한국정보전자통신기술학회논문지(jkiiect)'18-08, Vol.11 No.4
논문 18-11-04-319 http://dx.doi.org/10.17661/jkiiect.2018.11.4.319

ISO/IEC 10646과 멀티바이트 코드 세트간의 변환시스템의 설계 및 구현

김 철*

Design and Implementation of Conversion System Between ISO/IEC 10646 and Multi-Byte Code Set

Chul Kim*

요 약 본 논문에서는 ISO/IEC 10646과 멀티바이트 코드 세트간의 변환 시스템을 설계하고 구현한다. 65,000 문자 의 코드를 제공하는 UCS 세트는 128 문자의 코드 용량을 제공하는 ASCII 코드의 제한성을 해결하고, 전세계 언어의 표현, 전송, 교환, 처리, 저장 및 입출력을 단일 코드 페이지에서 적용하며, 다국어 소프트웨어 개발시 코드 변환을 단순화시킴으로써 프로그램의 코드 수정을 위한 시간과 비용을 효율적으로 절감하게 한다. 따라서 UCS 코드 시스템과 ASCII 및 EBCDIC 코드 시스템들이 혼용되어 사용되는 환경에서는 상호 시스템간의 코드 변환 방법은 시스템 마이그레이션시 제공되어야 하는 중요한 고려 사항이다. 본 논문의 코드 변환 유틸리티는 UCS와 IBM 호스트 코드간의 매핑 테이블을 포함하고 있으며 제안된 코드 변환 알고리즘을 시스템에서 구현하였다. 제안된 코드 변환 프로그램은 실제 시스템 환경에서 성공적으로 구동하였음을 검증하였고, UCS와 멀티바이트 코드 시스템간의 마이그레이션시 가이드라인으로 제공될 수 있다.

Abstract In this paper, we designed and implemented a code conversion method between ISO/IEC 10646 and the multi-byte code set. The Universal Multiple-Octet Coded Character Set(UCS) provides codes for more than 65,000 characters, huge increase over ASCII's code capacity of 128 characters. It is applicable to the representation, transmission, interchange, processing, storage, input and presentation of the written form of the language throughout the world. Therefore, it is so important to guide on code conversion methods to their customers during customer systems are migrated to the environment which the UCS code system is used and/or the current code systems, i.e., ASCII PC code and EBCDIC host code, are used with the UCS together. Code conversion utility including the mapping table between the UCS and IBM new host code is shown for the purpose of the explanation of code conversion algorithm and its implementation in the system. The programs are successfully executed in the real system environments and so can be delivered to the customer during its migration stage from the UCS to the current IBM code system and vice versa.

Key Words : Basic Multilingual Plane, Double Byte Character Set, ISO/IEC 10646, UCS, Universal Multiple-Octet Coded Character Set, UNICODE

1. 서론

컴퓨터에서 처리해야 할 문자수의 증가와 더불어 소프트웨어들을 각국의 언어 환경에 맞게 수정하는 비용이 점차 증대되고 있다. 따라서 이러한 문제를 해결하기 위해서 다국어 지원 소프트웨어의 개발이 절실이 요구되고 또한 일정한 길이로 전세계에서 사용되는 주요 문자들을 표현하고자 하는 요구가 대두하게 되었다. 이와 같은 요구들에 부응하기 위하여 국제표준화기구에서는 멀티바이트 부호체계인 UCS(Universal Multiple-Octet Coded Character Set, 국제 부호화 문자 세트)를 구성하였으며, 또한 미국 기업들을 중심으로 컨소시엄을 형성하여 UCS

*Corresponding Author : Department of Computer Science, Yongin University (chulkim@yongin.ac.kr)
Received July 25, 2018 Revised July 28, 2018 Accepted August 03, 2018

와 동일한 국제공용글자판인 기본 다국어 평면(BMP, Basic Multilingual Plane)을 사용하는 2바이트 부호체계인 UNICODE가 제정되었다[1, 2].

최근에 IBM과 Microsoft등과 같은 정보산업의 주도적인 업체들이 UNICODE를 발표되는 제품들에 속속 적용함에 따라 UCS 멀티바이트 코드체계와 SBCS(Single Byte Character Set)나 DBCS(Double Byte Character Set) 혹은 혼합 필드(Mixed Field)등과 같은 기존의 코드체계 상호간의 변환을 위한 시스템 및 사용자 지원 유틸리티의 개발에 대한 요구가 증대하여 왔다[3, 4].

