# 더 빠르고 더 정밀하고 더 정확한 한글자소

최농부

# 소개글

더 빠르고 더 정밀하고 더 정확한 정보 한글자소의 이해(한국이 행복해 질 수 있는 미래창조과학기술개발시대)라는 제목으로 연구 고찰된 반도체 최적화문자열 한글자소의 우수성을 적은 책이다.
한국인의 지능이 세계에서 가장 우수하다는 유대인 보다 지능지수가 평균 2점높다는 것은 한글자소를 사용했기 대문인 것으로 한글자소를 사용하게 되면 뇌 발달이 수직수평공간감각으로 발달하고 알파벳을 사용하게 되면 평면감각이 발달하게 되는 뇌 발달구조가 다르다는 것이 이 책의 주제이다..

# 목차

# 더 빠르고 더 정밀하고 더 정확한 정보 한글자소의 이해

2016.03.06 06:01

차례



## 1, 머리말

필자는 컴퓨터시대에 누구보다도 컴퓨터혜택을 많이 본 사람으로 앞으로 세계문화주도는 반드시 한국으로서 한국이 문화주도국이 되는 이유는 2 가지이며 이는 한국이 행복해 질 수 있는 이유이기도하는 것이다.

하나는 한글이 첨단반도체자소(字素)라는 것이고
다른 하나는 벼를 싹 틔워 젓가락으로 밥을 먹을 수 있기 때문이다.

이순의 나이를 앞두고 '식물의 광 분자생리' 에 대한 책을 집필하고 '태양광발생소멸시각에 대한 천체물리량을 계산하여 지구의 자전축을 실측하는 과정에서 태양광발생소멸시각변화량vector 와 생체시계

진자vector’ 라는 논물을 썼고 ‘지구자전축의 흔들림 량을 수치로 실측’ 하는 등의 결과로 ‘지구가 흔들리기 때문에 생체시계가 돌아간다,’ 라는 책을 저술하기도 했다.
실로 ‘지구 흔들림에 대하여’ 라는 저술은 지구의 환경과 에너지문제를 담고 있는 책이다.
이런 책들을 집필하는 자료가 모두 컴퓨터를 통하여 자료를 수집했고 실측 물리량 스칼라(scalar)를 vector화하는 작업의 고급물리학에 해당하는 것이지만 이를 컴퓨터를 통해서 해결할 수 있었기 때문에 컴퓨터혜택을 톡톡히 본 셈이다.
자연과학의 문제 해결을 위해 “미리 답을 내리고 맞는지를 확인하는 방법으로 다수의 논문을 썼고, 다수의 프로그램 S/W도 개발등록 한 것이 30여 편에 육박하게 되었지만 모두가 전공에 의한 것은 아니며 필요에 의해 공부하고 문제 해결을 위해서 공부하다보니 답을 내릴 수가 있었고 그 답을 확인할 수도 있었든 것이다.

세상에는 세계를 바꾸는 불가사이 한 인물이 더러 있다.
이들을 기인이라고 해야 하는지는 모르겠으나 세상을 바꾸어 놓은 것은 틀림이 없다.

1543년 5월 24일 폴란드의 니콜라스 코페르니쿠스는 당시 측량 기술과 장비로는 도저히 불가능한 지구공전을 찾아냈고 자전도 찾아냈다 1592년 7년간의 전쟁 속에서 정부지원하나 없이 23전 전승의 성웅이순신 같은 분이 있다.
본서에서 자주 거론되는 1446년 반포한 한글이 현 반도체시대 가장 최적화된 자소라는 것을 발견할 때 수세기를 앞서는 혜안의 눈을 가진 사람들이라 할 수 있다.
후일 여러 분야에서 한글의 근원을 모방의 글이라고 하는 이가 있으나 한글 24개의 기본글자는 수십세기역사에서 비슷한 것을 찾을 수가 있다 하드라도 기우에 지날 뿐이고 어찌하였든 정연하게 정리하여 사용해 온 것은 사실이고 그것이 지금에서는 모방할 수 없을 정도로 첨단반도체를 위한 최적글자로는 인정해야 한다.

