Data Storage Format Identifier |
| OID | Object IDentifier |
| XML | Extensible Markup Language |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| LOG4J | Log for Java |

# I. 서 론

## 1. 연구 배경

최근 차세대 패러다임인 유비쿼터스 컴퓨팅이 도래하면서 무선 식별 기술, 텔레매틱스 등 다양한 분야에서 활발한 연구가 이루어지고 있다. 유비쿼터스 컴퓨팅의 핵심 기술 중 하나인 RFID는 객체 및 사물에 태그를 부착하고 비접촉 방식으로 자동 인식할 수 있는 기술이다. 초창기에 RFID 리더 및 태그 등과 같은 RFID 하드웨어 분야에 많은 연구가 이루어졌으나, 요즘은 읽혀진 태그 데이터를 가공하여 응용에게 의미 있는 정보로 전달하는 RFID 미들웨어에 대한 연구가 활발히 이루어지고 있다[1][2].

RFID 기술에 관련해 사실상의 국제 표준 단체인 EPCglobal에서 제안하여 현재 표준으로 제정된 EPC(Electronic Product Code) 네트워크는 모든 객체 및 사물에 태그를 부착하여 객체 추적 및 조회, 상품 이동 조회 등을 할 수 있도록 설계되었다. RFID 사용자는 RFID 태그가 부착된 객체를 읽어서 EPC 네트워크를 통하여 객체에 관련된 상세한 정보를 얻을 수 있다[3].

이와 같은 RFID 기술을 적용하기 위하여 우선 RFID 코드 체계를 결정해야 한다. 그 이유는 어떤 코드 체계를 사용하느냐에 따라 태그 및 리더기 사양, 정보시스템 및 서비스 형태가 달라지기 때문이다[4][5].

EPC 네트워크를 구축하려면 EPC 코드 체계를 이용해야 하는데, 이럴 경우에 EPCglobal 단체에 가입하여 회사당 값비싼 EPC 코드 체계 비용을 별도로 납부해야 할 뿐만 아니라, 추가적으로 많은 비용과 시간이 소요된다[6].

한편, 현재 RFID 표준코드 체계는 크게 ISO와 EPC로 구분되고 있다. ISO 코드 체계는 ISO/IEC 기구에서 제정한 코드 체계로 유통 및 물류와 직접 관련되는 ISO/IEC 15459(수송단위), ISO/IEC 10374 등이 있으며, EPC 코드 체계는 SGTIN, SSCC, SGLN 등과 같은 코드 체계가 있다. 그 외에 일본 uID센터에서 제안한 ucode, 국내의 모바일 RFID 포럼에서 제안한 모바일 RFID 코드 체계가

있다. 주요한 코드 체계는 다음 표1과 같다[7].

표 1. 주요한 RFID 코드 체계

| 구분 | 주체 | 비고 |
|---|---|---|
| ISO/IEC 15459 | ISO/IEC | 응용별 정의 |
| 모바일 RFID 코드 | 모바일 RFID 포럼 | 온라인 콘텐츠 및 서비스 |
| EPC | EPCglobal | 유통 물류 |
| ucode | uID 센터 | 유무형 객체 |

이처럼 다양한 코드 체계를 이용하여 국내에서 2004-2006년 동안 RFID 시범사업을 수행하였다. 다음 표 2와 표 3은 2004년도, 2005년도 시범사업에 대한 설명이다. 그러나 표 2, 표 3과 같이 RFID 코드 표준 체계를 따르지 않았거나 서로 다른 RFID 표준 코드 체계를 사용하여 RFID 서비스 네트워크 간의 데이터 교환에 문제를 가져옴에 따라 RFID 서비스 영속성 및 네트워크화에 저해 요인이 되었다. RFID 코드 체계에 따라서 RFID 러더기와 미들웨어 사양을 구축해야 하며, 만약 RFID 코드 체계에 따라서 RFID 네트워크 서비스를 구현되었어도 차후에 RFID 코드 체계가 바뀌거나 표준 RFID 코드 체계가 확장되었으면 새로운 RFID 시스템을 적용하거나 기존 RFID 시스템을 수정해야 하는 문제점을 지니고 있다[7][8][9].

표 2. 2004년도 시범사업기관 및 사업명

| 시범 사업 기관 | 사업명 | 코드 체계 |
|---|---|---|
| 산업자원부 | RFID를 활용한 수출입 국가물류 인프라지원 사업 | EPC 기반 |
| 한국공항공사 | RFID 기반 항공수하물 추적 통제 시스템 구축 | IATA BTIC |
| 해양수산부 | RFID 기반 해운물류 효율화 사업 | 컨테이너 |
| 국방부 | RFID 기술적용 국방탄약관리 시스템 시범구축 | 자체정의 |
| 국립수의검역원 | RFID 이용 수입쇠고기 추적 서비스 | 자체정의 |
| 조달청 | RFID를 이용한 물품관리 시스템 구축 | 자체정의 |

표 3. 2005년도 시범사업기관 및 사업명

| 시범 사업 기관 | 사업명 | 코드 체계 |
|---|---|---|
| 환경부 | RFID 기반 감염성 폐기물 관리시스템 구축 | ISO/IEC 15459 기반 |
| 공군본부 | RFID기술 적용 신무기체계(F-15K) 자산관리시스템 구축 | 자체정의 |
| 통일부 | RFID 기술을 이용한 개성공단 통행 및 전략물자관리 구축 | ISO/IEC 15459 기반 |
| 국립현대미술관 | u-뮤지엄 서비스 | 자체정의 |
| 강원도청 | 대관령 한우 RFID시스템 구축 | 자체정의 |
| 인천광역시 | 동북아 물류중심 실현을 위한 차세대 지식기반 항공화물 RFID 선도사업 | ISO/IEC 15459 기반 |

## 2. 연구 목적 및 방법

본 연구의 목적은 ALE 기반 RFID 미들웨어를 위하여 다양한 RFID 코드 체계를 식별 및 변환할 수 있는 다중 RFID 코드 변환 처리를 하는 것이다. 본 연구에 의해서 다양한 RFID 표준 코드 체계를 ALE 미들웨어에서 처리 가능하기 때문에, EPC 코드 체계 대신에 다른 RFID 코드 체계를 사용할 수 있어서 저렴한 비용으로 RFID 시스템이 구축 가능하다. 그리고 다양한 RFID 코드 체계를 사용하여도 ALE 미들웨어에서는 일괄적으로 처리할 수 있어서 RFID 시스템의 영속성 및 상호 운용성이 가능하여 RFID 사업의 네트워크화를 이룰 수 있고, 본 연구에 의해서 ALE 미들웨어는 현존하고 있는 RFID 표준 코드 체계뿐만 아니라, 향후에 새로운 RFID 표준 코드 체계도 처리할 수 있어서 ALE 기반 RFID 미들웨어는 범용적인 미들웨어로 될 수 있다.

## 3. 논문 구성

2장은 주요한 RFID 코드 체계 및 EPCglobal의 ALE 미들웨어에 대해서 알아보며, 코드 체계에 대한 고려사항에 대해서 알아볼 것이다. 3장에서는 다양한 RFID 코드 체계를 변환할 수 있는 다중 RFID 코드 변환 처리 방법을 제안하고, 4장에서는 본 논문에서 제안하는 다중 RFID 코드 변환 처리 방법을 직접 구현하여 테스트한 결과를 분석한다. 5장에서는 본 논문의 결론과 향후 과제에 대해 논한다.

# II. 관련 연구 및 고려사항

이 장에서는 현존해 있는 다양한 RFID 코드 체계 중 EPC 코드 체계, ISO 코드 체계로 나누어 분석할 것이다. 그리고 각각 코드 체계를 지원하는 RFID Tag 메모리 구조에 분석하며, EPCglobal의 EPC 네트워크 중 하나의 구성요소인 ALE 미들웨어에서 RFID 코드 데이터를 어떻게 처리하는 지에 대해서 기술한다.

## 1. RFID Tag 메모리 구조

RFID Tag는 RFID 코드 데이터를 기록하는 요소로 안테나, 집적회로(IC), 기판의 3개의 구성요소로 이루어진다. RFID 태그의 표준은 ISO 18000-6에 의해 2가지 Type(Type A와 Type B)이 국제표준으로 확정되었다. 그러나 현재 널리 알려진 RFID 국제적 표준단체인 EPCglobal에서는 ISO와 다른 Class 0과 Class1의 태그를 사용해 왔다. 따라서 RFID 표준 태그는 ISO Type A, Type B, EPC Class 0, Class 1의 4가지 태그가 존재하여 하나의 리더기가 이를 완전히 수용할 수 없는 체제로 되어 있다[4].

2004년 12월 EPCglobal 단체에서 EPC Class 1 Generation2(C1 Gen2라 함) 태그를 기술 개발하였다. 이 태그는 기존의 EPC Class 0과 Class 1의 조합하여 만든 것으로 EPCglobal에서는 C1 Gen2 태그에 대한 표준을 ISO 18000-6 Type C로 정식 제안하였다. 그리고 ISO에서는 2006년도 EPCglobal Class 1 Generation 2는 ISO 18000-6 Type C로 국제 표준으로 채택되었다. ISO 18000-6C와 EPC C1 Gen2의 태그 메모리 구조는 다음 그림 1과 같다. 두 태그들의 메모리 구조는 Bank 00, Bank 01, Bank 10, Bank 11까지 4개의 Bank로 구성되어 있으며 동일한 구조를 가지고 있다. 단지 다른 점이 있다면, Bank 01의 명칭이 ISO 18000-6C의 경우 UII(Unique Item Identifier)이라 하고, EPC C1 Gen2 태그 같은 경우에는 EPC로 되어 있을 뿐이다[4][10][11].

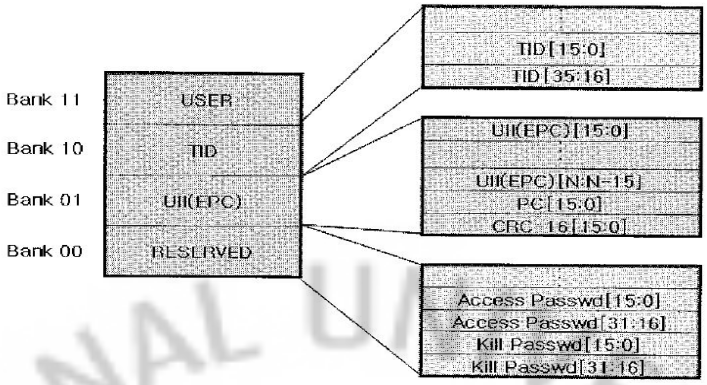



그림 1. ISO 18000-6C 및 EPC C1 Gen2 태그 메모리 구조

Bank 00(Reserved)은 kill 및 access 패스워드를 저장하기 위한 영역이며, Bank 01(UII, EPC)은 아이템(단위 상품, 박스, 팔레트, 컨테이너 등)의 식별코드를 저장하기 위한 영역이다. Bank 10(TID)은 ISO/IEC 15963에서 정의된 태그의 고유 식별자(Tag Identifier)를 저장하며, Bank 11(User)은 사용자가 원하는 임의의 데이터를 저장하기 위한 영역이다.

Bank 01 영역은 크게 태그와 리더의 통신 시 발생 가능한 데이터 오류 검증을 사용하는 CRC-16(Cyclic Redundancy Check), 태그가 리더에 UII(EPC)를 전달하기 위하여 필요한 정보를 기록하는 PC(Protocol Control), 실제 인코딩된 태그 정보가 들어있는 UII(EPC) 데이터로 구분된다. 다음 표 4는 Bank 01의 논리적 메모리 구성요소이다[7][10].

표 4. Bank 01의 논리적 메모리 구성요소

| 구성요소 | | Bank01내의 주소 | 기능 |
|---|---|---|---|
| CRC - 16 | | $00_h$ ~ $0F_h$ | 태그와 리더 사이의 이동정보 에러 검사 |
| PC | length | $10_h$ ~ $14_h$ | '(PC + UII 데이터)의 Word 길이' -1 |
| | RFU | $15_h$ ~ $16_h$ | 향후 사용을 위해 예약 |
| | Toggle(T) | $17_h$ | '0'이면 EPC, '1'이면 Non-EPC 정보가 기록 |
| | AFI | $18_h$ ~ $1F_h$ | 태그 응용 분야 식별 |
| UII(EPC) data | | $20_h$ ~ | 인코딩된 코드 및 관련 정보 저장 |

PC 영역에서 Length는 Bank 01에 사용되는 메모리 크기를 Word 단위로 표현하며, 이때 CRC-16 크기를 제외한 값에서 1을 차감한 값이 length가 된다. PC 영역의 RFU(Reserved for Future Use)는 향후 사용을 예약해 놓은 영역이며 Toggle bit는 EPC와 EPC 이외의 코드 종류 구분을 위하여 사용되는 bit로 최초 리더기가 RFID 코드를 읽어 들여 '0'일 경우에는 UII(EPC) data를 EPC로 해석하고, '1'일 경우에는 ISO/IEC 코드 체계 등 EPC 이외 코드로 해석하게 된다. PC의 마지막 영역인 AFI(Application Family Identifier)는 다양한 타입의 태그가 대량으로 존재하는 RFID 환경에서 희망하는 RFID 태그에 초점을 맞추어 데이터를 처리하기 위하여 사용된다[10].

마지막으로 UII(EPC) data는 실제로 인코딩된 코드 데이터가 저장되는 값이다. UII 데이터와 EPC 데이터는 다음 표 5와 같이 근본적인 차이점을 가지고 있다. UII 데이터가 이러한 구조를 갖는 것은 여러 종류의 ISO 코드 체계를 수용할 수 있도록 설계되었기 때문이다. 표 5에서의 Object ID는 흔히들 OID이라고 부르는데, OID란 통신 단계에서 그 용도를 식별하기 위한 이름을 필요로 하는 정보, 정의 또는 명세의 잘 정의된 요소인 객체를 구분하기 위한 식별체계이다. ISO 18000-6C를 지원하는 태그에 RFID 코드를 기록하기 위해서는 코드 체계별로 ISO/IEC 15962의 정의에 따라 인코딩 방식이 정의되어 다양한 형태를 가지게 된다[4].

표 5. UII 및 EPC 데이터의 구조상 차이점

| UII(EPC) 데이터 구성요소 | 기능 | 존재 여부 | |
|---|---|---|---|
| | | ISO 18000-6C | EPC C1 Gen2 |
| DSFID | 데이터 저장 구조 및 형식 정의 | ○ | X |
| Precursor | Object ID 및 Object 형식 정의 | ○ | X |
| ObjectId Length | Object ID의 byte 길이-1 | ○ | X |
| ObjectId | Object ID 값 | ○ | X |
| Object Length | Object byte 길이 | ○ | X |
| Object | 실제 물품 식별코드 | ○ | ○ |

## 2. EPC 코드 체계

### 1) EPC 코드 체계 개요 및 종류

EPC 코드 체계는 EPCglobal 단체에서 제안한 코드 체계로, 일반적인 형태로 GID(General Identifier)가 있고 EAN.UCC System Identity 타입으로는 SGTIN(Serialized Global Trade Item Number), SSCC(Serial Shipping Container Code), SGLN(Serialized Global Location Number), GRAI(Global Returnable Asset Identifier), GIAI(Global Individual Asset Identifer)이 있으며 마지막으로 DoD 코드 체계가 있다.

EPC 코드 체계는 그 종류에 따라 구조가 상이하나, 한 가지 동일한 구조가 있다. 이는 바로 헤더(Header) 부분인데, 이를 통해 EPC 코드 종류를 파악할 수 있다. 그림 2와 같이 EPC 코드 체계는 헤더 8비트에 따라 나머지 비트의 구조체계가 상이하다[12].

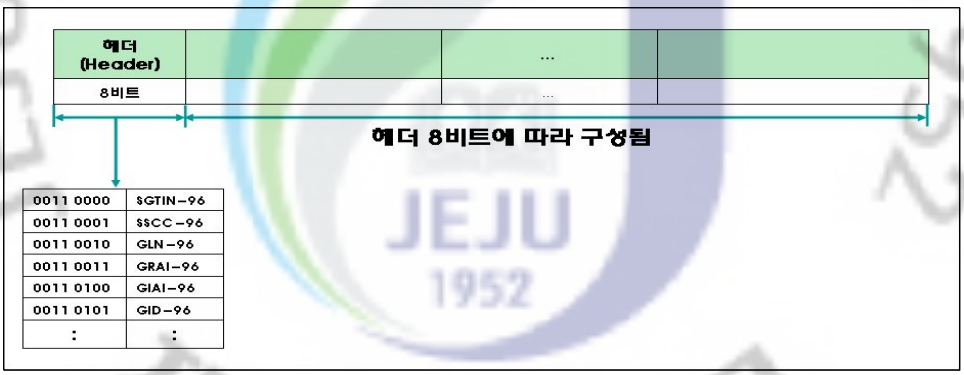



그림 2. EPC 코드 체계 개요

EPC 코드는 총 길이(비트)에 따라 그 구성 형태가 상이하게 되는데, 예를 들어 SGTIN-64, SGTIN-96, SGTIN-198 등은 각각 SGTIN의 코드 체계이며, 64비트, 96비트, 198비트 체계를 갖으며, 또한 헤더 값도 각기 다르다. EPCglobal Tag Data Standard Version 1.3.1에 의하면, 최소 64비트에서 최대 202비트의 코드 체계를 정의하였다. 다음 표 6은 EPC 코드체계의 헤더 값을 정리하였다[13].