따라서 본 연구에서는 컴퓨터상에서 문자표현에 강력한 영향력을 발휘할 것으로 예상되는 UCS 멀티바이트 코드체계의 적용방법을 연구하고 ASCII나 EBCDIC 혹은 업체 고유의 기존 코드와의 상호변환을 위한 방법을 제시하며 또한 그 코드변환 유틸리티를 시스템에 설계 및 구현한다. 2장에서는 국제 부호화 문자 코드의 체계에 대해 알아보고, 3장에서는 UCS와 IBM 코드체계간의 코드변환 유틸리티의 구현을 위한 코드 변환 알고리즘을 제안하고 설계한다. 4장은 결론과 추후 연구되어야 할 고려 사항들을 기술한다.

2. 국제 부호화 문자 코드의 체계

국제표준화기구에서 추진한 4바이트 코드체계인 UCS는 128개의 그룹에 각 그룹은 256개의 판(Plane)을 갖는 4차원의 문자코드 체계이다[1, 7]. 이 체계는 각 판이 256개의 행과 셀을 갖는 영역이 존재하며 각 판은 6만여 자를 2바이트로 코드화시켜 표현할 수 있다. UCS의 전체 구조는 그림 1과 같다. 이 구조의 규정은 다음과 같으며, 임의의 옥텟값은 ISO/IEC 10646에 있는 00부터 FF까지의 16진수로 나타낸다. UCS의 정규형에서는 이해를 쉽게 할 수 있도록 128개의 3차원 그룹으로 구성되는 4차원 부호화 공간을 단일 개체로 보고 사용한다. 각 그룹은 256개의 2차원 평면으로 구성된다. 각 평면은 256개의 1차원 행으로 구성되며, 각 행에는 256개의 셀이 포함된다. 문자는 이와 같은 부호화 공간에 속해 있는 셀에 위치하여 부호화되고 그렇지 않은 경우 이 셀은 사용하지 않은 것으로 선언된다. 정규형에서는 각각의 문자를 나타내기 위해 4옥텟을 사용하며, 이들 4옥텟은 그룹, 평면, 행, 셀을 각각 지정한다. 정규형을 4옥텟으로 구성하는 이유는 전 세계의 모든 문자들을 2옥텟으로 전부 다루기에는 부족하며, 현재 널리 사용되는 프로세서의 구조가 32비트 표현 방식을 채택하고 있기 때문이다. 4옥텟 정규형을 4옥텟 문자 부호계로도 사용할 수 있는데 이 경우에는 UCS-4라고 부른다. 첫째 평면(그룹 00의 평면 00)을 BMP(기본 다국어 평면)라고 부른다. BMP에는 영문자 및 발음 기호나 표의 문자 스크립트뿐만 아니라 다양한 기호나 숫자에서 일반적으로 사용하는 문자들이 포함되어 있다. 아울러 BMP는 문자가 특수한 성질을 가지는 제한 사용(RU : Restricted Use) 영역도 가지고 있다. 나머지 평면들은 보조 또는 전용 평면으로서 그래픽 문자들을 추가로 수용하고 있다. 그룹 00에서 평면 옥텟값 E0에서 FF인 32개의 평면은 전용 평면이다. UCS 그룹 옥텟값 60에서 7F까지의 32개 그룹 역시 사용자 정의용이다. 사용자 정의 영역에 속해 있는 셀의 내용에 대해서는 ISO/IEC 10646에서 규정하지 않는다. 각 문자는 그룹 옥텟, 평면 옥텟, 행 옥텟, 셀 옥텟의 형식으로 이 문자 부호계 내에 위치한다. 정규형뿐만 아니라 2옥텟 BMP형식도 규정되어 있다. 따라서 BMP를 UCS-2라고 하는 2옥텟 부호화 문자로서 사용할 수도 있다. 그래픽 문자의 하위 레퍼토리(sub-repertoire)를 제공하기 위해 부호화 공간의 하위 집합을 사용할 수도 있다. UCS 변환 형식(UTF-1 : UCS Transformation format)은 ISO 2022의 구조에 따라 부호화 제어 문자의 옥텟값에 민감한 통신 시스템을 통해 텍스트 데이터를 전송할 때 사용할 수 있다[5, 6].