한글은
자소(字素) = 성소(聲素)로서
11,172개 글자 값 = 자소조립 글자 11,172개 = 11,172개 발성기호
라는 반도체글자공학으로 2 대 특징인 ‘바로자소, 조립자소’ 모두를 갖춘 글자로 절대 모방할 수 없는 글자임은 분명하다.
필자는 언어학을 전공하지는 않았기 때문에 한글이 조립자소라고 단정을 내릴 수 있었다.
필자는 아스키코드가 어떤 것인지 유니코드가 어떤 것인지도 몰랐고 ‘조합형한글’ 이니 ‘완성형한글’ 이니 하는 것은 더 더욱 모르고 72개 입력단추의 ‘조립자소 판’ 을 개발하여 등록 하였다.
개발이유는 가장 최적화된 반도체적성자소로서 알파벳자소 자판의 언저리에 얹혀서 사용하다보니 매우 불편했고 자존심이 상한 것이 이유이었으며, 한글을 사용하는 문화권의 한 사람으로서 얼마나 행복한 사람인가! 하는 고마움에 이 책을 쓰기로 마음먹기도 했었다.
필자에게 최적화된 한글자소로서 초기 반도체기기인 컴퓨터용 프로그램을 개발할 수 있을까? 라고 질문한다면 그 대답은 절대적으로 ‘아니다’ 라고 단정한다.
한글이 반도체기기 ‘초기 개발’ 자소는 아니기 때문이다.
반도체기기 ‘초기개발시대’ 자소는 아니지만 성숙단계에 들어서면 반드시 한글에 의해서 운용되게 되도록 600년 전 1443년 기획되어 있었다는 착각을 이 책을 통하여 하게 된다.
지구인들이 살아가기 위해서 “경제 문화 등 모든 분야가 한글에 의한 한국인을 통하여” 이루어진다는 등식이 성립한다는 것을 알려주고자 하는 것도 이 책이다.
모든 발전의 원동력은 정보에 있고 가장 빠른 정보능력을 가진 문화가 세상을 주도하며, 그 정보매체는

사이버라는 공간에서 이루어지고 있다는 것이 이유이다.
본서를 읽다 보면 알파벳문화권의 모든 프로그래머들이 컴퓨터프로그램을 개발한 것은 마치 한글을 위한 것이었다는 착각을 하게 된다.
반도체시대 문명개발과 발전은 한글 문명개발이고 발전이며 컴퓨터 프로그램의 개발은 바로 한글프로그램개발임을 알게 된다는 것이다.
600년 전 세종임금이 미래에 반도체가 발달하므로 컴퓨터가 사용되며 그 컴퓨터자소로서 기획하고 설계 제작하였다는 착각을 하게 된다는 것이다.
2진법체계가 8진법 16진법 32진법 64진법으로 확장되어 갈수록 한글의 조립자소가 아름답다는 것을 알게 될 것이다.

세계는 왜 한류열풍에 매료되어야 하고 70억 인구 중에 0.7%에 겨우 미치는 5천도 못 미치는 한국인이 경제 7대 강국이고 과학 7대강국이며 특히 올림픽에서 조직력과 정밀한 변수 분석을 요구하는 경기에서 금 은 동의 매달을 싹 슬이 하는 것은 한글자소를 사용해온 한국인의 저력을 설명해 주고 있다.
단연 반도체부분은 첨단국가로서 IT강국이 될 수밖에 없다.
이 모두가 빠른 정보 매체인 한글자소 사용을 생활화한 한국인의 '빨리 빨리' 특유의 정신과 혼에서 발생된 산물로 봐야 한다.
한국인의 일상은 빨리 빨리라고 한다.
빠르다는 것은 느리다는 것에 대한 상대적인 속도의 비교이다.
한국은 결코 빨리 빨리는 아니고 정상적일뿐이다.
한국인은 한글자소를 쓰고 빠른 조립자소를 사용해 왔기 때문에 느린 알파벳자소의 생활과는 0.3 ~ 0.5배 빠른 일상생활을 하고 있다.
이러한 정보의 속도는 우리에게는 정상이나 알파벳 문화권에서 보면 한국인이 행동을 항상 빨리 빨리 하는 것으로 보인다.
한국인이 빨라진 것이 아니라 한국인이 아닌 문화권의 생활이 0.3 ~ 0.5 느린 것일 뿐이며 어찌했거나 한국인이 아닌 사람이 보기에는 "한국인이 빨리 빨리"로 보이는 것임에는 틀림없는 사실이다.

갈수록 아름다워지는 한글자소의 나아갈 길은 아름다운자소에 맞도록 프로그램기계언어가 개발되어야 하고 이에 맞는 운영체제도 개발되어야 한다고 주장한다.
이것은 한글을 위해서도 또 세계를 위해서도 반드시 투자와 개발이 이루어져야 한다는 것이다.
한국인이면 초등학교 수준에서도 소프트웨어를 개발할 수 있도록 체제가 개선된다면 한국은 과학1위 경제1위의 인력문화1위시대가 올 것이라는 미래가 보이기 때문이다.

필자는 이 책을 누가 읽을 것인가를 생각해 보았다.
모든 한국인이 다 읽어보기를 바란다.
초등학생에서부터 최고 경영자나 공무원 정치인 대학의 교수들까지 한국인이면 모두에게 읽혀지기를 바란다.
어렵다 난해하다를 따지지 말기를 바란다.
그저 부담 없이 읽어보기만을 바란다.
왜냐하면 어느 누구도 글자를 '조립자소와 바로자소'로 분류해본 사람이 없고 한글을 첨단반도체최적화자소라고 하지 않았기 때문이다.
밥을 먹는 한국인에게 밥이 무어냐고 묻거나 한글을 사용하는 한국인에게 한글이 어떤 글자이냐 하고 물으면 대답할 수 있는 한국 사람은 드물다는 것이 이 책을 반드시 읽으라는 이유이다.