표 6. EPC Header에 따른 코드 체계 종류

| Header(비트) | 코드 종류 | Haeder(비트) | 코드 종류 |
|---|---|---|---|
| 0000 1000 | SSCC-64 | 0010 1111 | DoD-96 |
| 0000 1001 | SGLN-64 | 0011 0000 | SGTIN-96 |
| 0000 1010 | GRAI-64 | 0011 0001 | SSCC-96 |
| 0000 1011 | GIAI-64 | 0011 0010 | SGLN-96 |
| 0011 0011 | GRAI-96 | 0011 0110 | SGTIN-198 |
| 0011 0100 | GIAI-96 | 0011 0111 | GRAI-170 |
| 0011 0101 | GID-96 | 0011 1000 | GIAI-202 |
| 0011 1001 | SGLN-195 | 1100 1110 | DoD-64 |
| 1000 0000 ~ 1011 1111 | SGTIN-64 | | |

특히, SGTIN-198, SGLN-195, GRAI-170, GIAI-202와 같이 96비트보다 큰 EPC 코드 체계는 다음 그림 3의 EPC C1 Gen2 태그 메모리 영역에서 Zero Fill 영역을 사용한다. 그림 3에서의 EPC Tag Encoding 영역과 Zero Fill 영역은 앞선 표 4에서 UII(EPC) data에 해당된다. 이러한 Zero Fill 영역에 기록되는 것을 Word Boundary이라고 부르며 오직 '0'으로만 기록되는 Word Boundary 길이는 가변적이다. SGTIN-198은 10비트, SGLN-195는 13비트, GRAI-170과 GIAI-202는 각각 6비트로 Word Boundary가 필요하다. 그럼에 따라 EPC 비트 길이는 SGTIN-198는 원래 EPC 코드 데이터의 길이인 198비트에 Word Boundary 길이인 10비트가 더해져서 208비트가 되고, SGLN-195는 196비트에 13비트를 더해서 208비트, GRAI-170는 170 비트에 6비트가 더해져서 176비트, GIAI-202는 202비트에 6비트가 더해서 208비트가 된다[14].

이러한 Word Boundary가 필요한 이유는 태그에 인코딩된 코드 데이터가 전송되기 위해서는 바이트로 변환되어 전송되는데, SGTIN-198 코드가 바이트로 변환하기 위해서는 코드 데이터에 10 비트인 Word Boundary가 추가되어 208 비트가 되어야 한다. 즉, Word Boundary는 코드 데이터를 바이트로 변환하기 위해

서 코드 데이터의 뒤에 '0'으로 기록되어 덧붙여진 것을 일컫는다.

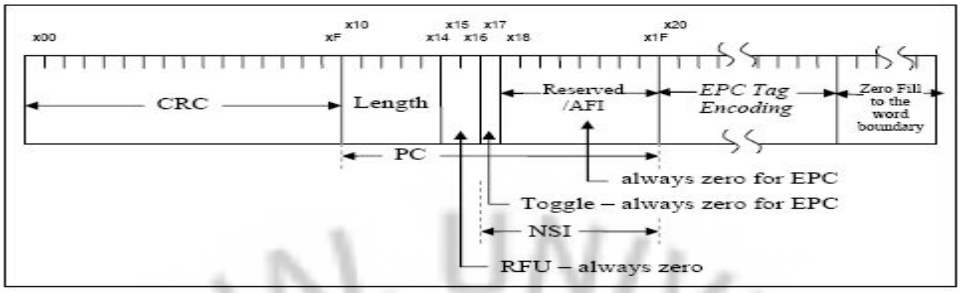



그림 3. EPC 코드 메모리 Bank의 전체 내부

### 2) GID 코드 체계

GID 코드 체계는 기존 스펙이나 식별체계 등에 영향을 받지 않는 새로운 코드 체계이다. GID 코드 체계는 표 7과 같이 헤더, 관리자 번호(General Manager Number), 오브젝트 클래스(Object Class), 일련번호(Serial Number)로 구성된다. 관리자 번호는 오브젝트 클래스 및 일련번호를 관리하고 유지할 책임을 갖는 관리자를 식별한다. 관리자 번호는 EPCglobal 단체에서 할당하고 유일성을 보장받는다. 10진수로 총 268,435,455 개까지 사용 가능하다. 오브젝트 클래스는 관리자 번호를 할당받은 기업 및 조직에서 상품의 종류나 유형을 식별하기 위해 사용되며, 관리자 번호 영역 내에서는 유일성이 보장되어야 한다. 일련번호는 각 오브젝트 클래스 내에서 유일하게 사용되어야 하며 유일한 객체 및 사물을 가리킨다 [13].

표 7. GID-96 구조

| 구분 | Header | General Manager Number | Object Class | Serial Number |
|---|---|---|---|---|
| 비트 크기 | 8 | 28 | 24 | 36 |
| 세부 내용 | $0011\ 0101_2$ (2진수) | 268,435,455 (최대 표현 개수) | 16,777,215 (최대 표현 개수) | 68,719,476,735 (최대 표현 개수) |

### 3) SGTIN 코드 체계

EAN.UCC에서 제정된 GTIN 코드를 기반으로 해서 개개의 물체에 유일한 식별자를 할당하기 위해 제안된 코드 체계이다. 기존의 GTIN 코드 체계는 개체의 종류만을 식별할 수 있기 때문에 RFID 개체 식별에는 적합하지 않았다. 따라서 하나의 물리적 개체를 개별적으로 식별하기 위해 GTIN과 유일 개체 식별자(unique serial number)를 결합하여 Serialized GTIN, 즉 SGTIN이 제안되었다. EPC 길이에 따라 SGTIN-64, SGTIN-96, SGTIN-198 등으로 나뉜다. 표 8은 SGTIN-96 구조이다. SGTIN-96과 SGTIN-198은 SGTIN-64와 다르게, Filter Value와 Partition Value를 사용한다. Filter Value는 물류 유형을 필터링하기 위해 사용되며, Partition Value는 파티션 값에 따라 Company Prefix의 최대 표현 수와 Item Reference의 최대 표현 수가 서로 다르게 나타난다. 만약에 Partition Value에 따라 최대 표현 수가 범위에 벗어난다면 이는 SGTIN 코드 체계가 아니다. SGTIN-198의 Filter Value와 Partition Value는 SGTIN-96과 동일하나, Serial Number에서 20 개 이상의 숫자와 문자의 일련번호를 나타나야 한다. 20 개 이상의 숫자와 문자는 Application Identifier를 사용하는 EAN.UCC-128 바코드 기호에서 일련번호의 모든 범위를 허용한다[13].

표 8. SGTIN-96 구조

| 구분 | Header | Filter Value | Partition | Company Prefix | Item Reference | Serial Number |
|---|---|---|---|---|---|---|
| 비트 크기 | 8 | 3 | 3 | 20-40 | 24-4 | 38 |
| 세부 내용 | 0011 $0000_2$ (2진수) | | | 999,999 ~ 999,999,999,999 (최대 표현 범위) | 9,999,999 ~ 9 (최대표현 범위) | 274,877,906,943 (최대 표현 개수) |

### 4) SSCC 코드 체계

SSCC 코드 체계는 기존의 EAN.UCC 표준에 정의되어 있는 체계이다. GTIN과 달리, SSCC는 개개의 물체에게 식별 번호를 부여할 수 있도록 제안되었다. EPC 코드의 길이에 따라서 SSCC-64, SSCC-96로 나누어진다. 다음 표 9는 SSCC-96 코드 구조를 가리킨다. Serial Reference는 고유 번호를 가지며, Unallocated 필드는 사용되지 않으며, 이 필드는 반드시 0을 포함한다[13].

표 9. SSCC-96 구조

| 구분 | Header | Filter Value | Partition | Company Prefix | Serial Reference | Unallocated |
|---|---|---|---|---|---|---|
| 비트 크기 | 8 | 3 | 3 | 20-40 | 38-18 | 24 |
| 세부 내용 | 0011 $0001_2$ (2진수) | | | 999,999 ~ 999,999,999,999 (최대 표현 범위) | 99,999,999,999 ~ 99,999 (최대표현 범위) | 사용하지 않음 |

### 5) SGLN 코드 체계

SGLN은 개개의 선반(slot)과 같은 개별적인 단위에서 전체 창고와 같은 집합적인 단위까지 포괄적으로 나타낼 수 있도록 제안된 코드 체계이다. SGLN 코드 체계는 코드 길이에 따라 SGLN-64, SGLN-96, SGLN-195로 나뉘며 SGLN-96 구조는 표 10과 같이 Header, Filter Value, Partition, Company Prefix, Location Reference, Serial Number로 구성되어진다. Partition에 따라서, Company Prefix와 Location Reference의 최대 표현 범위가 달라진다. 특히, Serial Number 필드 자리는 확보되어 있지만 EAN.UCC가 GLN을 위해 적절한 방법을 결정될 때까지 사용이 유보되었다[13].

표 10. SSCC-96 구조

| 구분 | Header | Filter Value | Partition | Company Prefix | Location Reference | Serial Number |
|---|---|---|---|---|---|---|
| 비트 크기 | 8 | 3 | 3 | 20-40 | 21-1 | 41 |
| 세부 내용 | 0011 $0010_2$ (2진수) | | | 999,999 ~ 999,999,999,999 (최대 표현 범위) | 999,999 ~ 0 (최대표현 범위) | 2,199,023,255,551(최대 표현 개수) [사용하지 않음] |

### 6) GRAI 코드 체계

GRAI 코드 체계는 기존의 EAN.UCC 표준에 정의되어 있는 체계이며 회수자산관리를 위해서 사용된다. SSCC와 같이 GRAI는 개개의 물체에게 식별 번호를 부여할 수 있도록 제안되었다. 코드의 길이에 따라 GRAI-64, GRAI-96, GRAI-170로 나눠진다. GRAI-96 구조는 표11과 같이 구성되어진다[13].

표 11. SSCC-96 구조

| 구분 | Header | Filter Value | Partition | Company Prefix | Asset Type | Serial Number |
|---|---|---|---|---|---|---|
| 비트 크기 | 8 | 3 | 3 | 20-40 | 24-4 | 38 |
| 세부 내용 | 0011 $0011_2$ (2진수) | | | 999,999 ~ 999,999,999,999 (최대 표현 범위) | 9,999,999 ~ 0 (최대표현 범위) | 274,877,906,941 (최대 표현 개수) |

### 7) GIAI 코드 체계

GIAI 코드 체계는 기존의 EAN.UCC 표준에 정의되어 있는 개별 자산을 위해 사용되는 코드 체계이다. GIAI 코드 체계도 코드 길이에 따라서, GIAI-64, GIAI-96, GIAI-202로 나뉜다. GIAI-96 구조는 다음 표 12와 같이 Header, Filter Value, Partition, Company Prefix, Individual Asset Reference로 구성되어진다.

GIAI-202는 GIAI-96의 Partition을 따르지 않고 독자적으로 Partition을 지니고 있다. GIAI-202의 Individual Asset Reference는 필수적인 숫자와 문자 자산번호를 포함한다. EAN.UCC-128 바코드 기호에서 가능한 전체 일련번호 범위를 허용하는 총 24개 이상의 숫자와 문자 일련번호를 나타낼 수 있다[13].

표 12. GIAI-96 구조

| 구분 | Header | Filter Value | Partition | Company Prefix | Individual Asset Reference |
|---|---|---|---|---|---|
| 비트 크기 | 8 | 3 | 3 | 20-40 | 62-42 |
| 세부 내용 | 0011 $0100_2$ (2진수) | | | 999,999 ~ 999,999,999,999 (최대 표현 범위) | 4,611,686,018,427,387,903 ~ 4,398,046,511,103 (최대 표현 개수) |

## 3. ISO 코드 체계

### 1) ISO 코드 체계 구조

상기의 표 5와 같이 ISO 18000-6C 지원 태그 메모리 구조 중 UII data 영역은 DSFID(Data Structure Formatted IDentifier), Precursor, ObjectId, Object로 이루어져 있다. EPC 코드 체계일 경우는 이와 같은 영역을 사용하지 않는다. DSFID는 데이터가 메모리에 저장되는 방식인 Access Method 2비트와 미정의 영역 1비트, OID 데이터 형태 5비트로 총 8비트로 구성된다. DSFID의 구조는 다음 표 13과 같다. 먼저 Access Method란, RFID 태그 메모리에 데이터가 기록된 방식을 의미하며, Data Format이란, OID(Object IDentifier)의 데이터 형식을 의미한다. OID는 RFID 코드가 기록되는 Object 영역의 ID로 RFID 코드 종류를 의미한다[15][16].

표 13. DSFID 구성표

| 구분 | Access Method | | Reserved | Data Format | | | |
|---|---|---|---|---|---|---|---|
| 비트열 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 |

Precursor는 Offset, Compaction type Code, Relative-OID 총 8비트 3부분으로 구성된다. Offset는 8비트의 Precursor가 추가로 더 있는지, 아니면 8비트에서 끝나는지를 알려주는 지시자 역할을 하며, Compaction type code는 UII data Object 영역에 기록되는 RFID 코드가 어떤 형태로 읽어져야 하는지를 의미한다. 마지막으로 Relative OID는 하부 OID 값을 기록하는 곳이다. Precursor의 상세 구조는 다음 표 14와 같다.

표 14. Precursor 구성표

| 구분 | Offset | Compaction type code | | | Relative-OID | | | |
|---|---|---|---|---|---|---|---|---|
| 비트열 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 |

### 2) ISO 15459 KKR 코드 체계

ISO/IEC 15459 코드 체계는 ISO/IEC JTC1 SC31/WG2 표준화 단체에서 제안되어 이동하는 운송 단위에 할당하기 위해 제안된 코드이며 물류, 유통, 교통 분야에서 국제표준 RFID 코드로 활용되고 있다. ISO/IEC 15459는 알파벳 대문자 [A-Z]와 숫자 [0-9]를 사용할 수 있으며, 발급자 코드(IAC : Issuing Agency Code) 3자리와 하위영역으로 구성된다. 발급자 코드란, ISO/IEC 15459 코드를 발급받아 사용하는 기관을 의미하며, 3자리로 구성된다. 또한, K 영역이 존재하는데 이는 각 국가에서 사용가능한 영역을 의미한다. 즉, K+국가코드 2자리(ISO 3166에서 정한 2자리 알파벳)를 통해 해당 국가를 나타내며, 한국의 국가코드는 KR이기에 ISO/IEC 15459 코드에서는 KKR이 한국을 나타낸다[15][16].

하위 영역은 발급자 코드를 할당받은 기관에서 정의하여 사용한다. 즉, 한국에서 KKR을 사용하도록 할당받은 기관은 하위영역을 정의하여 국내에서 사용할 수 있다. 한국 인터넷 진흥원이 2006년 하반기부터 코드 발급을 시작한 KKR 코

드는 ISO/IEC 15459 국제 표준을 준수하는 국가 코드 체계로 표 15와 같이 구성된다.

표 15. ISO/IEC 15459 KKR 코드 체계

| 구분 | 발급기관 코드 (IAC) | 기관코드 (CC:Company Code) | 구분자 (Prefix) | 객체종류 식별코드 (IC:Item Code) | 객체단위 식별코드 (SC:Serial code) |
|---|---|---|---|---|---|
| 문자 수 | 3 | 3 | 1 | 가변 | 가변 |
| 세부 내용 | KKR | 000 ~ 9ZZ | 향후를 위해 예약됨 | | |
| | | A00 ~ ZZZ | <표 16 참조> | 기관별 자체정의 | 기관별 자체정의 |

표 16. 구분자에 다른 IC의 문자수

| 구분자 (Prefix) | 0 | ~ | 9 | A | B | C | D | E | F | 중략 | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|
| IC의 문자수 | 향후를 위해서 예약됨 | | | 1 | 2 | 3 | 4 | 5 | 6 | 중략 | 26 |

기관 코드는 기관을 유일하게 식별할 수 있다. 기관 코드의 '000 ~ 9ZZ' 영역을 향후 확장을 위하여 예약되었다. 단, 기관코드 'ZZ0 ~ ZZZ'는 자산관리 등 내부 영역을 위한 코드로 사용된다.

위 표 16과 같이 하나의 문자로 이루어진 구분자(prefix)는 객체 종류 식별 코드의 문자수를 정의한다. 객체종류 식별코드는 가변 길이를 가지며 기관별로로 관리하는 개체간의 유일한 식별을 제공하며 구분자를 통하여 기관별 특수 상황에 맞게 길이의 정의가 가능하다.

객체단위 식별코드도 가변 길이를 가지면서 IC간의 유일한 식별을 제공하며 인코딩시 코드 전체 길이를 나타내는 Object Length를 통하여 기관별로 자유롭게 SC의 길이를 정의할 수 있다.

### 3) 모바일 RFID 코드 체계

모바일 RFID 코드 체계는 온라인 콘텐츠의 위치를 찾는데 필요하다. 기존의 EPC 및 ISO 코드는 물류 및 유통 분야의 목적을 위해서 설계되어 모바일 RFID 서비스를 하기에는 적절하지 못하다. 따라서 온라인 콘텐츠 및 서비스의 위치를 주목적으로 찾을 수 있는 모바일 RFID 코드 체계가 필수적이다. 기존의 2차원 바코드와 같이 작은 코드를 사용하여 서비스의 편의를 도와주는 RFID 코드 체계를 모바일 RFID 포럼에서 정의하였다.

표 17과 같이 mCode는 48 비트에서 128 비트까지의 길이를 가질 수 있다. 12 비트의 TLC(Top Level Code)는 최상위 기관에 할당되고, 4 비트의 Class는 TLC를 할당받은 최상위 기관이 하부 기관에 코드를 할당할 때 하부 기관의 규모 및 콘텐츠의 수를 고려하여 여러 구조의 코드를 할당하기 위해 사용된다. CC(Company Code), ICC(Item Category Code), IC(Item Code), SC(Serial Code)는 Class에 따라 각각 다른 길이와 구성을 갖는다[17][18][19].

표 17. mCode 코드 체계

<table>
<tr><th colspan="9">mCode</th><th colspan="2">설명</th></tr>
<tr><th rowspan="2">TLC (12bits)</th><th rowspan="2">Class (4bits)</th><th colspan="7">CC+ICC+IC+SC</th><th rowspan="2">길이 (bits)</th><th rowspan="2">Class 명칭</th></tr>
<tr><th>16bits</th><th>16bits</th><th>16bits</th><th>16bits</th><th>16bits</th><th>16bits</th><th>16bits</th></tr>
<tr><td>$000_H$</td><td colspan="8">예약됨</td><td>N/A</td><td>-</td></tr>
<tr><td rowspan="8">$001_H$~$FFF_H$</td><td>$0_H$</td><td colspan="2">IC</td><td colspan="5"></td><td>48</td><td>A</td></tr>
<tr><td>$1_H$</td><td>CC</td><td colspan="2">IC</td><td colspan="4"></td><td>64</td><td>B</td></tr>
<tr><td>$2_H$~$3_H$</td><td colspan="3">예약됨</td><td colspan="4"></td><td>64</td><td>C</td></tr>
<tr><td>$4_H$</td><td colspan="3">CC</td><td>IC</td><td colspan="2">SC</td><td></td><td>96</td><td>D</td></tr>
<tr><td>$5_H$</td><td>CC</td><td>IC</td><td colspan="3">SC</td><td></td><td>96</td><td>E</td></tr>
<tr><td>$6_H$</td><td>CC</td><td>ICC</td><td colspan="3">IC</td><td></td><td>96</td><td>G</td></tr>
<tr><td>$7_H$~$E_H$</td><td colspan="7">예약됨</td><td>N/A</td><td>-</td></tr>
<tr><td>$F_H$</td><td colspan="7">클래스 확장을 위해서 예약됨</td><td>N/A</td><td>-</td></tr>
<tr><td>$F00_H$~$FFF_H$</td><td colspan="8">다른 코드 구조를 위해서 예약됨</td><td>N/A</td><td>-</td></tr>
</table>

micro-mCode는 표 18과 같이 32 비트의 길이를 가지며, 3 비트 TLC는 최상

위 기관에게 할당된다. TLC를 할당받은 최상위 기관이 독자적으로 서비스를 제공하기 위해 29비트의 코드를 직접 할당한다.