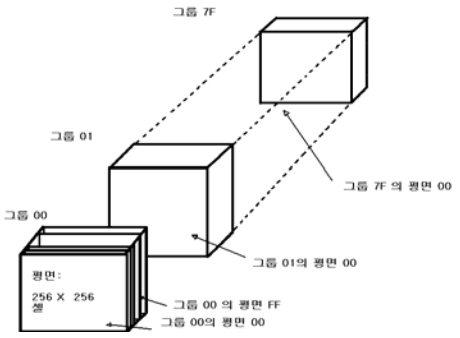

그림 1. UCS의 전체 부호화 공간
Fig. 1. Overall Coding Space of UCS

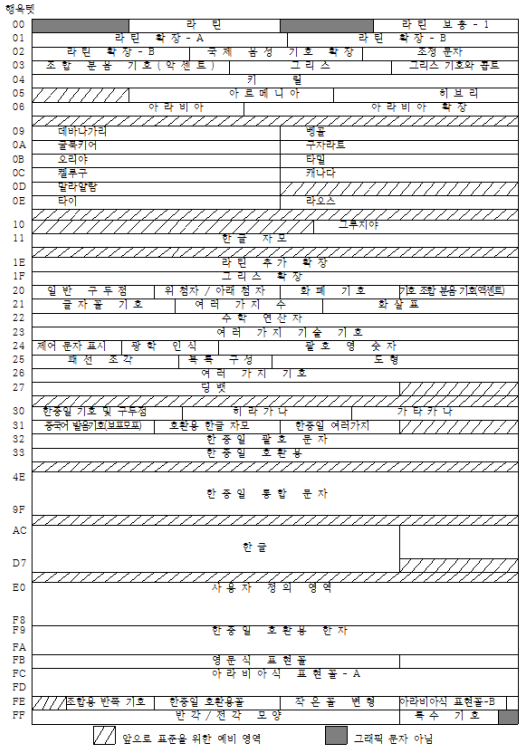

그림 2. BMP의 구조
Fig. 2. BMP Architecture

그룹 00의 평면 00는 BMP에 해당한다. 2옥텟의 문자 부호계로서 BMP를 사용할 수 있으며, 이 경우 UCS-2라고 부른다. BMP는 네 영역으로 나눌 수 있다. 즉, A-영역은 부호자리 0000에서 4DFF까지, I-영역은 부호자리 4E00에서 9FFF까지, O-영역은 부호자리 A000에서 DFFF까지, R-영역은 부호자리 E000에서 FFFD까지다. BMP에서 부호자리 0000에서 001F까지는 제어 문자용으로 남겨져 있으며, 부호자리 007F는 DELETE 문자용으로 남겨져 있다. 부호 자리 0080에서 009F까지는 예비용이다. 전체 BMP 구조는 그림 2와 같다[1, 7].

3. UCS와 IBM 코드 체계간의 코드변환 유틸리티의 설계 및 구현

3.1 IBM 문자 데이터 표현 구조

IBM의 문자 표현을 위한 구조는 문자들의 결집체인 문자집합(Character Set), 해당 문자 집합의 각 문자에 대한 이진코드 표현(Binary Code Representation)을 규정하는 코드 페이지(Code Page), 그리고 코드페이지를 위한 코딩공간(Coding Space)의 할당 방식을 규정하는 인코딩 체계(Encoding Scheme)로 이루어진다[3, 4]. 문자집합은 특정언어로 컴퓨터 시스템을 지원하기 위하여 필요한 디스플레이 또는 프린트 가능한 문자들의 결집체이다. 컴퓨터 시스템에서 정보처리를 위해서는 제어문자(Control Character)의 정의가 필요하지만 여기서 말하는 문자집합은 그래픽 문자집합을 의미한다. 이러한 문자집합은 주어진 언어에 따라, 그리고 주어진 시스템에 따라 다르게 정의될 수 있다. 컴퓨터에서 처리가능한 코드 구현시 8비트 1바이트로 표현이 가능한 문자집합을 SBCS(Single Byte Character Set)라 하고, 문자의 개수가 256보다 커서 내부 표현을 위해 2바이트가 필요한 문자집합을 DBCS(Double Byte Character Set)라 한다. IBM에서 사용하는 모든 문자집합은 고유한 식별자인 GCSGID(Graphic Character Set Global IDentifier)를 가지며, 이 식별자를 이용하여 다른 문자집합과의 구별이 가능하다. 코드페이지는 특정 문자집합의 각 문자에 대해 컴퓨터에서 처리가능한 이진코드(Binary Code) 또는 코드값(Code Point)을 부여한 것이다. 문자집합과 마찬가지로 IBM에서 사용하는 모든 코드페이지는 고유한 식별자인 CPGID(Code Page Global IDentifier)를 가지며, 이 식별자를 이용하여 다른 코드페이지와의 구별이 가능하다.현재 IBM의 문자 표현구조에서 정의하는 인코딩 체계는 대형시스템에서 주로 사용하는 EBCDIC 코드와 PC에서 사용하는 ASCII 코드로 구분된다. EBCDIC 및 ASCII 코드의 인코딩 체계는 코딩공간(Coding Space) 할당을 위해 사용되는 바이트수에 따라 각각 1바이트와 2바이트 체계로 나누어진다[3, 4].