필자는 언어학자도 컴퓨터공학자도 아니며 필부의 농사꾼일 뿐이었기 때문에 한글이 반도체시대 주체의 글자임을 발견할 수 있었고 또 이 글을 쓸 수 있다고 생각한다.
따라서 부담 없이 읽어 주기를 바라지만 분명한 것은 한글은 반도체최적성의 아름다운 글이기 때문에 반도체시대의 한국반도체진흥을 위해 한국인에 의해 한글프로그램기계언어가 탄생되어 초등학생수준에서도 쉽게 프로그램 소프트웨어 개발이 될 수 있게 되기를 바란다.
인간이 사용하는 수의 세계가 반도체와 더불어 점점 확대되어 가고 정밀지향으로 가고 있다. 한국의 혼도 한글정보세계에서 더 빨리 더 정밀하게 더 정확하게 확대지향으로 빠른 속도로 발전 해야만 한다고 생각한다.
인간이 사용하는 수가 테라($10^{12}$ ,tera)를 넘어 페타($10^{15}$ □, peta)를 향하고 정밀한 수로는 펨토($10^{-15}$ □, femto)를 넘어 아토(att)를 향하는 "펨토과학시대" 이다.
$2^{32}$ = 4,294,967,296,
$2^{64}$ = 115,281,504,606,846,976
모두가 컴퓨터시대가 가져온 산물이라고 한다면 32진법과 64진법으로 확대지향 하는 시대는 엑사($10^{18}$ □,exa)를 넘어서는 시대가 온다.

점차적으로 세계는 주방농업으로 잠재기능성강화식품의 밥을 먹게 된다.
한국인처럼 주방농업을 개발하고 음식문화의 혁명으로 밥을 먹게 된다.
벼만이 세계를 모두 먹여 살릴 수 있는 식량이고 벼로서 밥을 지어 먹으려면 젓가락을 사용해야한다.
한글을 사용하고 주방농업으로 잠재기능성강화식품을 생산하고 젓가락으로 밥을 먹는 것은 한 걸음 앞선 미래창조과학기술의 세상을 개발하여 살아갈 수 있게 한다.
그래서 한국인은 행복해 지는 것이다.

## 2, 정보바다와 한글

지난 '뿌리 깊은 나무 "라는 연속극은 한글창제에 대한 것을 다룬 사극이다.
세종대왕이 한글을 창제 하는 목적이 작개언로 달사총(作開言路 達四聰)에 있다는 것이 이 사극의 주제라고도 볼 수 있다.

'작개언로(作開言路)' 란 말의 길을 튼다는 것이고 '달사총(達四聰)' 이란 사방에서 듣는다는 한자말로서 '作開言路 達四聰' 을 반도체기기를 사용하는 현 인터넷시대어로 바꾸면 '정보화' 이고 한글은 '정보바다' 라고 할 수 있다.
한글은 이러한 의념과 배경 속에서 글자를 제작하여 1446년 10월 9일 공표한 생일 있는 글자가 되었다
당시 세종은 백성들의 소리를 들으려 하나 어려운 한문글자를 백성들이 배우고 익혀서 상소를 작성하여 임금에게 올린다는 것은 불가능한 것이어서 세종임금이 백성들도 쉽게 글을 익히고 쉽게 쓸 수 있다면 백성들의 소리도 쉽게 들을 수 있으리라는 것에서 시작된 것이 한글창제의 시작이 된 것이다.
반대가 심한 열악한 환경 속에서도 세종은 발성을 인체공학적 측면으로 접근하여 초성 중성 종성으로 된 28개의 성소(聲素)로 된 자소(字素)를 만들었다.
한글은 성대를 통하여 숨 쉴 때 발성기관을 진동하여 발생하는 목소리를 해부하고 그 소(素)를 기본으로 하여 글자를 설계하고 제작한 것이다.
때문에 목소리 소는 성소(聲素)이고 글자 소는 자소(字素)임으로
성소 = 자소

이렇게 탄생한 한글이 600여년의 세월 속에 갈고 다듬기를 반복하여 28개의 글자 중

4개는 옛글로 사라지고 현재 사용하는 “빛나는 현대한글” 이 되었다.

필자는 “가장 빠른 정보가 가장 빠른 판단을 한다.” 고 하며 한글은 철저한 반도체적성을 갖춘 빠른 “정보글자” 라고 정의한다.
그리고 “시력보호글자” 라고도 정의한다.
이런 차원에서 반도체적성을 갖춘 한글정보글자입력 자소 판을 반도체적성에 맞도록 다시 최적화시켜야 한다고 하고 새로운 ‘자소 판’ 을 창안하였다.