표 18. micro-mCode 코드 체계

| micro-mCode | |
|---|---|
| TLC(3 bits) | IC(29 bits) |
| $000_2$ | 예약됨 |
| $001_2$~$110_2$ | IC |
| $111_2$ | 예약됨 |

다음 표 19는 mini-mCode 코드 체계를 보여주고 있다. mini-mCode는 태그의 작은 메모리를 고려하여 설계된 코드로서 32 비트의 길이를 가진다.

표 19. mini-mCode 코드 체계

| 0~7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TLC | Class | | | | | | | | | | | | | | | | | | | | | | | | 명 |
| $00_H$ | 예약됨 | | | | | | | | | | | | | | | | | | | | | | | | - |
| $01_H$~$EF_H$ | $00_2$ | CC | | | | | | | ICC | | | | | | IC | | | | | | SC | | | | A |
| | $01_2$ | CC | | | | | | | ICC | | | | | | | | IC | | | | | | | | B |
| | $10_2$ | ICC | | | | | | | IC | | | | | | | | SC | | | | | | | | C |
| | $11_2$ | L1 | | | | | | | SC | | | | | | | | | | | | | | | | D |
| $F0_H$~$FF_H$ | 예약됨 | | | | | | | | | | | | | | | | | | | | | | | | - |

## 4. EPCglobal의 ALE 미들웨어

### 1) EPCglobal의 EPC Networks

EPCglobal의 EPC Networks는 유통 및 물류 망에서 상품을 자동 인식을 하고 정보를 공유하는 목적을 가진 국제 기술 표준 네트워크이다. 특정 개체의 식별과

데이터의 저장 및 전달 방법의 정의에 대한 시스템을 제공하는 목적에 초점을 맞추고 있다. EPCglobal의 EPC Networks는 EPC Tag와 Reader, EPC(Electronic Product Code), EPC Middleware(eg, ALE), ONS(Object Name Service), EPC IS(Information Service)과 같이 주요한 구성요소를 갖는다. 기본적으로 유일한 개체를 식별 가능하도록 하는 태그인 EPC가 RFID 리더기에 의해서 읽혀지면, RFID 미들웨어는 수많은 데이터의 의한 EPC Network의 트래픽을 감소시키기 위해 응용이 원하는 정보만 필터링하여 ONS에게 보낸다. ONS은 EPC 데이터를 더 많은 정보가 들어있는 정보를 얻을 수 있게 인터넷 주소로 변환한다[3].

### 2) ALE

EPCglobal에서 표준으로 제안한 RFID 미들웨어의 스펙인 ALE는 EPC 미들웨어에 대한 구체적인 구현 및 특정 소프트웨어 모듈 내에서의 내부 인터페이스를 기술하지 않고 외부 인터페이스만 정의하였다. 데이터를 수집하고 필터링 및 그룹핑하여 비즈니스 로직을 해석하는 이벤트를 생성하는 데에 초점을 둔다. 즉, 원시 EPC 데이터를 획득하는 하부 구조 모듈과 그 데이터를 필터링 및 카운팅하는 구조적 모듈, 그리고 데이터를 사용하는 클라이언트 응용 간의 독립성을 제공한다[20].

그리고 ALE는 Reader Manager 모듈에서 리더 장치에서 들어오는 RFID 코드 데이터를 PML(Physical Mark-up Language) 문서로 바꿔서 처리를 한다. PML 문서로 바꾸는 과정 중에 원시 EPC 데이터를 URN 코드로 변환해야 한다. 즉, ALE 미들웨어는 리더기에서 들어오는 원시적인 RFID 코드 데이터를 필터링 및 그룹핑을 하는데 URN 코드로 바꿔서 처리하고 있다. 따라서 다양한 RFID 코드 데이터가 입력될 경우 URN 코드로 변환해야 한다. 다음 그림 4는 RFID 시스템에서 RFID 코드 데이터가 어떠한 데이터 포맷으로 변화하고 있는지를 나타내고 있다[21].


그림 4. RFID 코드 데이터가 변화되는 과정

## 5. 타 RFID 미들웨어에서 RFID 코드 지원 현황

표 20. 타 RFID 미들웨어에서의 RFID 코드 지원 여부

| RFID 미들웨어 종류 | 기능 | RFID 코드 체계 지원여부 | | | | |
|---|---|---|---|---|---|---|
| | | EPC | ISO | KKR | 모바일 RFID | 새로운 코드 체계 |
| Sun Java System RFID Software (Sun Micro systems) | ● EPC 네트워크의 Savant와 유사한 기능 제공 | ○ | X | X | X | X |
| Sensor Edge Server (Oracle) | ● 센서 기반 서비스 통합 플랫폼<br>● RFID를 포함한 센서기반 서비스 제공 | ○ | X | X | X | X |
| RFID Solution Architecture (Microsoft) | ● Real-Time에 초점<br>● .Net 등의 Microsoft 플랫폼 기반의 RFID Network을 지향 | ○ | ○ | X | X | X |
| ALE-compliant RFID Middleware Software Platform (ETRI) | ● ALE 기반 RFID 미들웨어 | ○ | ○ | X | X | X |
| **제안하는 방법** | ● **ALE 미들웨어 기반 다중 RFID 코드 변환 처리** | **○** | **○** | **○** | **○** | **○** |

표 20은 대표적인 RFID 미들웨어와 본 논문에서 제안하는 방법에서 RFID 코드 체계의 지원 여부를 비교 분석한 것이다. 기존 RFID 미들웨어에서는 ISO/IEC 15459 KKR 코드 체계 및 모바일 RFID 코드 체계를 고려하고 있지 않을 뿐만 아니라, 향후 추가될 수 있는 새로운 RFID 표준 코드 체계에서도 고려하고 있지 않다[21][22].

## 6. 고려 사항

앞서서 살펴본 관련 기술 및 연구사례를 통하여 본 논문에서 핵심적으로 고려해야 할 사항을 도출하였다.

본 논문에서 제안하는 다중 RFID 코드 변환 방법은 다음과 같은 사항을 고려하여 설계 및 구현할 것이다. 첫째, 유비쿼터스 환경 지원 RFID 미들웨어는 다양한 RFID 코드 데이터를 처리할 수 있어야 한다. 앞으로 유비쿼터스 시대에서는 다양한 상황 및 객체 정보가 존재하기 때문에 이를 해결할 수 있는 기능을 가지고 있어야 한다. 둘째, 사실상의 국제 표준인 ALE 기반 RFID 미들웨어에 적합하도록 설계 및 구현할 것이다. 현재 다양한 RFID 미들웨어가 존재하지만 RFID 미들웨어의 사실 표준은 ALE이고 EPC 네트워크의 한 구성요소를 차지하고 있으므로 국제 표준을 준수함으로써 다양한 이득을 얻을 수 있다. 셋째, 국내 RFID 사업의 네트워크화와 상호 운용성을 고려할 것이다. 앞서 연구 배경에서 언급하였듯이, 국내 RFID 사업은 자체 정의한 코드 및 비표준 RFID 코드 체계를 사용하고 있음에 따라 국내 RFID 사업의 네트워크화 및 상호 운용성에 어려움이 많아서 이를 해결할 수 있어야 한다. 셋째, 마지막으로 향후 늘어나는 RFID 코드 체계를 쉽게 추가 및 수정할 수 있도록 확장성을 고려해야 하며, 또한 범용성을 동시에 지니고 있어야 한다.

# III. 제안하는 다중 RFID 코드 변환 처리 방법

## 1. 개요

다음 그림 5는 본 논문에서 제안하는 다중 RFID 코드 변환 처리 방법에 대한 개요도이다. 다양한 RFID 코드 체계가 인코딩된 RFID 태그를 읽은 RFID 리더기는 ALE 기반 RFID 미들웨어에 데이터를 보내게 된다. RFID 미들웨어에 탑재된 MRC(Multi-RFID code Conversion Module)은 다양한 RFID 코드 데이터를 ALE 미들웨어가 인식할 수 있는 URN 코드로 변환하여 전달하게 된다.

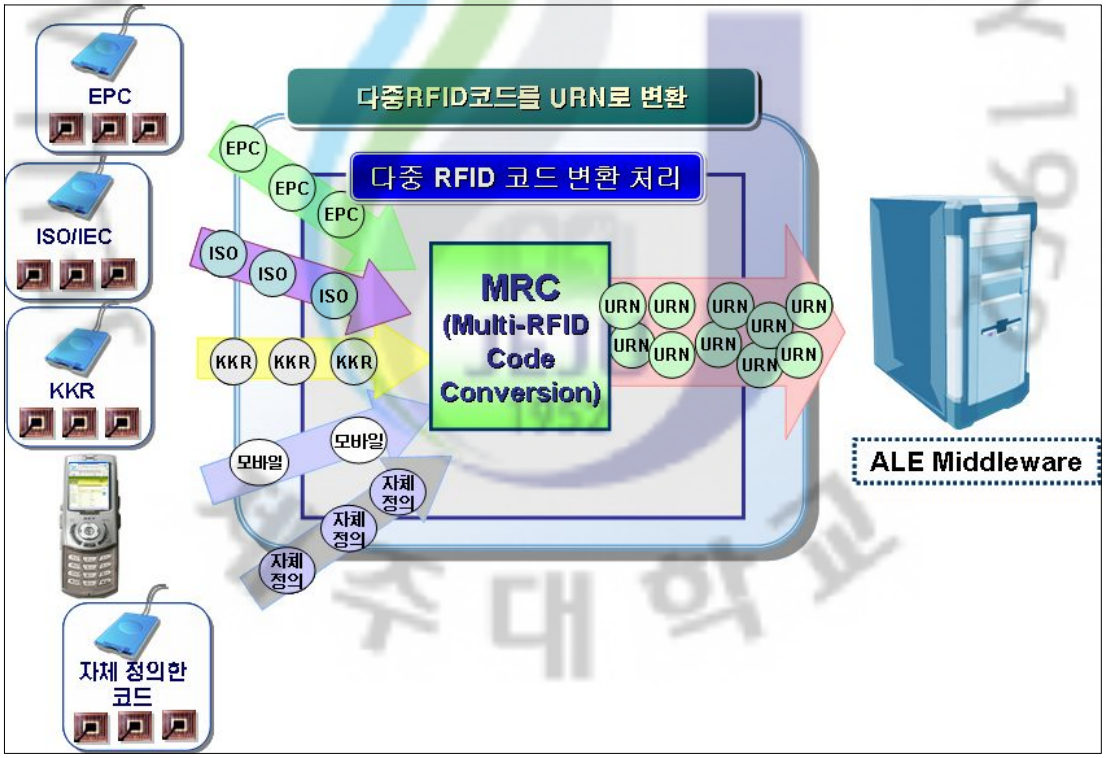



그림 5. 다중 RFID 코드 변환 처리 방법의 개요도

## 2. 모듈 구성도

다음 그림 6은 다중 RFID 코드 변환 처리 방법의 모듈 구성도이다. 다중 RFID 코드 변환 처리 방법, 즉 MRC는 크게 네 가지 모듈로 이루어져 있다.

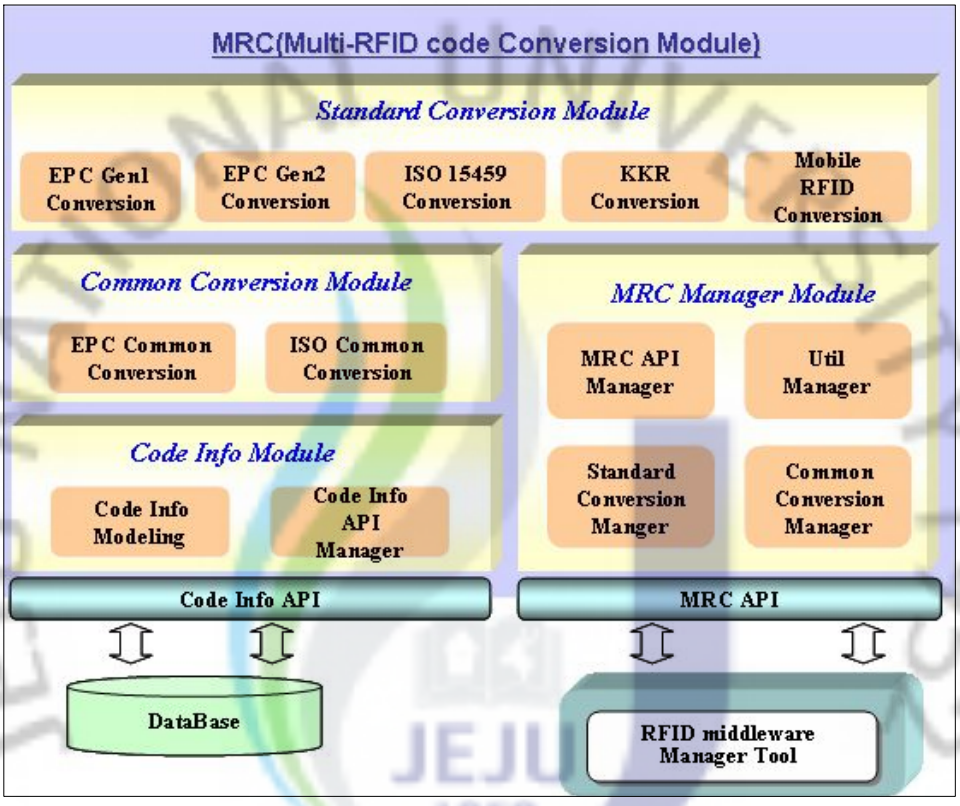



그림 6. 다중 RFID 코드 변환 처리 방법의 모듈 구성도

○ MRC Manager Module

- MRC API Manager

MRC API를 관리하는 관리자이다. ALE 미들웨어는 MRC API를 통해서 다양한 RFID 코드 데이터를 변환하여 처리 가능하고 또한 다양한 URN 코드 형태로 변환 가능하다. 주요한 MRC API 종류는 다음 표 21과 같다.

표 21. MRC API 주요 기능

| MRC API | 기능 |
|---|---|
| getURN<br>( String hexString ) | ● 리더기에서 들어오는 16진수를 인자 값으로 넣으면 URN으로 변환되어 문자열을 반환함 |
| getURNOct<br>( String octString ) | ● 리더기에서 들어오는 8진수를 인자 값으로 넣으면 URN으로 변환되어 문자열을 반환함 |
| translate2RawHex<br>( String urnString ) | ● URN으로 변환된 코드를 인자 값으로 넣으면 EPC Tag Translation Data Standard V1.3에서 규정한 RawHex 코드 형태로 문자열을 반환함 |
| translate2RawDecimal<br>( String urnString ) | ● URN으로 변환된 코드를 인자 값으로 넣으면 EPC Tag Translation Data Standard V1.3에서 규정한 RawDecimal 코드 형태로 문자열을 반환함 |
| translate2Binary<br>( String urnString ) | ● URN으로 변환 코드를 인자 값으로 넣으면 2진수의 문자열로 반환함 |
| translate2RawCode<br>( String urnString ) | ● URN으로 변환 코드를 인자 값으로 넣으면 URN 코드가 아닌 원시적인 코드(KKR 코드 등)로 반환함 |

- Util Manager

Code Info Module, Common Conversion Module, Standard Conversion Module에서 필요한 함수가 들어 있는 관리자이다. 자주 사용되면서 공통적으로 사용되는 함수가 주로 들어 있다.

- Standard Conversion Manager

Standard Conversion Module을 제어하는 관리자이다. MRC API는 이 관리자를 통해서만 Standard Conversion Module을 제어 가능하다.

- Common Conversion Manager

Common Conversion Module을 제어하는 관리자이다. 이 관리자를 통해서만 Common Schema Manager를 제어 가능하다.

○ Code Info Module

- Code Info Modeling

XML으로 저장되어 있는 RFID 코드 데이터 변환 정보를 객체화 시킨다. 객

체화된 코드 변환 정보는 다양한 모듈에서 사용 가능하다.

- Code Info API Manager

Code Info API를 관리하는 관리자이다. RFID 코드 데이터의 변환 정보는 Code Info API를 통해서 DB에 들어 있는 데이터를 추출 및 저장한다. 주요한 Code Info API는 다음 표 22와 같다.

표 22. Code Info API 주요 기능

| Code Info API | 기능 |
|---|---|
| setNewCodeInfo<br>( File xmlFile ) | ● 새로운 코드 정보의 파일인 XML 파일을 DB에 삽입함 |
| getFileNames() | ● 현재 DB에 저장되어 있는 코드 정보의 파일의 이름을 문자열 배열로 반환함 |
| getCodeInfo<br>( String fileName) | ● 인자 값인 파일 이름에 해당되는 XML 파일을 반환함 |
| removeCodeInfo<br>( String fileName) | ● 파일 이름에 해당되는 DB에서 XML 파일을 삭제함 |
| updateCodeInfo<br>( File updateCodeInfo,<br>String fileName ) | ● 수정할 코드 정보의 XML 파일과 수정할 파일 이름을 인자 값으로 받아서 DB에 저장되어 있는 코드 정보를 수정함 |

○ Common Conversion Module

- EPC Common Conversion

EPC 코드 체계에 관련된 알고리즘 중 일반적으로 사용되는 함수를 모아두었다. 예를 들어, 96비트 EPC 코드 체계와는 다르게 Partition 필드를 사용하지 않는 64비트 EPC 코드 체계는 변환 알고리즘이 거의 유사하여 오직 각각 필드의 자리 수만 맞추어 주면 된다. 그럼에 따라 EPC Common Conversion는 일반적인 알고리즘을 이용하여 구현한 함수가 들어있다. EPC Gen1 Conversion이나 EPC Gen2 Conversion에서 EPC Common Conversion의 함수를 사용한다.