3.2 UCS와 IBM 코드 체계간의 코드변환

UCS 코드체계와 IBM 코드체계간의 코드변환 테이블 및 프로그램은 클라이언트와 서버, 호스트나 워크스테이션등과 같은 상이한 기종에 관계없이 사용할 수 있는 표준 C 언어와 사용자 인터페이스를 이용하여 윈도우즈 환경에서 개발하였다. 또한, 특히 우리나라에서 현재 많이 사용중인 IBM 코드체계를 선정하여 코드변환 유틸리티를 구현하였다. 이 UCS와 IBM 코드체계간의 상호 변환 관계는 그림 3과 그림 4와 같다. 이중에서 본 논문에서는 UCS와 IBM의 DBCS인 New Host 코드(CP:834) 상호간의 코드변환의 경우를 설명한다.

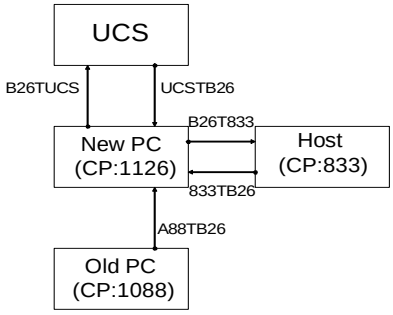

그림 3. UCS 코드 체계와 IBM의 SBCS 코드 체계간의 코드변환 상호관계

Fig. 3. Co-relationship of Code Conversion between UCS and IBM SBCS

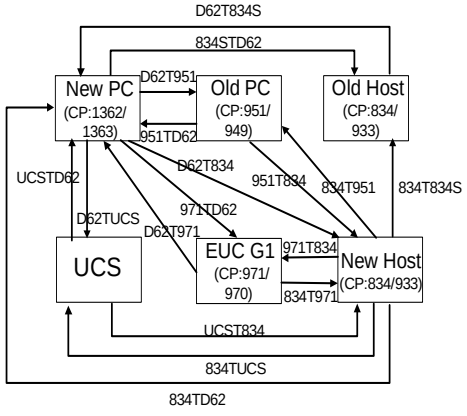

그림 4. UCS 코드 체계와 IBM의 DBCS 코드 체계간의 코드변환 상호관계

Fig. 4. Co-relationship of Code Conversion between UCS and IBM DBCS

먼저 코드변환 알고리즘을 설명하면 다음과 같다. UCS 코드와 IBM의 New Host 코드와의 변환을 위한 매핑을 구현하는 가장 간단한 방법은 전체 코드공간에 대해 UCS에서 New Host로의 매핑 테이블과 New Host에서 UCS로의 매핑테이블을 두는 방법이다. 예를 들어 UCS 코드 4e00을 New Host 코드로 변환하기 위해서는 단순히 UCS에서 New Host로의 매핑 테이블의 4e00번째 내용을 읽어오면 된다. 이 방법은 구현이 매우 간단하다는 장점이 있으나, UCS와 New Host 코드체계가 모두 2 Byte 코드체계를 가지므로 테이블의 크기가 216 = 65536 개의 코드를 가지고 있어야 한다. 따라서 UCS to New Host Table이 64 K * 2 Byte, 그리고 New Host to UCS Table이 64 K * 2 Byte를 유지해야 하므로 사용해야 하는 메모리 공간이 256 KByte에 이른다.이는 메모리 공간의 측면에서 비효율적이다. 실제로 UCS 코드체계와 IBM의 New Host 코드체계는 전체 64K의 코드 공간 중 18K의 코드공간을 사용한다. 따라서 각각의 코드체계를 코드값의 순서에 의해 정렬하고 빈공간이 없도록 압축하여 테이블을 만든다. 이를 각각 TUCS, THOST라 하고 이들 간의 매핑은 두 개의 테이블의 인덱스간의 매핑으로 정의된다. 매핑 테이블의 내용은 변환 대상 코드 테이블의 Index로 구성된다. 예를 들어 UCS 코드 AC00을 변환하기 위해서는 먼저 TUCS에서 AC00이 몇번째 내용인가를 검색한다. 검색과정은 테이블이 정렬되어 있으므로 이진탐색을 이용하여 효율적으로 수행할 수 있다. TUCS로부터 AC00에 대한 Index를 얻어내면 이 Index로부터 UCS to New Host Index Mapping Table에서 Target Table(THOST)의 Index를 얻어낸다. 