반도체기기인 컴퓨터나 휴대용기기에서 글꼴은 가독성과 집중도를 높여 주고 입력속도에도 영향을 주기 때문이다.
그 이유로는 글꼴 폭이 고정을 유지하기 때문인 것으로 이러한 반도체글자가 되기 위해서는
1). 고정 폭 글꼴
2). 글꼴과 부호의 확실한 구별
3). 한글의 굴림체에 버금가는 가독성
을 갖추어야 한다.
이런 이유에서 알파벳은 2)항목에서 굴림체로 할 경우 전혀 구별할 수 없는 ‘l’ , ‘I’ , ‘1’ , ‘0’ , ‘O’ (영대문자 I, 영소문자 l, 숫자1, 숫자 0, 영대문자 O) 자소들이 있다.
그러나 한글에는 구별 불가능한 이러한 글자가 한자도 없다는 것이어서 확실한 ‘시력보호글자’ 라고 할 수 있다.
2000년이라는 새로운 천년의 시대가 되면서 가장 빠른 정보 수단은 인터넷사이버라는 공간에서 이루어지기 시작하였다.
인터넷사이버 공간에서는 새로운 정보를 얻기 위해서 자신이 가지고 있는 자기정보가 빠르게 인터넷사이버 공간으로 입력시켜 전달한 후 이 정보를 통하여 인터넷사이버에서 들어온 새로운 정보는 빠르게 자기정보로 판단하고 해석하면서 살아가야 한다.
이렇게 인터넷사이버라는 공간정보매체가 2000년대 전후에 새롭게 출현하였고 빠른 자기정보를 얻기 위해서는 인터넷사이버라는 공간에서 글자를 이용해야 하고 소리로도 이용해야 한다.
인터넷사이버에서 ‘사이버’ 란 가상이라는 것으로 실체나 형체 없는 무형이라는 것을 말한다.
즉, 허상이라고 하는 것이다.
미래사회를 “ cybernetics + punk ” 라고 하는 경우도 있다.
허상이라는 가상공간에서 물건을 사고팔며 결제하고 결정한다.
정보를 주기도 하고 받기도 한다.
이렇게 가상공간에서 자기정보가 실현되므로 정보의 생명이 속도라고 할 때 “입력과 실행이 빠른 글자로서 소리와 일치하고 꼴이 분명한 약속된 글” 이 필요하다.
여기서 글자와 소리가 일치하고 꼴이 분명한 약속이란 ‘발성기호가 필요 없고 모양이 확실하게 구분되는’ 글자라야 한다는 것이다.
이러한 요건을 갖춘 글자는 한글이다.
한글은 인터넷사이버글자로서는 가장 최적화글자라는 것이다.

인터넷사이버는 반도체로 구성되는 것이기 때문에 인터넷사이버최적화글자는 곧 반도체적성최적화글자라고도 할 수 있다.
한글을 반도체적성이라는 측면에서 관찰하면 정말 매력적이고 환상적인 글자라는 것을 알게 된다.

## 3, 한글의 특징

왜 한글이 환상적인 반도체적성글자인가에 대한 가장 간단한 답은 “11,172개의 글자는 24개 기본자소와 27개 파쟁자소를 합한 51개 자소로 조립”된다는 것이다.
한글은 목소리글자이다.
목소리 글은 곧 발성기호와도 같다.
한글은 발성기호가 필요 없는 글자이다.
소리를 오실로스코프(oscilloscope)로 기록하여 나타내면 인간은 인간대로 만인 만색으로 나타나고 동물은 동물대로, 자연의 소리, 기계소리 등 무수하게 많은 소리들이 진동으로 나타내며 각각 다르게 나타내지만 인체공학적으로 발생하는 소리인 목소리를 해부하여 분류하면 첫소리, 중간소리, 끝소리를 구성요소로 하고 있다.

우리는 일상생활에서 발생하고 들리는 소리를 음이라 하기도 하며 성이라 하기도 한다.
인체발성기관인 성대에서 나는 것을 통상 소리라고 하고 발성이라 표현한다.
음은 인체의 성대에서 발성되는 이외의 모든 소리를 말한다.
잡음은 목소리가 아닌 소리이고 잡소리는 사람의 소리이다.
연주하는 사람을 음악가라하고 노래하는 사람을 성악가라 한다.
악기에서 나오는 소리를 음이라고 하며 성이라 표현하지는 아니한다.

목으로 나오는 소리는 발성이라 하고 초성 중성 종성으로 소(素)를 분류한다.
발성기관에서 끝소리가 없이 발성되는 경우도 있다.
지금까지 목소리를 해부적으로 구분하여 초성 중성 종성의 글자로 설계하고 제작한 경우는 한글뿐이며 한글은 11,172개로 되어 있다.
이러한 목소리는 19개의 첫소리와 21개의 중간소리 27개의 끝소리로 조립되어 구성하며 합하면 11,172개로 조립되어 나타난다는 것이다.
또 14개의 기본자성소에 파생자성소 5개가 더해져서 19개의 초성자소가 설계 제작되었고, 14개의 기본자성소에 파생자성소 13개가 합쳐서 종성자소 27개가 설계 제작되었고, 10개의 기본모성소로 파생모성소 11개가 더해져서 21개의 모성자소를 설계 제작하였었다.
이렇게 하여 인체발성기관인 성대에서 생성되는 목소리의 성소 51개가 11,172개의 발성소를 조립하고 발성글자를 조립한다.
한글은 이러한 발성소를 글자로 하여 사용하고 있게 된 것이다.
한글의 자소는 51개이며 세분하면 아래와 같다.