- ISO Common Conversion

ISO 코드 체계 중 ISO 코드 체계임을 나타낼 수 있는 알고리즘을 모아 두었다. ISO Common Conversion에 들어 있는 함수를 이용하여 ISO 코드 체계임을 식별한 이후, ISO 15459 코드 체계, KKR 코드 체계, 모바일 RFID 코드 체계로 식별해 나간다.

○ Standard Conversion Module

이 모듈은 EPC Gen1 코드 체계 및 EPC Gen2 코드 체계, 아니면 ISO 15459 코드 체계, KKR 코드 체계, 모바일 RFID 코드 체계 등으로 식별 및 변환 처리하거나 다양한 URN 코드 형태로 변환하는 알고리즘이 구현되어 있다.

## 3. 제안하는 방법의 URN 구조

앞서서 관련 연구 및 기술에서 언급하였듯이 ALE 미들웨어에서는 RFID 코드 데이터를 중복 제거 및 필터링, 그룹핑을 하기 위하여 내부적으로 URN 코드를 사용한다. 그래서 다양한 RFID 코드 데이터가 ALE 미들웨어에서 인식하려면 URN 코드로 변경해야 한다. 기본적으로 EPC 코드 데이터의 URN는 다음과 같다.

표 23. EPC 코드 체계의 URN 종류

| EPC 코드 체계의 URN 종류 | URN 코드 형태의 예 |
|---|---|
| Pure Identity | urn:epc:id:sgtin:0652642.800031.400 |
| EPC 태그 | urn:epc:tag:sgtin-64:3.0652642.800031.400 |
| 원시적 비트열을 위한 URN | urn:epc:raw:64.x00001234DEADBEEF |
| EPC 패턴을 위한 URN | urn:epc:pat:sgtin-64:3.0652642.[1024-2047].* |

EPC 코드 체계에서의 Pure Identity 타입은 기본적인 형식으로 특정한 물리적인 객체를 식별하기 위한 고유 정보만을 담고 있다. EPC 태그는 EPC 코드 체계의 코드 길이에 따라 각각의 코드가 존재한다. 원시적인 비트열을 위한 URN은

비트열을 단순히 16진수 및 10진수로 변환하여 URN으로 변환한다. EPC 패턴을 위한 URN은 ALE 미들웨어의 필터링 및 그룹핑을 하기 위한 EPC 패턴을 위한 URN이다. 이 중에서 ALE 미들웨어에서는 내부적으로 하나의 코드를 정확하게 나타낼 수 있는 EPC 태그를 사용한다. 그럼으로 다른 RFID 코드 체계도 EPC 태그의 형식으로 URN을 작성해야 한다.

한편, 한국 인터넷 진흥원에서 제안한 ISO 15459 KKR 코드 체계의 URN는 "urn:ods:iso-iec:15459:1:KKR.AAA.C.ROM.ABCD"이다. 이 URN은 한국 인터넷 진흥원에서 ODS(Object Discovery Service)를 위하여 제안하였고, 'KKR.AAA.C.ROM.ABCD'와 같이 알파벳 대문자로 코드 데이터를 표현하므로 기존의 URN과 형식 다르기 때문에 ALE 미들웨어에서는 한국 인터넷 진흥원에서 제시하고 있는 URN 코드를 효과적으로 처리하기에 어려운 점이 있다. 이를 해결하기 위하여 본 논문에서는 아래의 예와 같은 URN 형식을 제안하였다. 이 형식은 한국 인터넷 진흥원에서 제안하는 URN 형식을 따르면서도 ALE 미들웨어 내에서도 처리할 수 있다.

urn:ods:15459:1:11634.27457.2.130.1083492

URN 코드 부분 중 헤드(header) 부분인 "urn:ods:15459:1:은 URN 코드이면서 ISO/IEC 15459-1 코드 체계를 의미한다. 나머지 몸체(body) 부분에 해당하는 "11634.27457.2.130.1083492"는 "KKR.AAA.C.ROM.ABCD"를 십진수로 변환한 것이다. 알파벳 대문자를 십진수로 변환한 이유는 표 23의 EPC 태그를 보면 알다시피 ALE 미들웨어는 URN의 몸체 부분을 오직 숫자로만 인식할 수 있기 때문이다.

다음 표 24는 모바일 RFID 포럼에서 제안한 모바일 RFID 코드의 URN 코드 예제이다. 모바일 RFID 포럼에서 제안한 URN 코드도 ODS를 위해서 제안되었지만 ALE 미들웨어에서 사용되더라도 어려운 점이 없을 정도로 EPC Pure Identity의 URN 형식과 유사하다. 그래서 ALE 미들웨어에서 사용될 수 있어 본 논문에서 구현할 때 모바일 RFID 포럼에서 제안한 모바일 RFID 코드의 URN 형식을 그대로 사용하였다.

표 24. 모바일 RFID 코드의 URN 예제

| 모바일 RFID 코드 체계의 URN 종류 | URN 코드 형태의 예 |
|---|---|
| mCode | urn:mcode:id:3602.4.305502420.22136.1 |
| mini-mCode | urn:minimcode:id:1.1.15.48.52 |
| micro-mCode | urn:micromcode:id:4.4660 |

## 4. 코드 변환 정보

본 논문에서 제안하는 다중 RFID 코드 변환 처리 방법을 위하여 표준 RFID 코드 체계를 정의한 코드 변환 정보는 다음 그림 7과 같다. 코드 변환 정보는 코드 변환을 할 때 필요한 코드 체계의 구조적 메타 데이터를 XML 문서로 정의한 것을 일컫는다. 이러한 코드 변환 정보를 XML 문서로 정의함으로써 향후 새로운 코드 체계가 추가되거나 기존의 코드 체계가 수정되더라도 효율적으로 코드 변환 정보를 추가 및 수정할 수 있다. XML의 주요한 속성은 다음 표 25와 같다.