이제 THOST테이블의 Target Index의 내용을 읽어오면 변환이 완료된다. 이를 그림으로 표현하면 그림 5와 같다. 이 경우 사용되는 메모리 공간은 TUCS와 THOST가 각각 36K, Index Mapping Table 2개가 각각 36K를 차지하게 되므로 총 36K * 4 = 144 KByte의 메모리 공간을 사용하게 된다. 단순매핑에 비해 43.7%의 메모리 절감효과를 가져온다. 이에 대한 알고리즘은 각각 Algorithm 1과 Algorithm 2에 기술하였다.

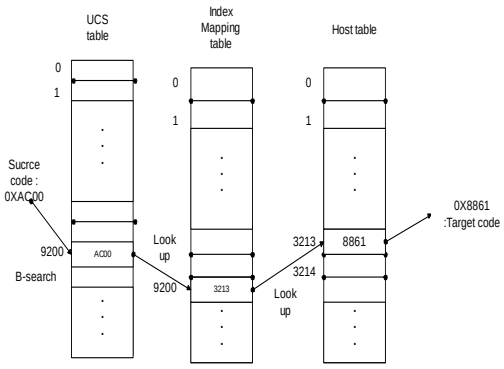

그림 5. UCS 코드와 IBM New Host 코드간의 매핑관계
Fig. 5. Mapping Relationship between UCS and IBM New Host

TUCS : sorted UCS code table.
THOST : sorted New Host code table.
FU2H : index mapping table from TUCS index to THOST index.
FH2U : index mappint table from THOST index to TUCS index.

***Algorithm 1**: Simple Algorithm for UCS to HOST Code Conversion*

1. *Let SCode as source code to be converted.*
2. *Find SCode index from TUCS using binary search method and let's call it 'SIndex.'*
   i.e. SCode is TUCS's SIndexth entry.
   If SIndex is NULL then Return "Invalid UCS Code", Step 5
   Else Step 3.
3. *Find THOST index using F and let's call it 'DIndex'*
4. *target code is THOST(DIndex)*
5. *Stop*

***Algorithm 2**: Simple Algorithm for New Host to UCS Code Conversion*

1. *Let SCode as source code to be converted.*
2. *Find SCode index from THOST using binary search method and let's call it SIndex.*
   i.e. SCode is THOST's SIndexth entry.
   If SIndex is NULL then Return "Invalid New Host Code", Step 5
   Else Step 3.
3. *Find TUCS index using F and let's call it DIndex*
4. *target code is TUCS(DIndex)*
5. *Stop*

그림 6은 윈도우즈 환경에서 수행한 코드변환 유틸리티의 실행 화면을 보여주고 있다. 이 프로그램은 윈도우즈 환경에서 작성되었고, 사용자에게 친숙한 Wizard 형식의 사용자 인터페이스를 채택하였다. 표 1은 본 논문에서 사용한 UCS 코드와 IBM New Host 코드(CP:834)간의 매핑 테이블의 매핑관계의 일부분을 보여준다.