기본자성소 14 ; ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ
파생자성소 16 ; ㄲ ㄳ ㄵ ㄶ ㄸ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅃ ㅄ ㅆ ㅉ

총 30개자성소 중
초성자성소 19 ; ㄱ ㄲ ㄴ ㄷ ㄸ ㄹ ㅁ ㅂ ㅃ ㅅ ㅆ ㅇ ㅈ ㅉ ㅊ ㅋ ㅌ ㅍ ㅎ
종성자성소 27 ; ㄱ ㄲ ㄳ ㄴ ㄵ ㄶ ㄷ ㄹ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅁ ㅂ ㅄ ㅅ ㅆ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ
기본모성소 10 ; ㅏ ㅑ ㅓ ㅕ ㅗ ㅛ ㅜ ㅠ ㅡ ㅣ
파생모성소 11 ; ㅐ ㅒ ㅔ ㅖ ㅚ ㅘ ㅝ ㅟ ㅢ ㅙ ㅞ
총 21개모성소 중
2중모성소 9 ; ㅐ ㅒ ㅔ ㅖ ㅚ ㅘ ㅝ ㅟ ㅢ
3중모성소 2 ; ㅙ ㅞ

로 구분되어진다.
2진법의 반도체상에서도 자성소 30개와 모성소 21개를 합한 51개의 자소로 11,172개의 글자를 조립한다.

초성과 중성으로 조립되는 글자는 399개
19 × 21 = 399

초성 중성 종성으로 조립되는 글자는 10,773개
19 × 21 × 27 = 10,773

한글자소로서 조립되는 총 글자는 11,172개
(19 × 21) + (19 × 21 × 27) = 11,172개

인체의 목소리는 성대에서 만들어지며 발성기관별 소리를 해부하면 표와 같이 분류할 수 있다.

| | | 입술소리 | 혀끝소리 | 입천장소리 | 여린입천장소리 | 목청소리 |
|---|---|---|---|---|---|---|
| 안울림소리 | 파열음 | ㅂ ㅃ ㅍ | ㄷ ㄸ ㅌ | | ㄱ ㄲ ㅋ | |
| | 파찰음 | | | ㅈ/ㅉ/ㅊ | | |
| | 마찰음 | | ㅅ ㅆ | | | ㅎ |
| 울림소리 | 비음 | ㅁ | ㄴ | | ㅇ | |
| | 유음 | | ㄹ | | | |

| 조음 위치 조음 방식 | | | 양 순 음 | 전 설 음 | 후 설 음 | 성 문 음 |
|---|---|---|---|---|---|---|
| 장 애 | 폐 쇄 음 | 평음 | ㅂ | ㄷ | ㄱ | |
| | | 경음 | ㅃ | ㄸ | ㄲ | |
| | | 유기음 | ㅍ | ㅌ | ㅋ | |
| | 파 찰 음 | 평음 | | ㅈ | | |
| | | 경음 | | ㅉ | | |
| | | 유기음 | | ㅊ | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 음 | 마찰음 | 평음 | | ㅅ | | |
| | | 경음 | | ㅆ | | |
| | | 유기음 | | | | ㅎ |
| 공명음 | 비 음 | | ㅁ | ㄴ | ㅇ | |
| | 유 음 | | | ㄹ | | |

| 혀의 앞 뒤 위치 | 전 설 모 음 | | 중 설 모 음 | | 후 설 모 음 | |
|---|---|---|---|---|---|---|
| 입술 모양 혀의높이 | 평 순 | 원 순 | 평 순 | 원 순 | 평 순 | 원 순 |
| 고 모 음 | 이 [i] | | 으 [□] | | | 우[u] |
| 반 고 모 음 | 에 [e] | | 어 [□] | | | 오[o] |
| 반 저 모 음 | 애 [□] | | | | 어 [ʌ] | |
| 저 모 음 | | | 아 [a] | | | |

| 구 분 | | | 전설평순 | 후설평순 | 후설원순 |
|---|---|---|---|---|---|
| 상 승 | y 계 | 고 모 음 | | | 유 yu |
| | | 중 모 음 | 예 ye | 여 y□ | 요 yo |
| | | 저 모 음 | 얘 y□ | 야 ya | |
| | w 계 | 고 모 음 | 위 wi | | |
| | | 중 모 음 | 웨 we | 워 w□ | |
| | | 저 모 음 | 왜 w□ | 와 wa | |
| 하 강 | y 계 | 고 모 음 | | 의 iy | |

**한글은 인체목소리의 성소글자로 소리를 조립하고 글자로 나타낼 수 있어서 말과 글이 일치하여 발성기호가 필요 없다.**
**따라서 이를 정의하면 조립된**

**11,172개 발성소 = 11,172개 조립글자 = 11,172개 발성기호**

**라는 등식이 성립한다.**
**소리와 글자 그리고 발성기호가 일치하면 인터넷사이버 공간에서는 일치하지 않은 자소와는 자기정보의 입력전달과 판단해석에 엄청난 차이가 발생한다.**
**이것이 최적화된 반도체적성글자로 환상적인 매력을 갖는 첫 번째 이유이다.**

**인터넷사이버 공간에서 자기정보는 입력과 실행속도가 생명력이라고 한다면 성소정보이던 자소정보이던 초중종성의 수평수직조립방식이어야 입력속도가 빠르고 획득정보의 판단해석속도 또한 빠르다.**

이것이 최적화된 반도체적성글자로 환상적인 매력을 갖는 두 번째 이유이다.