'CodeInfos' elements의 속성에서 type는 어떠한 코드 체계인 지를 기술하고 date는 XML 문서의 생성 날짜를 기술한다. date를 통해서 XML 문서가 수정되었는지의 여부를 판단한다. 'CodeInfo'는 CodeInfos의 type에 기술된 RFID 코드 체계 중 어떠한 코드 체계인 지를 기술한다. 그림 7에서 보면 CodeInfo의 name 속성은 SGTIN-64 코드 체계임을 기술하였다. 'Header'와 'Body'는 각 코드 체계의 헤더 값과 그에 따른 코드 변환 정보를 기술한다. 예를 들어, SGTIN-64 코드 체계는 헤더 값이 '01'이며, 코드 변환 정보에서의 Filter Value는 3, Company Prefix는 14, Item Reference는 20, Serial Number는 25이다. 이러한 정보를 각각 Body 노드에 기술한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<CodeInfos type="EPC" version="1.00" date="2008-10-16T16:05Z"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema" >
        <CodeInfo name="sgtin-64" size="64">
                <Header urnTagHeader="urn:epc:tag:sgtin-64">
                        <Ref name="header">01</Ref>
                </Header>
                <Body>
                        <Refs>
                                <Ref name="FilterValue">3</Ref>
                                <Ref name="Company">14</Ref>
                                <Ref name="Item">20</Ref>
                                <Ref name="Serial">25</Ref>
                        </Refs>
                </Body>
        </CodeInfo>
        <CodeInfo name="sgtin-96" size="96">
                <Header urnTagHeader="urn:epc:tag:sgtin-96">
                        <Ref name="header">
                        00110000
                        </Ref>
                </Header>
                <Body>
                   <Partitions>
                        <Partition value="6">
                .... 생략
```

그림 7. 코드 변환 정보의 XML 문서

표 25. 코드 변환 정보의 XML element 설명

| 속성 | 설명 |
|---|---|
| CodeInfos | 파일의 버전, 파일 생성 날짜, 파일의 타입이 들어 있는 root element |
| CodeInfo | 코드 체계 이름과 코드 길이를 나타내고 있는 element로 CodeInfos의 자식 노드임 |
| Header | 헤더 값을 표시하고 URN 헤더를 가지고 있는 element로 CodeInfo의 자식 노드임 |
| Ref | Header의 자식 노드로 헤더 값을 실제로 가지고 있는 element |
| Body | 코드 변환 정보를 가지고 있는 Header element의 동격 element |
| Partitions | 복수 개의 Partition element를 가지고 있는 element |
| Partition | EPC 코드 체계 중 Partition 값이나 ISO 코드 체계에서 코드 변환 정보를 가지고 있는 Element |
| Fragment | EPC 코드 체계의 Partition 값 중 어떤 종류를 가지고 있는 지 나타내고 있는 element<br>(예, Company, Item) |
| value | Fragment의 자식 element로 실제적인 Partition 값을 가지고 있는 element |
| Refs | 복수 개의 Ref element를 가지고 있는 element로 Partions element와 동격임 |
| Ref | Partition과 동격 element로 코드 변환할 때 주요한 비트열의 자리 수 값을 가지고 있음<br>(예, FilterValue, Serial, WordBoundary) |

## 5. 순차 데이터 흐름도

다음 그림 8은 본 논문에서 제안하는 다중 RFID 코드 변환 처리 방법에 관련한 순차 데이터 흐름도이다.

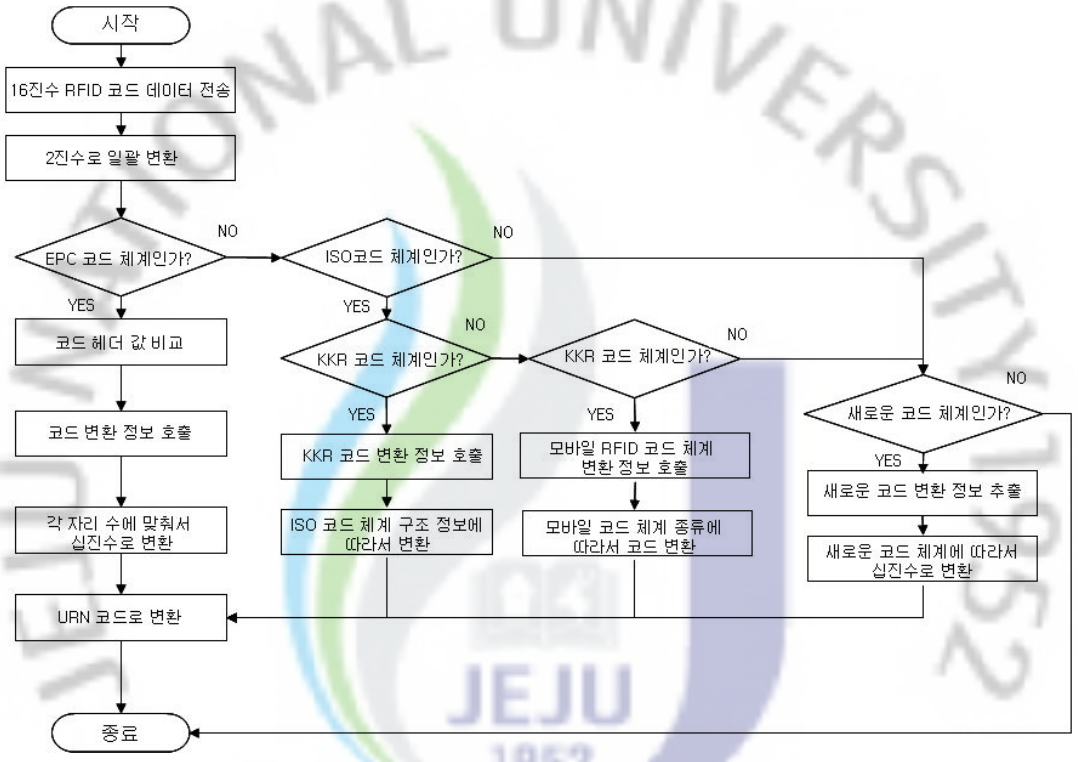



그림 8. 다중 RFID 코드 변환 처리 방법의 순차 데이터 흐름도

RFID 태그에서 데이터를 읽은 RFID 리더기가 ALE 미들웨어에 코드 데이터를 전송할 때, 일반적으로 16진수로 코드 데이터를 변환하여 보낸다. 그러면 전송 받은 데이터는 일단 2진수로 일괄 변환한다. 2진수로 변환된 코드 데이터가 어떠한 코드 체계를 가지고 있는지를 알기 위해서 2진수에서 앞의 8비트를 추출한다. 추출한 8비트를 가지고 코드의 헤더 값과 비교하여 일치하는 코드 체계가 있는 지를 분석한다. 만약에 추출한 8비트 데이터가 EPC 코드 체계 중 어느 하나의 헤더 값과 일치한다면 리더기에서 송신한 코드 데이터는 EPC 코드임을 알

수 있고 ISO 코드 체계의 헤더 값과 일치한다면, 즉, ISO 코드의 DSFID 중 어느 하나와 일치한다면 그것은 ISO 코드 체계이다. EPC 코드 체계일 경우, EPC Gen1과 EPC Gen2 코드 체계가 존재한다. 현재 표준으로 제정되어 있는 코드 체계는 EPC Gen2 코드 체계이지만, 이는 앞서 언급하였듯이 2006년도 표준으로 제정되어서, RFID 시장에서는 EPC Gen1 코드 체계와 EPC Gen2 코드 체계를 둘 다 사용하고 있다. 그럼에 따라 EPC Gen2 코드 체계는 물론, EPC Gen1 코드 체계도 처리할 수 있어야 한다.

EPC 코드 체계는 헤더 값에 따라서 어떠한 코드 체계인지를 확실히 알 수 있기 때문에 그에 맞는 코드 체계의 변환 처리 방법대로 식별 및 변환하면 된다. 그 이후에 URN 형식의 요구 사항이 있으면 URN을 그에 맞게 변환하고 종료가 된다. ISO 코드 체계의 순차 흐름은 EPC 코드 체계의 순차 흐름보다 약간 복잡하다. DSFID 값의 비교로 RFID 코드 데이터가 ISO 코드 체계임을 식별한 이후에 다시 RFID 코드 데이터의 9번째 비트에서부터 8비트를 추출한다. 그 추출한 8비트로 ISO 코드 체계의 Precursor 값과 비교한다. 즉, Precursor 값이 '0011 0001' 혹은 '0011 0100'이 된다면 이것은 ISO 15459-1 코드 체계이거나 ISO 15459-4 코드 체계이다. 그 이유는 ISO 15459 코드 체계가 한 개의 Object만 사용하므로 Precursor의 offset 필드가 '0'으로 기록되며 Object가 5비트 압축을 사용함에 따라 Compaction type code가 '011'로 기록된다. 또한 ISO 15459-1 코드 체계를 나타내는 Relative-OID 값이 '0001'이며, ISO 15459-4 코드 체계를 나타내는 Relative-OID 값은 '0100'이다. ISO 15459-1 코드 체계이나 ISO 15459-4 코드 체계 모두를 KKR 코드 체계는 준수하고 있다.

Length of Object는 Object 바이트 수를 2진수로 기술하는데, Length of Object를 통해서 Object의 총 길이를 알아낸다. 예를 들어, Length of Object를 추출한 값이 '0000 1001'이면 이는 곧 Object 길이가 9바이트가 된다는 것을 의미하고 비트로 변환하면 72비트가 된다. KKR 코드 체계는 Compaction type code에 기록한 대로 1 개의 아스키 문자를 표현하는 데에 5비트 압축을 사용하므로 Object에는 총 14개의 아스키 문자가 있다. 여기에서 남은 2비트는 아스키 문자가 아니라 단지 '0'을 붙인 것이다.

RFID 코드 데이터에서 Object 값을 추출하여 그것을 각각 5비트씩 자른다. 5

비트 압축을 사용함으로 원래의 아스키 문자를 알기 위해서는 5비트 앞에 '01'을 추가하여 7비트로 만들어서 아스키 문자를 구한다. 총 14개의 아스키 문자로 변환한 이후 7번째 문자인 Prefix 값을 얻어서, IC 자리수가 몇 자리인 지를 알아낸다. 마지막으로 IC 값을 제외한 나머지는 SC이다. 아스키 문자를 파악한 이후 이 아스키 문자에 해당되는 비트열을 다시 10진수로 만들어서 URN에 표시한다.

ISO 코드 체계를 준수하는 모바일 RFID 코드 체계는 DSFID 값이 '0000 0001'이다. 모바일 RFID 코드 체계의 mCode, micro-mCode, mini-mCode 모두 DSFID와 Precursor가 같기 때문에, 어떤 종류의 모바일 RFID 코드 체계인 지를 알아내려면 OID를 비교해야 한다. OID Length를 통해서 OID 값을 추출하고 이를 통해서 다음 표 26과 같은 모바일 RFID 코드의 OID 값을 비교한다. OID 값 중 첫 번째, 두 번째, 세 번째 값이 같기 때문에 모바일 RFID 코드 유형을 알아내기 위해서는 네 번째 값을 통해서 알아낸다.

표 26. 모바일 RFID 코드의 OID 값

| 모바일 RFID 코드 체계의 유형 | OID |
|---|---|
| mCode | {0.2.450.1} |
| mini-mCode | {0.2.450.1} |
| micro-mCode | {0.2.450.1} |

OID 값을 통해서 모바일 RFID 코드 체계의 유형을 알아낸 이후에, Object Length를 통하여 Object 값을 얻어낸다. 그리고 각 모바일 RFID 코드 체계의 각 코드 변환 정보를 추출하고 그 코드 변환 정보에 따라서 코드 데이터를 URN으로 변환해야 한다.

헤더 값을 비교해서 EPC 코드 체계도 아니고 ISO 코드 체계가 아닐 경우 새로운 코드 체계가 추가되었는지를 알아본다. 새로운 코드 체계가 추가되었더라면 새로운 코드 체계의 코드 변환 정보에 따라서 URN으로 변환하면 된다. 새로운 코드 체계가 추가되지 않았다면 리더기에서 보내온 코드 데이터는 예외 처리가 되면서 종료된다.

# IV. 구현 및 테스트

## 1. 구현 환경

본 논문에서 제안한 다중 RFID 코드 변환 처리 방법을 구현하기 위한 환경은 다음 표 27과 같다.

표 27. 다중 RFID 코드 변환 처리 방법의 구현 환경

| 구 분 | 하드웨어 | 소프트웨어 | 비 고 |
|---|---|---|---|
| MRC | AMD Athlon(tm) 64 Processor 3000+ 1.81GHz RAM 2.00 GB | Microsoft Windows XP Pro<br>Sun Java SDK 1.5.0_07<br>Eclipse SDK 3.2 | Java 이용 |
| RFID Reader Emulator | AMD Athlon(tm) 64 Processor 3000+ 1.81GHz RAM 2.00 GB | Microsoft Windows XP Pro<br>Sun Java SDK 1.5.0_07 | Java 이용 |
| ALE Middleware | AMD Athlon(tm) 64 Processor 3000+ 1.81GHz RAM 2.00 GB | Microsoft Windows XP Pro<br>Sun Java SDK 1.5.0_07<br>Eclipse SDK 3.2 | Java 이용 |

MRC는 이클립스 3.2를 이용하여 자바로 구현하였다. 다양한 RFID 코드 체계로 인코딩되어 있는 RFID 태그를 읽을 수 있는 리더기와 유사한 성능을 보일 수 있는 RFID 리더 에뮬레이터를 사용하여 본 논문에서 제안한 다중 RFID 코드 변환 처리 방법을 테스트하였다. 에뮬레이터는 EPC Gen1 코드, EPC Gen2 코드, ISO 15459 KKR 코드, mCode, micro-mCode, mini-mCode 등 각각 코드 체계가 변환이 가능한지를 알기 위하여 총 6개의 에뮬레이터를 사용하였다. ALE 미들웨어는 [21] 연구에서 구현하였던 시스템을 사용하였다.

## 2. 패키지 구조 및 클래스에 대한 설명

본 논문에서 제안한 다중 RFID 코드 변환 방법의 전체 패키지 구조는 다음 그림과 같다.

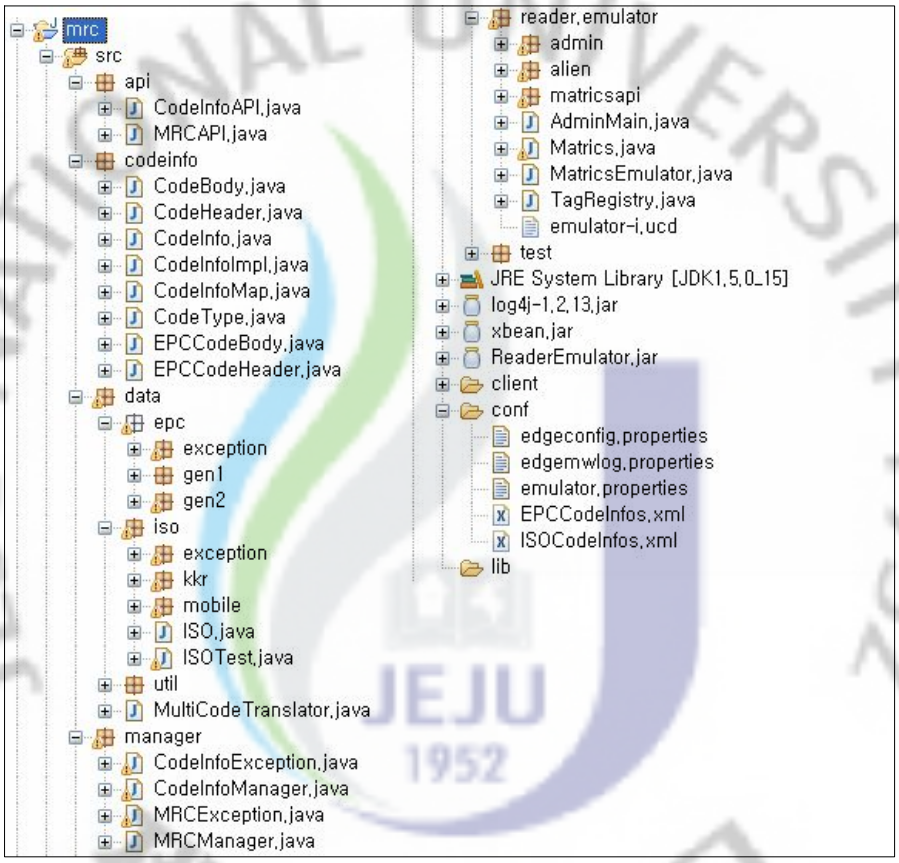



그림 9. 다중 RFID 코드 변환 처리 방법의 패키지 구조

### 1) api 패키지

그림 10은 api 패키지에 포함된 클래스 다이어그램이고 표 28는 각 클래스에 대한 설명이다. api 패키지는 ALE 미들웨어에서 다양한 RFID 코드 변환을 할 수 있도록 API를 제공하는 인터페이스로 이루어져 있다.

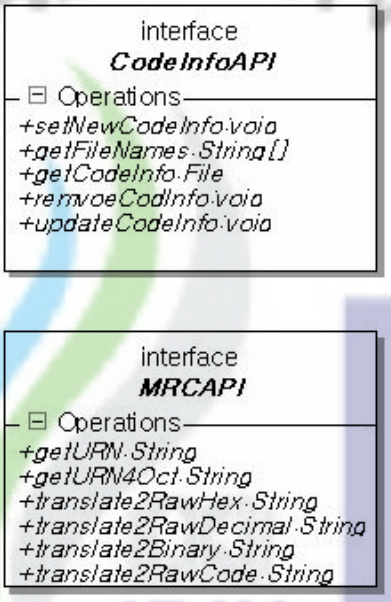



그림 10. api 패키지의 클래스 다이어그램

표 28. api 패키지의 클래스에 대한 설명

| class Name | 설 명 |
|---|---|
| CodeInfoAPI | RFID 코드 변환 정보에 대한 인터페이스 |
| MRCAPI | ALE 미들웨어를 위한 다중 RFID 코드 변환에 대한 인터페이스 |

### 2) codeinfo 패키지

그림 11은 codeinfo 패키지에 포함된 클래스 다이어그램이고 표 29는 각 클래스에 대한 설명이다. codeinfo 패키지는 XML로 이루어진 코드 변환 정보를 객체화 시켜서 메모리에 담을 수 있도록 하였다. 또한 차후에 새로운 RFID 코드 체계에서도 적용 가능하도록 구조화되었다.