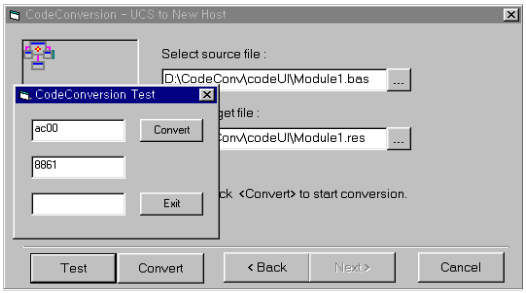

그림 6. UCS 코드와 IBM New Host 코드간의 코드변환 실행화면
Fig. 6. Execution Screen of Code Conversion between UCS and IBM New Host

표 1. UCS 코드와 IBM New Host 코드(CP:834)간의 매핑 테이블

Table 1. Mapping Table between UCS and IBM New Host Code(CP:834)

Basic Latin	CJK Compatible Hanja Code
U-0000 H-0000 U-0001 H-0001 U-0002 H-0002 U-0003 H-0003 U-0004 H-0037 U-0005 H-002d U-0006 H-002e U-0007 H-002f U-0008 H-001f6 U-0009 H-0005 U-000a H-0025 U-000b H-000b U-000c H-000c U-000d H-000d U-000e H-003f	U-4e00 H-5fa8 U-4e01 H-60dc U-4e03 H-6470 U-4e07 H-696e U-4e08 H-5ff4 U-4e09 H-58d7 U-4e0a H-59e5 U-4e0b H-65a9 U-4e0d H-699a U-4e11 H-63e1 U-4e14 H-62a1 U-4e15 H-5942 U-4e16 H-5ac9 U-4e18 H-526f U-4e19 H-5871 .....
Hangul Syllables	**Private Use**
U-ac00 H-8861 U-ac01 H-8862 U-ac02 H-8863 U-ac03 H-8864 U-ac04 H-8865 U-ac05 H-8866 U-ac06 H-8867 U-ac07 H-8868 U-ac08 H-8869 U-ac09 H-886a U-ac0a H-886b U-ac0b H-886c U-ac0c H-886d U-ac0d H-886e U-ac0e H-886f U-ac0f H-8870	U-e000 H-d441 U-e001 H-d442 U-e002 H-d443 U-e003 H-d444 U-e004 H-d445 U-e005 H-d446 U-e006 H-d447 U-e007 H-d448 U-e008 H-d449 U-e009 H-d44a U-e00a H-d44b U-e00b H-d44c U-e00c H-d44d U-e00d H-d44e U-e00e H-d44f U-e00f H-d450

4. 결론

본 논문에서는 ISO/IEC 10646과 멀티바이트 코드 세트간의 변환 시스템을 설계하고 구현하였다. 제안 시스템은 다국어 소프트웨어 개발시 코드 변환을 단순화시킴으로써 UCS 코드 시스템과 ASCII 및 EBCDIC 코드 시스템들이 혼용되어 사용되는 환경에서 시스템 마이그레이션 시 가이드라인으로 사용될 수 있다. 본 논문의 코드 변환 유틸리티는 UCS와 IBM 호스트 코드간의 매핑 테이블을 포함하고 있으며 제안된 코드 변환 알고리즘을 시스템에서 구현하였다. 제안된 코드 변환 프로그램은 실제 시스템 환경에서 성공적으로 구동하였음을 검증하였다.

REFERENCES

[1] ISO, ISO/IEC 10646, Information technology - Universal Coded Character Set(UCS) - part 1 : Architecture and Basic Multilingual Plane, 2017.

[2] The Unicode Consortium, The Unicode Standard, Version 11.0, 2018.

[3] IBM, National Language Support Reference Manual Vol. 2, 1992.

[4] IBM Korea, IBM Code User Manual, 1992.

[5] ISO, ISO 2022, Information processing - ISO 7-bit and 8-bit coded character set - Code extension techniques, 1986.

[6] ISO, ISO/IEC 6429, Information technology-Control functions for coded character sets, 1992.

[7] Korean Standards Association, Universal Coded Character Set : KS C 5700, 1995.

Author Biography

Chul Kim **[Regular member]**

- Feb. 1977 : Yonsei Univ., Electronics Engineering, B.S.
- Aug. 2000 : Yonsei Univ., Computer Science, Ph.D.
- Jul. 1981 ~ Jan. 1984 : Samsung LtD, Researcher
- Feb. 1984 ~ Jan. 1993 : IBM Korea, Engineer
- Mar. 1994 ~ current : Yongin Univ., Dept. of Computer Science, Professor

<Research Interests> Protocol Engineering, Computer Security