11,172개의 한글자소는 ‘숫자나 문장부호와 확연히 구분’ 된다.
한글11,172개의 글자에는 구분이 불확실한 글자가 없다.
알파벳자소에서는 책을 읽거나 자소 판으로 글을 입력할 때 구분이 불확실한 경우를 경험하게 되지만 한글에서는 한글과 숫자 부호 또는 한글과 한글사이에 구분이 되지 아니하는 경우가 없다.
이를 가독성이라 하고 따라서 한글은 실행되었을 때 즉, 가독성에서 확실하게 구분된다.
이것이 최적화된 반도체적성글자로 환상적인 매력을 갖는 세 번째 이유이다.

반도체기기인 컴퓨터가 등장하면서 인간의 글자사용에 있어서 크게 달라진 것으로는 글자를 필기하는 것이 아닌 선택한다는 것이고
필기체가 사라진 대신에 한글은 한글대로 128개의 서체를 자유자제로 선택하여 사용할 수 있고 알파벳으로는 무려 320개의 서체를 사용할 수 있다는 것이다.
따라서 컴퓨터시대에는 글씨를 쓰는 것이 아니라 글자를 128개의 서체 혹은 320개의 서체로 선택한다는 것이 된다.

인터넷사이버를 위한 반도체적성글자에는 글자의 소에 따라 2 가지의 방법으로 분류할 수 있다.
하나는 낱개자소(낱 자소)와 조립자소이며
낱개자소는 자소 자체가 글자라는 것이고
조립자소는 자소가 부품과 같은 것으로 부품끼리 조립되어 기계가 되듯이 2개 이상의 자소가 조립되어 글자로 되는 것이다.

다른 하나는 자소가 바로 입력되느냐 전환하여 입력되느냐에 따라 '직접 바로 입력하는 자소(직접자소, 바로자소, 직 자소) 와 전환하여 입력하는 자소(전환자소)'로 구분할 수가 있다.

반도체기기인 컴퓨터는 “조립 자소로 된 바로자소” 를 가장 최적화된 것으로 받아들인다.
한글은 직접(바로) 입력되는 조립자소이다.

이렇게 낱자소와 조립자소의 대표적인 것으로는 알파벳은 낱개자소이고 중국글자와 한글은 조립되는 조립자소에 해당하지만 바로자소에는 알파벳과 한글을 들 수 있고 전환자소에는 중국글자를 들 수 있지만 중국글자는 반도체 글자로는 2 중성이 있다.
두 가지로 분류되는 자소에 따라 입력전달속도에는 엄청난 차이가 난다.
타자기로 글자소를 선택하여 두들겨 인쇄기처럼 글자를 종이에 인쇄하던 시대에는 자소가 많으면 글쇠를 배열할 수 없어서 타자기로는 사용이 불가능하였으나 반도체에서는 프로그램화하여 내장하고 전환하는 방법이 가능해짐으로 이제 어떤 글자도 컴퓨터를 통한사용이 활발해졌다.
다만 이렇게 하여 사용되는 전환자소일 때에는 전환시간으로 인하여 입력전달속도가 느려진다.

**컴퓨터에 빠른 입력전달속도를 가진 자소는 조립자소에 바로자소이라는 것은 쉽게 짐작이 간다.**
**한글은 어떤 자소인가 하면 조립자소에 바로자소이다.**

4 한글자소와 알파벳자소의 특성
반도체로 사용하는 현대한글에 대한 바른 이해를 하기 위해서는 먼저 2진법 반도체 체계를 알아야 한다.
2진법을 알게 되면 정녕 한글은 반도체를 위하여 만들어진 아름다운글자라는데 새삼 놀라게 된다.
반도체로 만들어진 인터넷사이버는 2진법의 체계로 되어 있고 이를 프로그램화하여 여러 반도체기기에 접목한 것이다.
이러한 반도체기기에는 컴퓨터, RFID, 휴대전화기 등의 단말기를 통하여 이용된다.
글자로 입력되어 정확하게 판단하고 해석되도록 전달되기 위해서는 말과 글이 일치하고 자소들끼리 구분이 확실하고 간단하면서도 입력속도가 빨라야 하는 것이다.
초기 인터넷사이버에 필요한 운영체제가 알파벳자소에 의해서 개발되고 발전될 수밖에 없었던 것은 사실이지만 성숙단계에 들어서면 구조적으로 바로자소이며 조립자소 중에 하나가 그 주체가 될 수밖에 없다고 본다면 이는 한글에 해당한다.