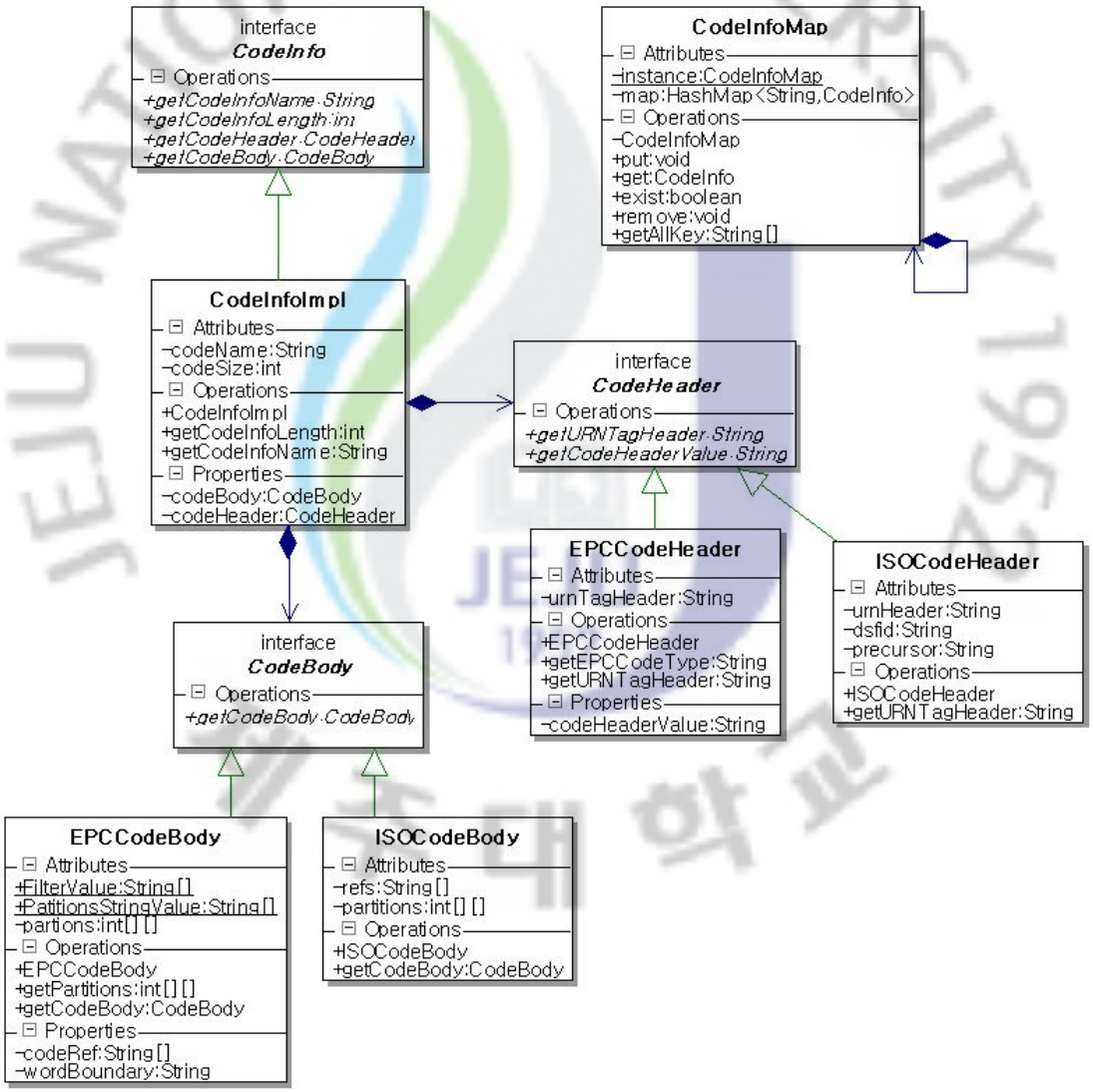



그림 11. codeinfo 패키지의 클래스 다이어그램

표 29. codeinfo 패키지의 클래스에 대한 설명

| class Name | 설 명 |
|---|---|
| CodeInfo | RFID 코드 이름, 코드 길이 등 코드 변환 정보를 담을 수 있는 인터페이스 |
| CodeInfoImpl | CodeInfo 인터페이스를 구현한 클래스로 CodeBody, CodeHeader 인터페이스를 가지고 있는 클래스 |
| CodeHeader | RFID 코드 체계에서 Header 값을 담을 수 있도록 설계된 인터페이스 |
| EPCCodeHeader | CodeHeader 인터페이스를 구현한 클래스로 EPC 코드 체계의 Header 정보가 들어 있는 클래스 |
| ISOCodeHeader | CodeHeader 인터페이스를 구현한 클래스로 ISO 코드 체계의 Header 정보, 즉 DSFID와 Precursor 정보가 들어 있는 클래스 |
| CodeBody | RFID 코드 체계에서 Body 값을 담을 수 있도록 설계한 인터페이스 |
| EPCCodeBody | CodeBody 인터페이스를 구현한 클래스로 EPC 코드 체계의 Body 정보, 즉 Header 정보를 제외한 나머지의 코드 변환 정보가 들어 있는 클래스 |
| ISOCodeBody | CodeBody 인터페이스를 구현한 클래스로 ISO 코드 체계의 Body 정보가 들어 있는 클래스 |
| CodeInfoMap | RFID 코드 체계의 이름을 Key 값으로 하여 CodeInfo 인터페이스의 객체가 들어 있는 HashMap 구조를 지닌 싱글톤 클래스 |

### 3) data 패키지

그림 12는 data 패키지에 포함된 클래스 다이어그램이고 표 30은 각 패키지 및 클래스에 대한 설명이다. data 패키지는 EPC 코드 체계의 변환 알고리즘이 들어 있는 iso 패키지와, ISO 코드 체계의 변환 알고리즘이 구현되어 있는 iso 패키지 및 이러한 패키지에서 사용되는 각종 함수를 정의한 util 패키지, 그리고 마지막으로 MultiCodeTranslator 클래스로 구성되어 있다.

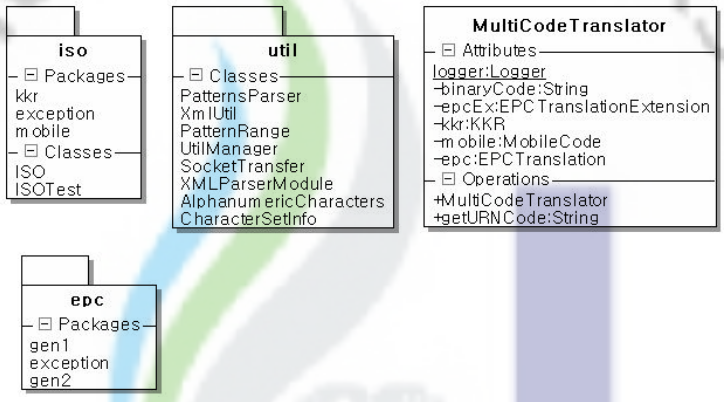



그림 12. data 패키지의 클래스 다이어그램

표 30. data 패키지의 클래스에 대한 설명

| package & class Name | 설 명 |
|---|---|
| epc | EPC Gen1 코드와 EPC Gen2 코드 체계의 변환 알고리즘이 들어 있는 패키지 |
| iso | ISO 코드 구조와 ISO 코드 체계를 준수하는 KKR 코드 변환 알고리즘과 모바일 RFID 코드 변환 알고리즘이 구현되어 있는 패키지 |
| util | epc 패키지와 iso 패키지에서 필요한 함수들을 구현한 클래스가 구현되어 있는 패키지 |
| MultiCodeTranslator | epc 패키지와 iso 패키지를 쓸 수 있는 함수를 구현한 클래스 |

### 4) gen1 패키지

그림 13은 gen1 패키지에 포함된 클래스 다이어그램이고 표 31은 각 패키지 및 클래스에 대한 설명이다. gen1 패키지는 EPC 코드 체계 중 Gen1 코드 변환 알고리즘을 구현한 패키지이다.

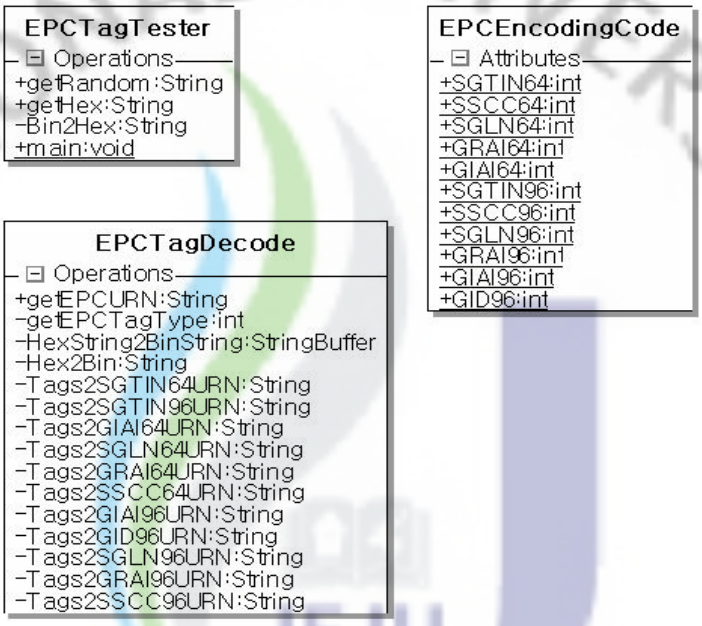



그림 13. gen1 패키지의 클래스 다이어그램

표 31. gen1 패키지의 클래스에 대한 설명

| class Name | 설 명 |
|---|---|
| EPCTagDecode | EPC 코드 체계 중 EPC Gen1의 URN 코드 데이터를 다양한 URN형식으로 변경할 수 있는 클래스 |
| EPCEncodingCode | EPC 코드 체계 중 EPC Gen1 코드 데이터를 URN으로 변환 알고리즘이 들어있는 클래스 |

### 5) gen2 패키지

그림 14는 gen2 패키지에 포함된 클래스 다이어그램이고 표 32는 각 패키지 및 클래스에 대한 설명이다. gen2 패키지는 EPC 코드 체계 중 Gen2 코드 변환 알고리즘을 구현한 패키지이다. EPC Gen2 코드 체계는 EPC Gen1 코드 체계보다 FilterValue, Partition, Wordboundary 등 다양한 계산식이 추가되어 epc.gen1 패키지에 들어있는 클래스보다 많은 클래스가 필요하다.

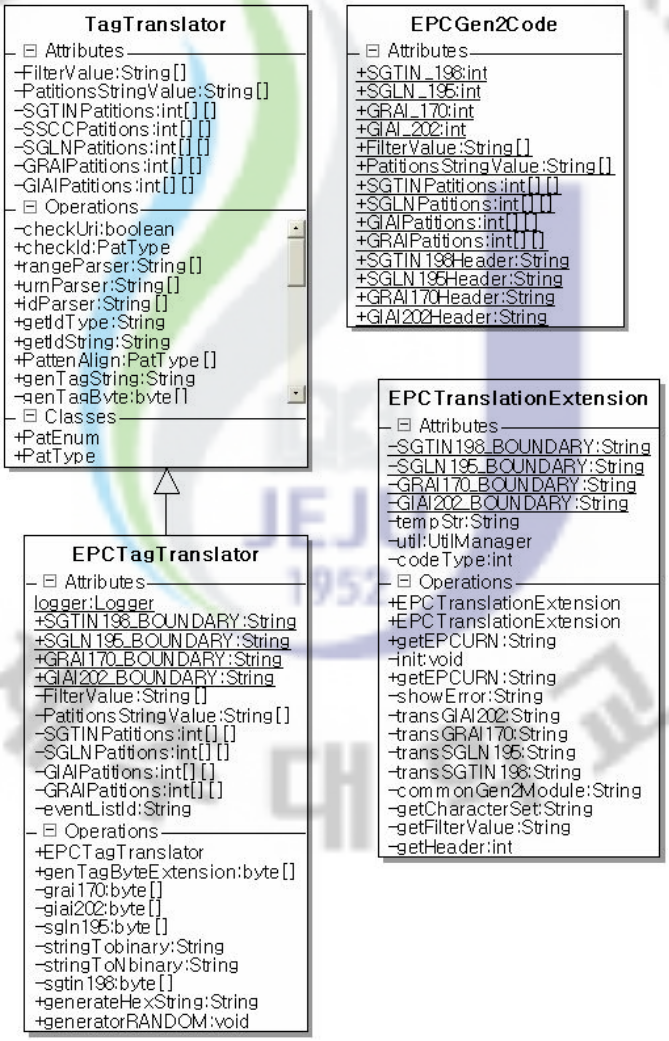



그림 14. gen2 패키지의 클래스 다이어그램

표 32. gen2 패키지의 클래스에 대한 설명

| class Name | 설 명 |
|---|---|
| TagTranslator | CodeInfo 객체에게서 다양한 EPC Gen2 코드 변환 정보를 추출하는 클래스 |
| EPCTagTranlator | URN으로 변경된 코드 데이터를 다양한 URN 형식으로 변환하는 클래스 |
| EPCGen2Code | 96비트 이상의 코드 데이터의 변환 정보를 추출하는 클래스 |
| EPCTarnslationExtension | EPC Gen2 코드 데이터의 변환 알고리즘을 구현한 클래스 |

### 6) iso 패키지

그림 15는 iso 패키지에 포함된 클래스 다이어그램이고 표 33은 각 패키지 및 클래스에 대한 설명이다. iso 패키지에서 크게 ISO 클래스와 모바일 RFID 코드 체계의 변환 알고리즘을 구현한 mobile 패키지, KKR 코드 체계의 변환 알고리즘을 구현한 kkr 패키지, 마지막으로 각 코드 체계에 따른 Exception 클래스가 들어 있는 exception 패키지로 구성되어진다.

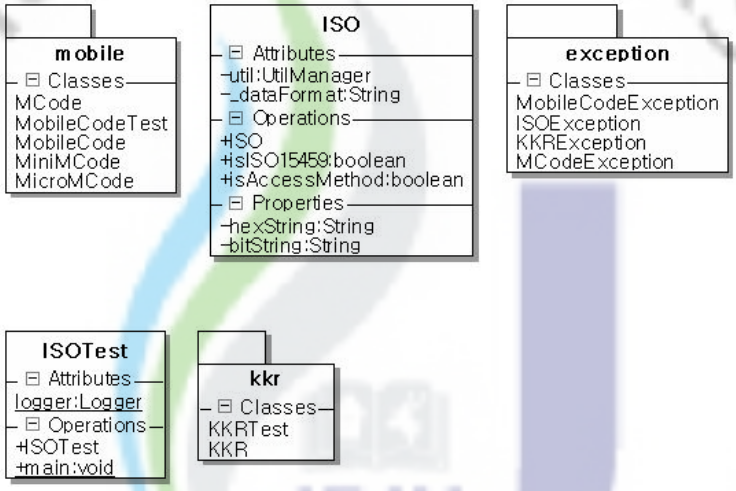



그림 15. iso 패키지의 클래스 다이어그램

표 33. iso 패키지의 클래스에 대한 설명

| package & class Name | 설 명 |
|---|---|
| ISO | 코드 데이터가 ISO 코드 체계인지 확인하는 클래스 |
| mobile | 모바일 RFID 코드 체계의 변환 알고리즘을 구현한 패키지 |
| kkr | ISO 15459 KKR 코드 체계의 변환 알고리즘을 구현한 패키지 |
| exception | ISO 코드, 모바일 RFID 코드, KKR 코드 체계의 변환 알고리즘의 Exception을 모아둔 패키지 |

### 7) mobile 패키지

그림 16은 mobile 패키지에 포함된 클래스 다이어그램이고 표 34는 각 패키지 및 클래스에 대한 설명이다. mobile 패키지는 모바일 RFID 코드 체계의 변환 알고리즘을 구현하였다.

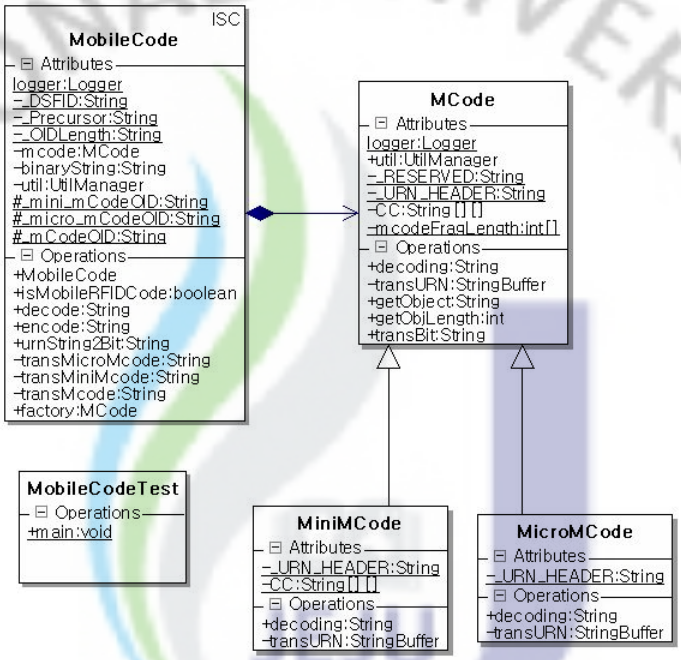



그림 16. mobile 패키지의 클래스 다이어그램

표 34. mobile 패키지의 클래스에 대한 설명

| class Name | 설 명 |
|---|---|
| MobileCode | 모바일 RFID 코드 체계인지를 식별하는 클래스로 입력받은 RFID 코드 데이터의 Header 값을 추출하여, DSFID 및 Precursor 값을 비교하는 클래스 |
| MCode | 모바일 RFID 코드 체계 중 mCode의 변환 알고리즘을 구현한 클래스 |
| MiniMCode | 모바일 RFID 코드 체계 중 mini-mCode의 변환 알고리즘을 구현한 클래스 |
| MicorMCode | 모바일 RFID 코드 체계 중 micro-mCode의 변환 알고리즘을 구현한 클래스 |

### 8) util 패키지

그림 17은 util 패키지에 포함된 클래스 다이어그램이고 표 35는 각 패키지 및 클래스에 대한 설명이다. util 패키지는 모바일 RFID 코드 체계의 변환 알고리즘을 구현하였다.

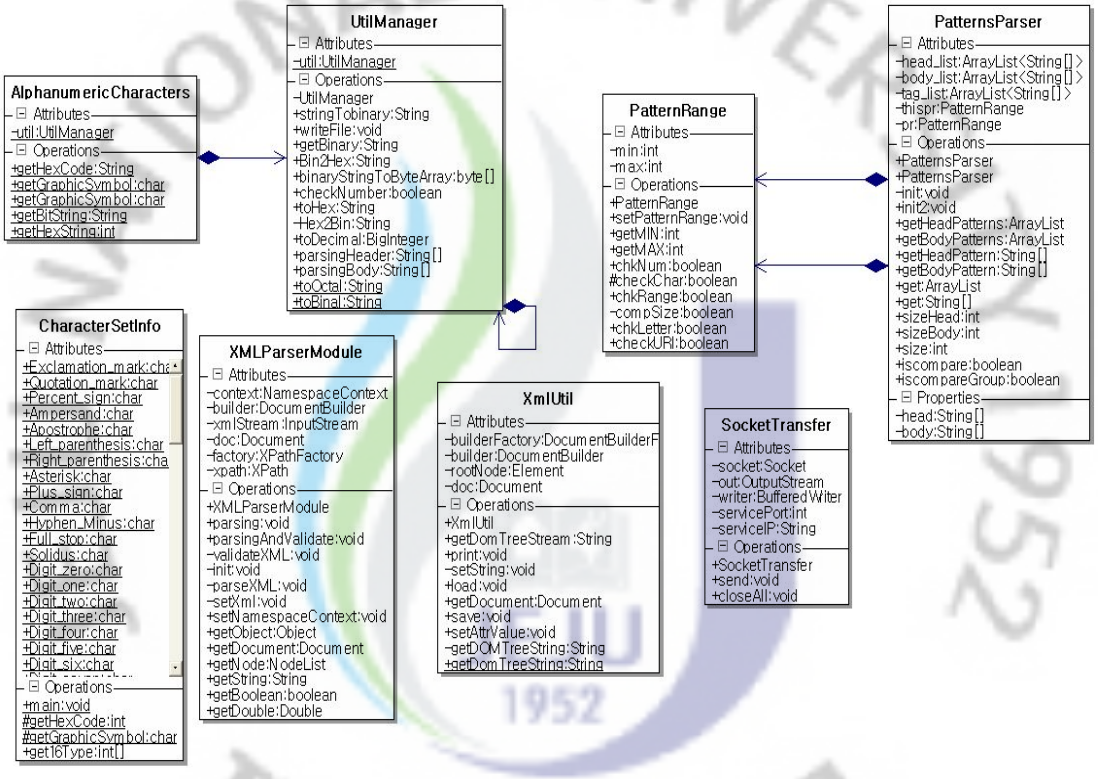



그림 17. util 패키지의 클래스 다이어그램

표 35. util 패키지의 클래스에 대한 설명

| class Name | 설 명 |
|---|---|
| Alphanumeric Characters | 아스키 문자를 2비트로 표현하거나 2비트에서 아스키 문자를 변환할 수 있는 클래스 |
| UtilManager | 다중 RFID 코드 변환 알고리즘에 이용되는 함수를 정의한 클래스 |
| PatternRange | 다양한 URN 코드 형식의 패턴을 파싱하는 클래스 |
| PatternsParser | ALE 미들웨어에서 사용되는 URN 코드 형식을 파싱하는 클래스 |
| CharacterSetInfo | 아스키 문자를 변환하는 데에 필요한 클래스 |
| XMLParserModule | XML 문서를 Dom 방식으로 파싱할 수 있는 함수가 들어 있는 클래스 |
| XmlUtil | XML 문서를 XPath를 이용하여 파싱할 수 있는 함수가 들어 있는 클래스 |

### 9) manager 패키지

그림 18은 manager 패키지에 포함된 클래스 다이어그램이고 표 36은 각 패키지 및 클래스에 대한 설명이다. manager 패키지는 CodeInfo 인터페이스와 MRC 인터페이스를 구현하기 위한 클래스로 이루어져 있다,

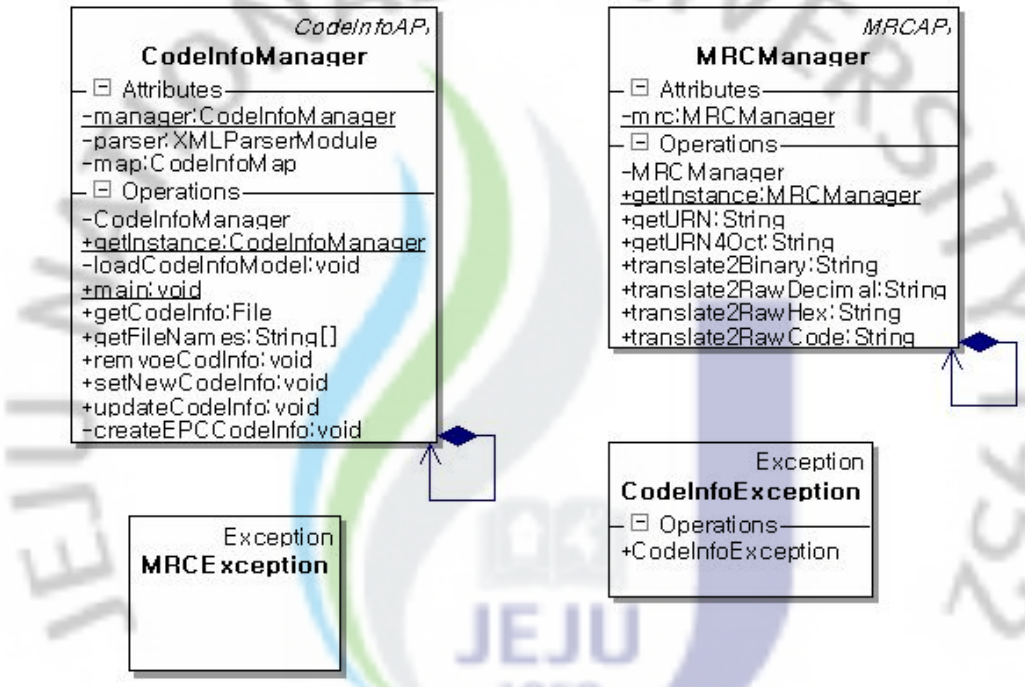



그림 18. manager 패키지의 클래스 다이어그램

표 36. manager 패키지의 클래스에 대한 설명

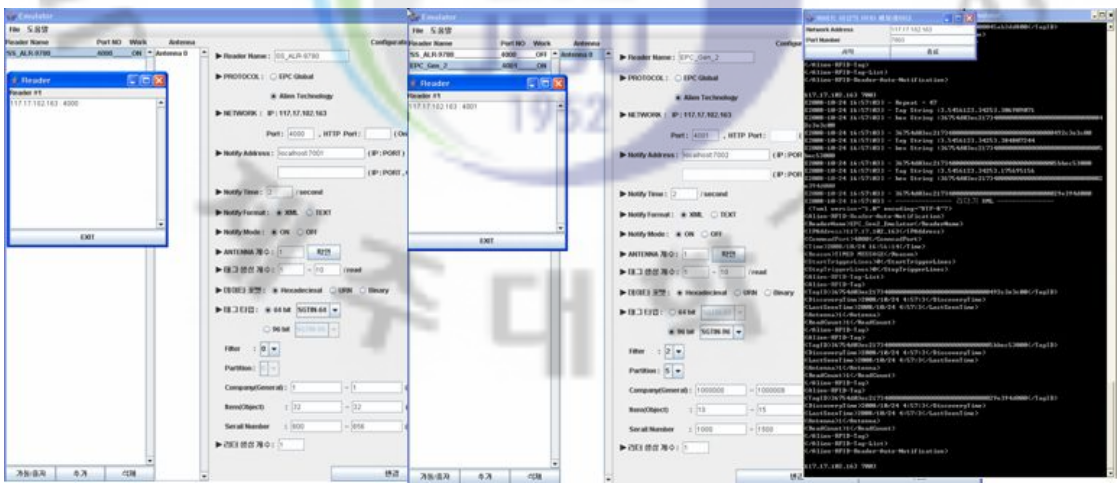
| class Name | 설 명 |
|---|---|
| CodeInfoManager | CodeInfoAPI를 구현한 싱글톤 클래스 |
| MRCManager | MRCAPI를 구현한 싱글톤 클래스 |
| MRCException | MRCManager 클래스에 대한 Exception 클래스 |
| CodeInfoException | CodeInfoManager 클래스에 대한 Exception 클래스 |

### 3. 각 코드 체계별 변환 처리 결과

본 장에서는 EPC Gen1 코드 변환, EPC Gen2 코드 변환, ISO15459 KKR 코드 변환, 모바일 RFID 변환 처리 결과 등에 대해서 다룬다. RFID 에뮬레이터를 통해서 입력된 RFID 코드 데이터를 각 코드 체계의 모듈 별로 처리된 결과를 보여주며 에뮬레이터를 통해서 들어온 데이터에 따른 URN의 결과 값을 표시한다.

#### 1) EPC 코드 변환 결과

ALE 미들웨어에 직접 적용하여 EPC 에뮬레이터를 이용하여 EPC 코드 체계를 변환하였다. 64비트, 96비트, 그리고 96비트 이상 되는 코드 데이터를 코드 길이로 나누어서 각각 변환하였다. 다음 그림 19는 SGTIN-64 코드(a)와 SGTIN-96 코드(b), SGTIN-198 코드(c)를 생성하는 에뮬레이터 모습이며 콘솔창은 에뮬레이터에서 코드 데이터가 생성되는 과정을 log4j를 이용하여 표시한 것이다.

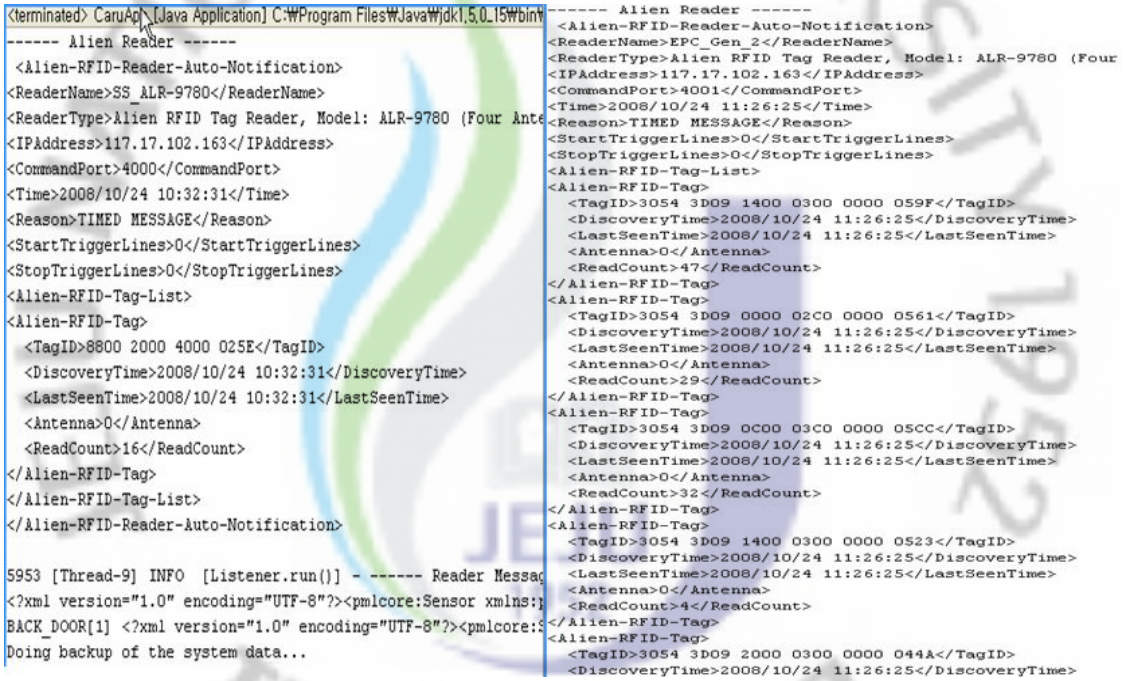


(a) (b) (c)

그림 19. EPC 64비트 및 96비트 코드 데이터, 96비트 이상의 RFID 에뮬레이터

다음 그림 20은 그림 19에서의 (a) 에뮬레이터와 (b) 에뮬레이터가 생성한 EPC 64비트 코드 데이터(a)와 96비트 코드 데이터(b)를 Alien 리더기의 XML 문서의 형태로 TCP/IP 통신을 사용하여 ALE 미들웨어에 전송한 그림이며, 그림 21은 본 논문에서 제안한 방법을 사용하여 ALE 미들웨어에서 PML 문서로 변환한 결과이다.

이와 같이 기존 [21] 연구에서도 EPC 64비트 및 96 비트 코드 데이터를 처리할 수 있으나 본 논문에서 제안한 방법을 적용하여도 기존의 연구와 동일하게 처리하고 있음을 보여주고 있다.




(a) (b)

그림 20. EPC 64비트 및 96비트 코드 데이터의 Alien 리더기의 XML 문서 형태

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema
<pmluid:ID>117.17.102.163</pmluid:ID>
<pmlcore:Observation>
    <pmluid:ID>0</pmluid:ID>
    <pmlcore:DateTime>2008/10/24 11:26:25</pmlcore:DateTime>
    <pmlcore:Tag>
        <pmluid:ID>urn:epc:tag:sgtin-96:2.5.1000005.12.1439</pmluid:ID>
    </pmlcore:Tag>
</pmlcore:Observation>
<pmlcore:Observation>
    <pmluid:ID>0</pmluid:ID>
    <pmlcore:DateTime>2008/10/24 11:26:25</pmlcore:DateTime>
    <pmlcore:Tag>
        <pmluid:ID>urn:epc:tag:sgtin-96:2.5.1000000.11.1377</pmluid:ID>
    </pmlcore:Tag>
</pmlcore:Observation>
<pmlcore:Observation>
    <pmluid:ID>0</pmluid:ID>
    <pmlcore:DateTime>2008/10/24 11:26:25</pmlcore:DateTime>
    <pmlcore:Tag>
        <pmluid:ID>urn:epc:tag:sgtin-96:2.5.1000003.15.1484</pmluid:ID>
    </pmlcore:Tag>
</pmlcore:Observation>
<pmlcore:Observation>
    <pmluid:ID>0</pmluid:ID>
    <pmlcore:DateTime>2008/10/24 11:26:25</pmlcore:DateTime>
    <pmlcore:Tag>
        <pmluid:ID>urn:epc:tag:sgtin-96:2.5.1000005.12.1315</pmluid:ID>
    </pmlcore:Tag>
```