왜 ? 그러한가는 뒤에 설명의 기회를 갖기로 하고 성숙단계에서 한글이 인터넷 사이버의 운영체제 중심이 되는 이유부터 설명하기로 하자.
한글은 초성 중성 종성으로 생성되는 인체공학적인 발성원리를 기초로 하여 목소리의 성소(聲素)를 자소(字素)로 설계제작 했다.
14개 기본자성에 파생자성 16개를 더하여 30개 자성자소를 초성에 19개로 설계 제작하였고, 종성으로 27개로 설계 제작하였다.
10개의 기본모성에 파생모성 11개를 더한 21개의 모성자소는 2중 모성 9개 3중 모성 2개로 한글의 총 자소는 51개로 설계하고 제작되었다.
그래서 초성과 중성으로만 조립되어 종성 없는 글자는 399개가 된다.
399개의 글자에 27개의 종성을 조립하면 10,773개의 글자가 조립되고 종성 없는 글자 399개와 합하면 11,172개의 글자가 조립된다.
따라서 조립된 한글의 수는 11,172개가 되고 기본자소 24개와 파생자소 27개를 합한 51개 자소까지 하여 11,223개가 된다.

반도체기기에서 자소 판에서 자소를 선택하여 입력하는 것으로 양손의 손가락 10개를 사용하여 자소를 선택하는 방법과 양손의 엄지손가락으로 선택하는 자소선택방법은 다르게 하여야 효율적이다.
어떤 기기는 한 손가락만으로 선택하기도 한다.
이동전화기와 같은 적은크기로 휴대하기 위해서는 자소입력단추가 숫자 10개에 *와 #을 더한 12개의 단추로 한정되어야 한다.
따라서 24개의 자소는 *의 획 추가로 만들어져야 한다.
중간소리는 “ㅣ, ㅡ”에 “ · ”으로 하나 혹은 둘을 상하좌우에 추가하면 되기도 한다.
자음의 자소는 “ㄱ, ㄴ, ㅁ, ㅅ, ㅇ”에 획의 추가하는 방법으로 ㅋ, ㄷ, ㅌ, ㅂ, ㅍ, ㅈ, ㅊ, ㅎ 을 만들지는 글자로 조립할 수 있고 ㄹ 만 그대로 선택하여 사용한다.
쌍자성은 #를 사용하여 중복시켜 사용할 수 있다
이런 경우는 9개의 단추로 11,172개의 글자를 표현해 내는 것이 된다.
다른 방법도 있다.
자성의 자소는 앞에서와 같이 “ㄱ, ㄴ, ㅁ, ㅅ, ㅇ”에 획의 추가하는 방법으로 ㅋ, ㄷ, ㅌ, ㅂ, ㅍ, ㅈ, ㅊ, ㅎ 을 만들어 글자를 조립할 수 있어서 ㄹ 과 합하여 추가하거나 중복선택으로 사용하고, 모성은 다른 방법인 “(ㅏ,ㅓ) 와 (ㅗ,ㅜ)” 선택하는 횟수로 즉, 1회 선택은(ㅏ, ㅗ)가 되고 2회 중복선택 하면 (ㅓ, ㅜ)가 되도록 하는 방법으로도 11,172개의 전 한글이 조립되어 표현하는 방법이 있다.
이런 경우는 10개의 단추가 필요하다.
한글자소는 한손으로 선택하는 방법으로도 글자를 바로 입력된다.
알파벳자소도 바로 입력되기는 한다.
그러나 숫자 10개 단추에 자소를 배열하면 2.6개의 자소가 되고 대소단추까지 배열하려면 하나의 단추 3 ~ 4개의 자소가 배당되어 1 ~ 4회 단추를 중복으로 눌러야 자소가 바로 입력되어 입력속도가 매우 늦어지는 단점이 있다.
알파벳자소는 이 이외에도 반도체자소적성으로는 부적합한 것이 많이 발견된다.
알파벳의 경우 12번째 소문자인 글자 ‘l’ 과 숫자 ‘1’과의 구별이 되지 못하고 15번째 글자 ‘O' 와 숫자 ‘0’ 과의 구별이 되지 않는다.
조명이 어두우면 ‘f' 와 't'의 구별이 어려우며 ’ S ‘와 ’ 5 ‘의 구별이 어려우며 'q'와 ’ g'도 구별이 어렵다. ‘i' 와 't'의 구별도 어렵고 'b, d, q, p,'는 뒤집고 돌리면 같은 형태의 단순한 글자가 되어버린다.
이러한 관계로 조금만 어두워도 분별에는 어려움이 있어서 필요이상의 신경을 써야하는 글자가 된다.
일본의 글자나 알파벳문화권 일부에서는 한 글자에다 단순하게 점을 찍거나 표를 하여서 표현하는 방식은 한글자소 방법과는 대조적인 것이 된다.
이처럼 한글자소는 어떤 방법으로든 쉽게 구별되는 특징이 있다.
양손의 엄지로도 번갈아 가면서 쉽고 빠른 속도로 자소를 만들어 조립할 수 있다.
이렇게 반도체기기를 위하여 미리 설계된 자소이면서 자소 와 자소의 구별이 확실해서 시력보호자소라고도 할 수 있어서 놀라게 된다.
가독성이 가장 좋은 서체를 굴림체라고 하며 굴림체에서는 가독성을 높이기 위해서 숫자와 부호 그리도 다른 문화권의 글자와 확연한 구분이 되는 것을 필요로 한다.
최근 글자의 선택방법이 단추(button)식에서 접촉(touch)식으로 발전하여 더욱 편리하게 되었다.