그림 21. EPC 64비트 및 96비트 코드 데이터의 PML 문서

다음 그림 22는 그림 19의 (c) 에뮬레이터가 생성한 EPC 96비트 이상 코드 데이터의 Alien 리더기의 XML 문서 안에 있는 SGTIN-198 코드 데이터를 그림 23과 같이 ALE 미들웨어를 위하여 URN 형식으로 변환하였다. 그림 22에서 TagID의 속성에 있는 16진수 코드 데이터를 그림 23의 PML 문서처럼 URN 코드로 변환된 결과를 보여주고 있다.

```
------ Alien Reader ------
 <?xml version="1.0" encoding="UTF-8"?>
<Alien-RFID-Reader-Auto-Notification>
<ReaderName>EPC_Gen2_Emulator</ReaderName>
<IPAddress>117.17.102.163</IPAddress>
<CommnadPort>4000</CommnadPort>
<Time>2008/10/24 17:12:03</Time>
<Reason>TIMED MESSAGE</Reason>
<StartTriggerLines>0</StartTriggerLines>
<StopTriggerLines>0</StopTriggerLines>
<Alien-RFID-Tag-List>
<Alien-RFID-Tag>
<TagID>36754d03ec2173400000000000000000000000000613dea6400</TagID>
<DiscoveryTime>2008/10/24 5:12:45</DiscoveryTime>
<LastSeenTime>2008/10/24 5:12:45</LastSeenTime>
<Antenna>1</Antenna>
<ReadCount>1</ReadCount>
</Alien-RFID-Tag>
<Alien-RFID-Tag>
<TagID>36754d03ec2173400000000000000000000000006bdc699c00</TagID>
<DiscoveryTime>2008/10/24 5:12:45</DiscoveryTime>
<LastSeenTime>2008/10/24 5:12:45</LastSeenTime>
<Antenna>1</Antenna>
<ReadCount>1</ReadCount>
</Alien-RFID-Tag>
<Alien-RFID-Tag>
<TagID>36754d03ec2173400000000000000000000000039c5e39000</TagID>
<DiscoveryTime>2008/10/24 5:12:45</DiscoveryTime>
<LastSeenTime>2008/10/24 5:12:45</LastSeenTime>
<Antenna>1</Antenna>
<ReadCount>1</ReadCount>
</Alien-RFID-Tag>
</Alien-RFID-Tag-List>
</Alien-RFID-Reader-Auto-Notification>
```

그림 22. EPC 96 비트 이상의 코드 데이터의 Alien 리더기의 XML 메시지 형태

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1 ../SchemaFiles/Inte
<pmluid:ID>117.17.102.163</pmluid:ID>
<pmlcore:Observation>
    <pmluid:ID>1</pmluid:ID>
    <pmlcore:DateTime>2008/10/24 5:12:45</pmlcore:DateTime>
    <pmlcore:Tag><pmluid:ID>urn:epc:tag:sgtin-198:3.5456123.34253.B=u</pmluid:ID></pmlcore:Tag>
</pmlcore:Observation>
<pmlcore:Observation>
    <pmluid:ID>1</pmluid:ID>
    <pmlcore:DateTime>2008/10/24 5:12:45</pmlcore:DateTime>
    <pmlcore:Tag><pmluid:ID>urn:epc:tag:sgtin-198:3.5456123.34253.W 4g</pmluid:ID></pmlcore:Tag>
</pmlcore:Observation>
<pmlcore:Observation>
    <pmluid:ID>1</pmluid:ID>
    <pmlcore:DateTime>2008/10/24 5:12:45</pmlcore:DateTime>
    <pmlcore:Tag><pmluid:ID>urn:epc:tag:sgtin-198:3.5456123.34253.sEqd</pmluid:ID></pmlcore:Tag>
</pmlcore:Observation>
</pmlcore:Sensor>
```

그림 23. SGTIN-198 코드 데이터를 URN 코드로 변환 처리된 PML 문서

SGTIN-198 이외에도, GRAI-170, GIAI-202, SGLN-195 코드 데이터를 URN 코드로 변환되는 테스트 결과를 다음 표 37과 같이 정리하였다. 이와 같이 기존 [21] 연구에서는 처리하지 못하였던 96 비트 이상의 EPC 코드 데이터를 식별 및 변환 가능하다는 것을 보여주고 있다.

표 37. EPC 코드 데이터 변환 테스트 결과

| EPC 코드 체계 종류 | EPC 코드 데이터 | URN |
|---|---|---|
| SGTIN-198 | 363a2bfc002dc14000000000000000476ec61b060ca9c362c000 | urn:epc:tag:sgtin-198:1.569328.46853.G71C002SC10 |
| GRAI-170 | 3734b74695c990400000000000001c3860c59306ccc0 | urn:epc:tag:grai-170:1.3002789.468545.88012063 |
| GIAI-202 | 3834b74694014275870c19b668e16b060cc2849880ddbb360e00 | urn:epc:tag:giai-202:1.3002789. PNX803648-003 PID 77308 |
| SGLN-195 | 3934b74694225e000006edd9b070b5ab46ab58b168c1cb86a000 | urn:epc:tag:sgln-195:1.3002789.4399.77308-545-1140985 |

### 2) KKR 코드 변환 결과

다음 그림 24는 RFID 리더기와 유사한 RFID 에뮬레이터를 통해서 ISO 15459 KKR 코드 데이터를 alien 메시지 타입인 XML 문서로 만들어서 ALE 미들웨어에 전송한 결과이다. 그림 25는 ALE 미들웨어에서 내부적으로 사용되는 PML 문서를 메모장으로 옮겨서 나타낸 것이다. 그림 24에 있는 TagID element에 있는 '0534095AE410882110863210'이 다음 그림 25와 같이 PML 문서의 pmlcore:Tag element에 있는 'urn:ods:15459:1:11634.1058.1.1.69307524'처럼 URN 코드 형식으로 바뀐다.


그림 24. RFID 에뮬레이터로 생성한 KKR 코드 데이터


그림 25. URN 코드로 변환된 KKR 코드 데이터

다음 그림 26은 16진수인 KKR 코드 데이터가 어떻게 URN 코드로 변환되는지의 과정을 보여주고 있다. 입력받은 16진수의 코드 데이터가 ISO 15459 KKR 코드 체계가 맞는지를 일단 식별한 이후에, 어떠한 압축 비트를 사용하였는지를 코드 데이터를 통해서 알아내며, 그에 따라서 Object Length가 얼마인지를 확인한다. Object Length를 통해서 Object를 얻어낸 후에 KKR 코드 체계의 변환 정보에 맞게 2진수로 변환하고 2진수는 ALE 미들웨어에서 사용되는 URN 코드 형식으로 변환된다.


그림 26. KKR 코드 데이터의 변환 모습

다음 그림 27은 URN 코드 형식으로 변환된 KKR 코드 데이터를 응용이 원하는 다양한 URN 코드 형식으로 변환한 것을 보여주고 있다. 그림 27에서 'NIDA URN'은 한국인터넷진흥원에서 제안한 URN 코드 형식을 의미하며, 'URN for ALE Middleware'는 본 논문에서 제안하는 URN 코드 형식이며, 'EPC RawDecimal'과 'EPC RawHex'는 ALE 미들웨어의 스펙에서 정의한 EPC의

URN 코드 형식 중 코드 데이터의 10진수 값과 16진수 값을 일컫는다. 이와 같이 다양한 URN 형식을 제공함에 따라 응용이 원하는 URN 코드 형식으로 전송이 가능하여 미들웨어와 응용 간의 연계가 용이하다.

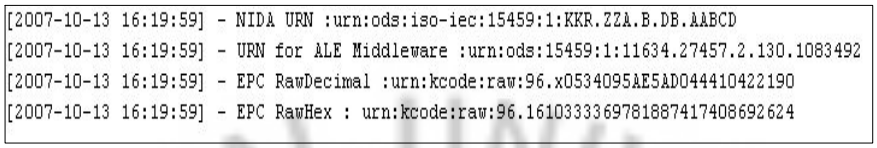



그림 27. KKR 코드 데이터의 다양한 URN 코드 형식

ISO 15459 KKR 코드 체계에 대한 테스트 결과를 다음 표 38과 같이 정리하였다. 표 38을 살펴보면 'KKR'에 해당되는 부분이 '11634'의 10진수로 똑같다는 것을 알 수 있다. 다음 10진수는 기관코드에 해당되며, 한 자리의 10진수는 구분자에 해당되고, 다음 영역은 IC와 SC에 해당된다.

표 38. KKR 코드 데이터 변환 테스트 결과

| 16 진수의 KKR 코드 데이터 | URN |
|---|---|
| 0534095AE5AD044410422190 | urn:ods:15459:1:11634.27457.2.130.1083492 |
| 0534095AE4318C94A483AD68 | urn:ods:15459:1:11634.3171.4.676417.27482 |
| 0534095AE410882110863210 | urn:ods:15459:1:11634.1058.1.1.69307524 |
| 0534095AE4119058C0443214 | urn:ods:15459:1:11634.1124.2.792.1117317 |
| 0534095AE4110C2108864298 | urn:ods:15459:1:11634.1091.1.1.35754150 |

### 3) 모바일 RFID 코드 변환 결과

다음 그림 28은 위의 KKR 코드 데이터 변환 결과와 비슷하게 RFID 리더기와 유사한 RFID 에뮬레이터를 통해서 모바일 RFID 코드 데이터를 ALE 미들웨어에 전송한 결과다.


그림 28. RFID 에뮬레이터로 생성한 모바일 RFID 코드 데이터


그림 29. URN 코드로 변환된 모바일 RFID 코드 데이터

그림 29는 ALE 미들웨어에서 내부적으로 사용되는 PML 문서를 나타내며, PML 문서 안에는 URN 코드로 변환된 모바일 RFID 코드 데이터를 보여주고 있다. 그림 28에 있는 16진수의 모바일 RFID 코드인 '10FC3028342010CE12412341234123456789012'를 그림 29와 같이 URN 코드 형식으로 바뀐 결과를 보여주고 있다.

모바일 RFID 코드 변환의 테스트 결과를 보여주기 위하여 log4j를 이용하여

변환 과정을 표시하였다. mCode와 mini-mCode는 TLC와 Class에 따라서, IC, CC, SC, ICC가 존재하지 않거나 길이가 가변적이라는 것을 고려하였다. 그림 30은 코드 변환을 수행한 테스트 과정의 일부분이다. 모바일 코드가 16진수로 입력될 경우 일괄적으로 2진수로 변환되고 모바일 RFID 코드 체계의 DSFID 값을 통하여 모바일 RFID 코드 체계임을 식별한 이후에, OID 길이를 구한다. OID 길이를 통하여 OID 값을 추출한 다음, OID에 따라서 코드 데이터가 mCode, mini-mCode, micromCode 중에 어떠한 종류인지를 식별한다. 식별된 모바일 RFID 코드 체계의 코드 변환 정보를 추출하고 코드 변환 정보에 따라서 2진수로 된 코드 데이터를 URN으로 변환한다. 그림 30의  첫 번째 예를 보자면 코드 데이터가 mCode이고 Object가 'E12412341234123456789012'일 경우, TLC가 'E12'이며 Class는 '4'가 되고 CC는 '305402420', IC는 '4660', SC는 '1450741778'이 된다. 따라서 URN 코드는 'urn:mcode:id:3602.4.305402420.4660.1450741778'이 된다.

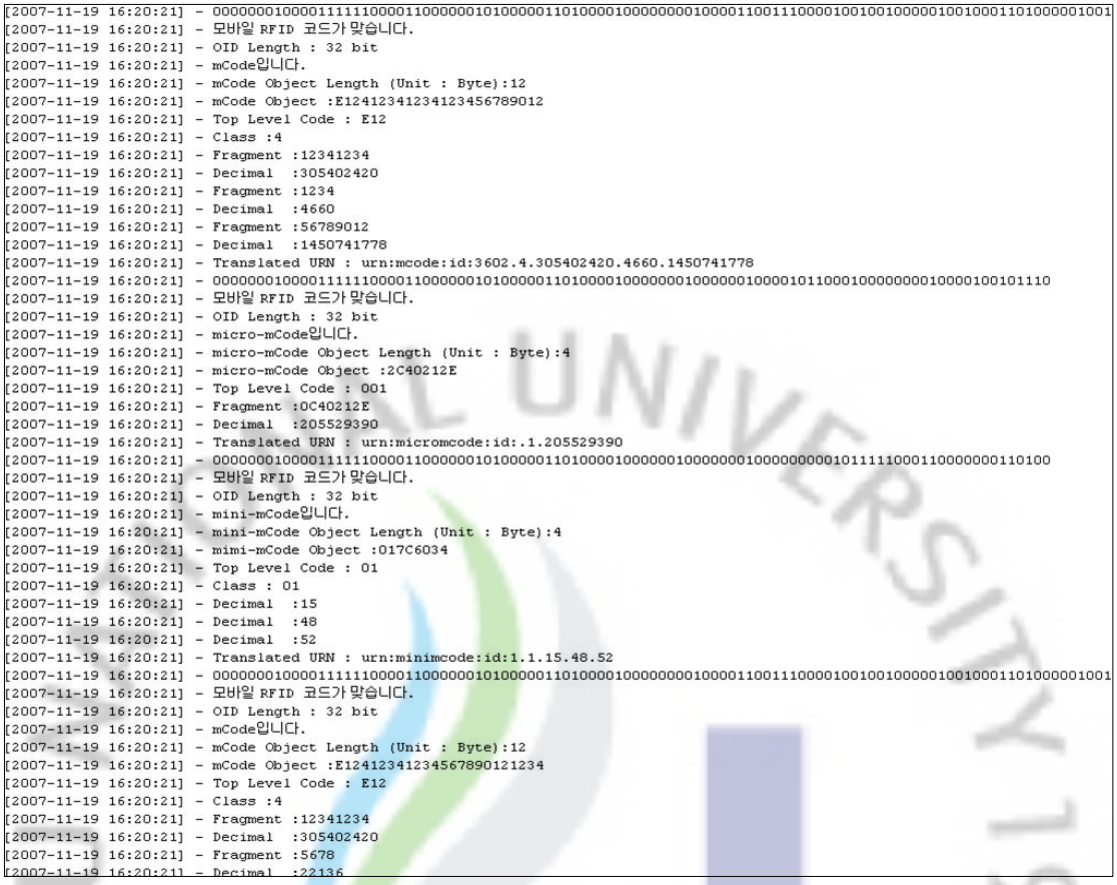



그림 30. 모바일 RFID 코드 변환 과정

그림 30과 같이 모바일 RFID 코드 데이터가 URN 코드로 변환되는 테스트 결과를 정리한 것이 다음 표 39와 같다. 모바일 RFID 코드 데이터가 mCode, mini-mCode, micro-mCode의 코드 변환 정보에 따라서 각각 변환된 URN 코드 형식의 결과 데이터를 볼 수 있다. 16진수의 모바일 RFID 코드 데이터를 살펴보면 처음 16진수의 몇 자리는 동일하다는 것을 볼 수 있다. 그 이유는 mCode, mini-mCode, micro-mCode는 모바일 RFID 코드 체계에 속하기 때문이다. 즉, 모바일 RFID 코드 체계의 DSFID와 Precursor 값이 동일하고 OID 값에 따라서 어떠한 코드 체계인지를 파악하기 때문이다.

표 39. 모바일 RFID 코드 데이터 변환 테스트 결과

| 종류 | 16진수의 모바일 RFID 코드데이터 | URN |
|---|---|---|
| mCode | 010FC3028342010CE12412341234123456789012 | urn:mcode:id:3602.4.305402420.4660.1450741778 |
| | 010FC3028342010C00261234567890789012 3456 | urn:mcode:id:2.6.4660.22136.158847487587414 |
| | 010FC3028342010CE12412341234567890121234 | urn:mcode:id:3602.4.305402420.22136.2417103412 |
| | 010FC3028342010CE121001200001234 | urn:mcode:id:3602.1.18.4660 |
| micro-mCode | 010FC302834202042C40212E | urn:micromcode:id:1.205529390 |
| | 010FC3028342020480001234 | urn:micromcode:id:4.4660 |
| | 010FC302834202049009125C | urn:micromcode:id:4.269029980 |
| | 010FC30283420204502F1890 | urn:micromcode:id:2.271521936 |
| mini-mCode | 010FC30283420404017C6034 | urn:minimcode:id:1.1.15.48.52 |
| | 010FC30283420404027F6001 | urn:minimcode:id:2.1.15.432.1 |
| | 010FC30283420404032C6034 | urn:minimcode:id:3.0.11.6.0.52 |
| | 010FC30283420404047C6F78 | urn:minimcode:id:4.1.15.55.376 |

## 4. 다중 RFID 코드 변환 처리의 성능 평가

다음 그림 31은 기존의 ALE 미들웨어[21]이 동작되었을 때의 Summary, 그림 32는 메모리 사용량이다. 그림 33은 본 논문에서 제안하는 방법을 ALE 미들웨어에 적용하였을 때의 Summary, 그림 34는 메모리 사용량을 표시하고 있다. 그림 31과 그림 33을 살펴보면 그 차이가 미비하며, 그림 32와 그림 34에 표시된 메모리 사용량도 큰 차이를 나타나지 않는다. 이와 같이 본 논문에서 제안하는 방법을 이용할 경우, 기존의 ALE 미들웨어의 수정 및 변경 없이 다양한 RFID 코드 체계를 변환 처리할 수 있으며, 기존의 ALE 미들웨어의 리소스 점유율에도 큰 영향을 주지 않는다.

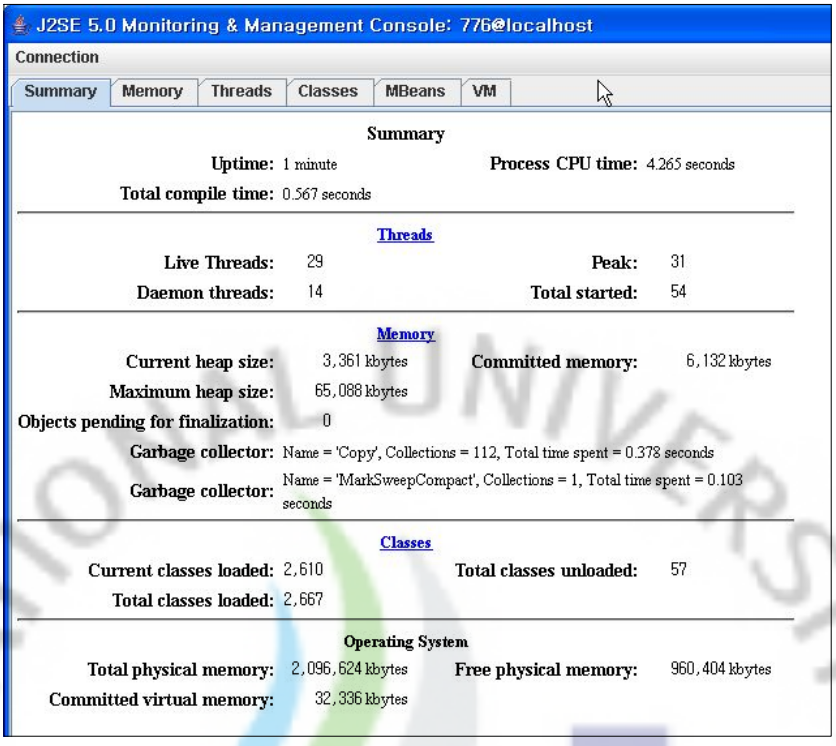



그림 31. 기존 ALE 미들웨어의 Summary

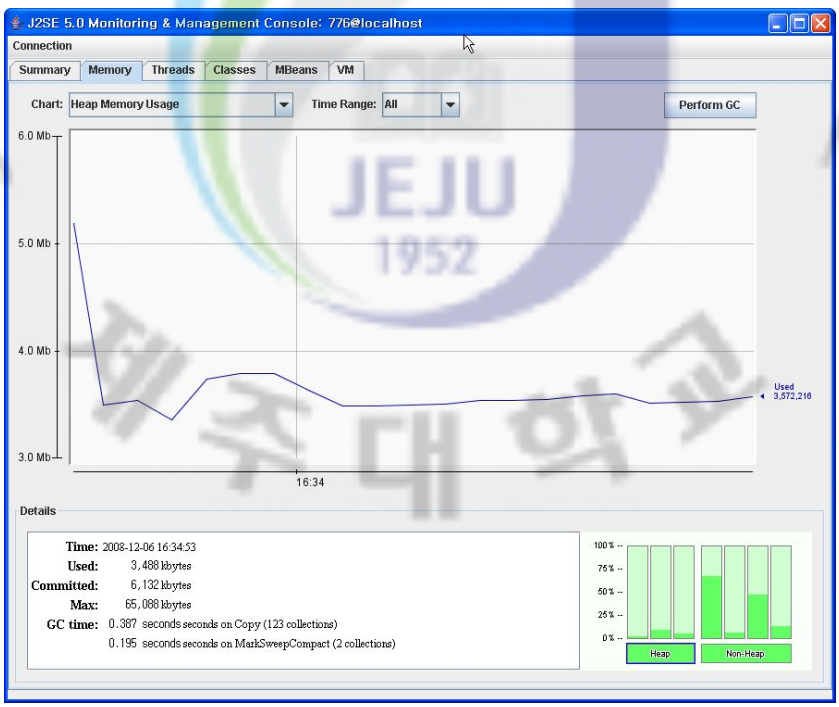



그림 32. 기존 ALE 미들웨어의 메모리 사용량

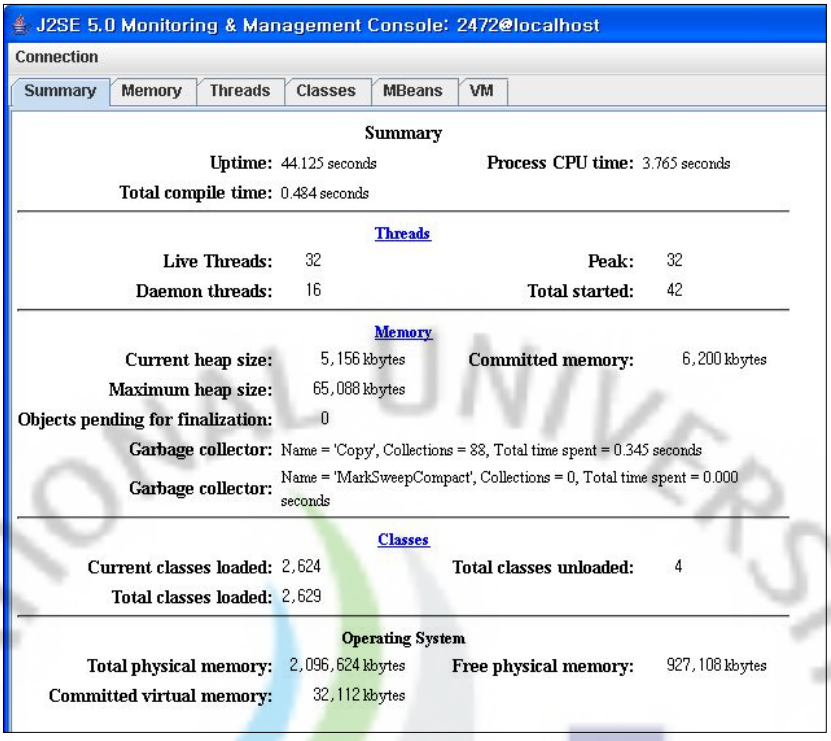



그림 33. 제안하는 방법을 적용한 ALE 미들웨어의 Summary

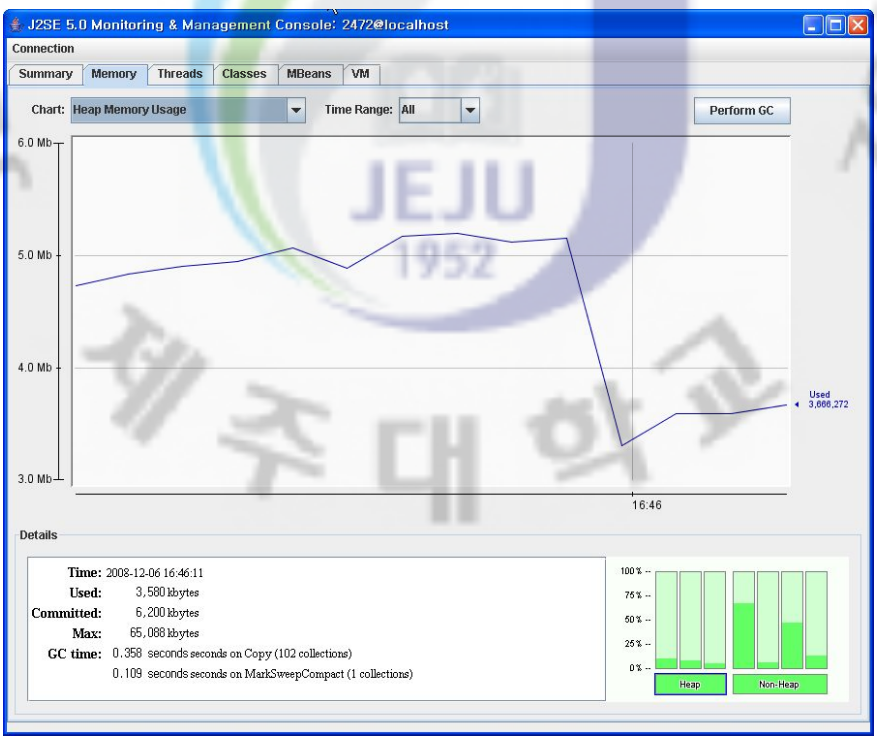



그림 34. 제안하는 방법을 적용한 ALE 미들웨어의 메모리 사용량

# V. 결론 및 토의

본 논문에서는 ALE 미들웨어 기반 다중 RFID 코드 변환 처리 방법을 제안하였고, 제안한 방법을 이용하여 설계 및 구현하였다. 다양한 RFID 코드 체계를 식별하고 식별된 코드 데이터를 URN 코드로 변환하며, 또한 URN 코드를 다양한 URN 코드 형식으로 변환할 수 있게 하였다. RFID 코드 변환 및 처리를 하기 위하여 RFID 코드 체계의 코드 변환에 관련된 구조 정보를 XML 문서로 만들었고 XML 문서를 파싱하여 코드 변환 정보의 객체로 저장하게 하여 차후에 새로운 RFID 코드 체계가 추가되거나 기존의 RFID 코드 체계가 수정되더라도 코드 데이터를 쉽게 변환 처리할 수 있었다. 또한 주요한 표준 RFID 코드 체계에서 ALE 미들웨어의 내부적 처리할 경우에 필요한 URN 코드 형식을 제안하였다.

본 논문에서 제안한 ALE 미들웨어 기반 다중 RFID 코드 변환 처리 방법을 ALE 미들웨어에 적용할 경우, 현존해 있는 다양한 RFID 코드 체계를 처리할 수 있고 자체 정의한 코드 체계도 변환 처리 가능함에 따라 ALE 미들웨어는 유비쿼터스 미들웨어로 발돋움을 할 수 있을 것이다. 또한 값비싼 EPC 코드 체계를 쓰지 않고서 상대적으로 저렴한 ISO 코드 체계를 사용할 수 있어서, RFID 시스템 구축 비용이 줄어들 것이다. 더불어서 다양한 RFID 코드 체계를 ALE 미들웨어가 처리함에 따라 ALE 미들웨어는 더 이상 RFID 태그에 따라서 RFID 시스템을 구축하지 않아도 될 것이다.

앞서 언급하였듯이 다양한 RFID 코드 체계로 인하여 국내 RFID 산업의 영속성 및 네트워크화를 이루는 데에 어려운 점이 많았으나, 본 논문에 의해서 국내 RFID 산업의 영속성 및 네트워크화가 효율적으로 이루어질 것이다. 예를 들어서 하나의 ALE 미들웨어로 국제 RFID 산업에서는 EPC 코드 체계를 사용하고 국내 RFID 산업에서는 ISO 15459 KKR 코드 체계를 사용하거나 EPC 코드 체계를 사용하여 유통 및 물류 중심의 RFID 산업을 할 수도 있고 모바일 RFID 코드 체계를 사용하여 개인 맞춤형 중심의 온라인 콘텐츠 서비스를 할 수도 있다.

본 논문에서 제안하는 코드 변환 정보에 의해서 향후에 새로운 RFID 표준 코드 체계를 효율적으로 추가 및 수정할 수 있음에 따라 ALE 미들웨어는 범용성과 확장성을 가질 수 있게 되어, ALE 미들웨어를 더 효율적으로 사용할 수 있게 하였다.

향후에는 실제로 EPC Gen1, Gen2 리더기 및 ISO 15459 리더기, 모바일 RFID 리더기를 설치하여 실제적으로 RFID 태그가 삽입된 물품을 읽힌 코드 데이터를 필드 테스트해야 할 것이다. 더불어서 URN 코드로 변환된 RFID 코드 데이터를 응용이 원하는 다양한 URN 코드 형식으로 변환하여 응용에게 전송하는 테스트도 실행해야 할 것이다. 그에 따른 기존의 ALE 미들웨어와 성능 테스트를 수행하며 대용량 코드 데이터가 일시적으로 들어올 경우 다양한 RFID 코드 데이터가 오차 없이 식별 및 변환이 되는 지의 테스트도 수행해야 한다.

# 참고문헌


[1] 변지웅, 이동철, 변영철, “ALE 미들웨어를 위한 KKR 코드 변환”, 한국해양정보통신학회논문지, 제12권, 10호, pp.1759-1766, 2008.10

[2] 김영일, 김말희, 이용준, “RFID 미들웨어 기술 동향 및 응용 사례”, 정보처리학회지, 제12권 5호, pp.43-45, 2005.

[3] EPCglobal Inc., "EPCglobal Architecture Framework Final Version", 1 July 2005.

[4] 김선호, 윤지호, 김진용, 안종환, “U-국가물품자산관리를 위한 식별코드체계 설계”, 대한산업공학회, 산업공학지, Vol. 20, No.2, pp.227-234, 2007.

[5] 김선호, 김진용, 박정재, 송주형, 김현민, 안종환, “국가물품자산관리를 위한 RFID 식별코드체계 운용방안”, 한국전자거래학회지, 제12권 3호, pp.19-30.

[6] 김경호, 정한영, 이상훈, “국방 RFID 태그 코드 선정 및 ONS 구축방안”, “정보과학회지, 제25권 9호, 2007.9

[7] 한국인터넷진흥원, “RFID 코드 인코딩 지침서 V1.0", 2007.9

[8] 한국인터넷진흥원, “국내외 RFID 정책 및 기술동향 보고서”, 2006.10

[9] 변지웅, 양문석, 차지윤, 노영식, 변영철, “온톨로지를 이용한 다중 RFID 코드 식별 변환 시스템”, 한국콘텐츠학회 춘계 종합학술대회 논문집, pp.591-593, 2008.5

[10] EPCglobal Inc,, EPCglobal Tag Data Translation 1.0, 2006.

[11] 한국인터넷진흥원, “RFID 코드 설계 및 적용 지침서”, 2007.12

[12] EPCglobal Inc., EPCglobal Tag Data Standard Version 1.3, 2006.

[13] 한국유통물류진흥원, “EPC 태그 데이터 표준 Version 1.3 활용 가이드라인”, 2007.5

[14] 변지웅, 홍연미, 노영식, 양문석, 변영철, “RFID EPC 코드 변환에 대한 연구”, 전자・전기통신학회, 2007 합동학술발표회논문집, 제22권, pp.69-71, 2007.8

[15] 한국인터넷진흥원, “RFID 검색 시스템 구축 및 운영 지침서 V1.2", 2006.12

[16] 한국인터넷진흥원, “사례제시를 통한 RFID 적용 본사, 지사간 자산출입관리

시스템 구축 가이드", 2006.12
[17] 한국정보통신기술협회, "모바일 RFID 코드 체계 및 태그 데이터 구조", 2005.12
[18] 한국인터넷진흥원, "모바일 RFID 코드 인코딩 지침서", 2007.7
[19] 변지웅, 이상준, 변영철, "ALE 미들웨어를 위한 모바일 RFID 코드 처리 방법", 한국해양정보통신학회논문지, 제12권 3호, pp.461-468, 2008.3
[20] EPCglobal Inc., The Application Level Events Specification, Version 1.0, 2005.9
[21] 홍연미, 변영철, "ALE 기반 RFID 미들웨어 설계 및 구현", 한국해양정보통신학회논문지, 제11권 4호, 2007.4
[22] 노영식, 변영철, "온톨로지 기반 EPC 코드 자동 변환 방법", 한국해양정보통신학회논문지, 제12권 3호, pp.452-460, 2008.3
[23] ISO/IEC 1545-1, "Information technology - Unique identifiers", 2005. 12
[24] Min Kyu Han, Il Woo Paik, Byung Hee Lee and Jin Pyo Hong, "A Framework for Seamless Information Retrieval between an EPC Network and a Mobile RFID Network", 2006 IEEE International Conference on, 2006 CIT, pp.98-98, Sept. 2006.
[25] Jun Seob LEE, Hyoung Jun KIM, "RFID Code Structure and Tag Data Structure for Mobile RFID Services in Korea", ICACT2006, pp.1053-1055, Feb. 2006.
[26] Myunghee Son, Yongjoon Lee, Cheolsig Pyo, "Design and Implementation of mobile RFID technology in the CDMA networks", ICACT2006, pp. 1033-1036, Feb. 2006
[27] Taesu Cheong, Youngil Kim, Yongjoon Lee, "REMS and RBPTS : ALE-compliant RFID Middleware Software Platform", ICACT2006, pp.699-704, Feb. 2006.
[28] Gi oug Oh, Doo yeon Kim, Sang il Kime, Sung yul Rhew, "A Quality Evaluation Technique of RFID Middleware in Ubiquitous Computing", ICHIT06, 2006.