2000년대 들어와서 사용하는 현대한글은

기본자음소 14 ; ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ
파생자음소 16 ; ㄲ ㄳ ㄵ ㄶ ㄸ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅃ ㅄ ㅆ ㅉ
기본모음소 10 ; ㅏ ㅑ ㅓ ㅕ ㅗ ㅛ ㅜ ㅠ ㅡ ㅣ
파생모음소 11 ; ㅐ ㅒ ㅖ ㅖ ㅚ ㅘ ㅝ ㅟ ㅢ ㅙ ㅞ
으로서 2012년 7월 7일 필자에 의해 글자의 구성이 수직수평조립글자로 최첨단반도체 적성글자임이 밝혀졌다.

따라서 반도체기기에 의한 글자 사용이 주류를 이루면서 글자의 분류방식도 반도체기기방식에 맞도록 바뀌어야한다는 필요성 느끼게 된다.

반도체기기 사용이전에 한글을 비롯하여 알파벳글자들이 활자 인쇄방법 이외 타자기로서 기계적으로 글자를 찍어서 사용한 때가있었다. 이러한 시대에도 중국글자의 경우는 타자기로는 도저히 사용이 불가능한 글자들이어서 오로지 손으로 필기하는 방법뿐이었다.

언어학자들이 지난 시대에 '표의글자와 표음글자' 라고 하는 글자의 분류방법을 사용하고 있었지만 반도체기기 사용이 시작되면서 이러한 분류방법은 의미가 없어지게 되어버렸다.

오늘날에는 글자가 반도체기기 사용적성에 여부에 근거하여 분류하는 방법이 더 효과적이라고 할 수 있다.

반도체기기 사용적성의 분류방법이란 글자입력과 실행 관계가 어떠한 경로로 이루어지는가에 따라 분류하는 방식으로 반도체기기에 글자를 입력하고 실행면상으로 이를 읽어내기 위해서는 필수적으로 갖추어야 하는 하드웨어(hardware), 작동체제(operating system), 프로그램(program)이라는 운영체제가 반도체기기에 내장되어야 하고 글자를 입력하는 장치와 입력된 글자가 실행 면에 나타나거나 출력기를 통하여 문서로 생성되어 나오는 장치도 필요하다

이 시대는 정보수단이 반도체로 구성한 컴퓨터사용이 필수적이기 때문에 인간이 사용하는 말과 글도 반도체에 기준을 두고 연구되고 개발되어야 한다.

따라서 반도체를 구성한 언어속성을 알기 위해서는 인간이 사용해온 수의체계인 10진법에서 2진법의 체계도 이해를 할 수 있어야 한다.

지금부터는 반도체를 운용하는 2진법체제에 대해서 알아보기로 하자.

한글이 국제화 컴퓨터에 정상적으로 사용되기까지는 세계 어느 나라의 언어보다 많은 진통을 겪은 것으로 기록되어 있다.

2013년 현재까지 유니코드에 인코딩된 글자 중 중국글자와 같이 가장 많은 글자 값을 가진 자소로서 16진법으로 확장하여 사용할 경우

$2^{1}$ □ = 65,536이므로

65,536개의 글자 값을 인코딩할 수 있으나 중국글자 50,000개에 옛한글까지 합한 한글

이 26,000개가 되니 두 나라 글자 값이 76,000개가 필요하다.

프로그래머들은 중국 글과 한글의 76,000개의 글자 값을 어떻게 처리하여 운용하였을까 하는 것이 궁금해질 것이다.

일반적이고 단순한 생각으로 한글이 24개의 글자로 된 것이라서 반도체기기인 컴퓨터 글자로서 가장 이상적이라고 설명하다가는 크게 낭패를 당하게 된다.

컴퓨터세계화를 위한 개발에 있어서 프로그래머들에게 한글의 존재와 중국 글자는 정말 큰 부담을 주어온 것이다.

그러니 프로그래머들은 한글 세계화가 곧 컴퓨터세계화라는 것에 크게 비중을 두어 온 것에는 부정할 수 없는 사실되었다고 할 수가 있다.

중국글자 50,000개의 글자는 한글과 같은 순수한 수직수평조립글자이었지만 반도체기기에서 한글의 자소부품이 51개인데 반해 무려 277개의 부품에 해당하여 자소(字素)로서는 도저히 입력방법을 찾지 못하여 결국 전환글자로 처리할 수밖에 없는 조립글자의 소질을 가지고 있으면서도 전환자소로 된 글자이이다.

중국은 글자정책에서 간자체를 개발하고 알파벳을 가져다가 변환하여 입력하는 글자이어서 반도체기기상으로는 알파벳글자로 보아야 한다.

알파벳으로 변환하여 입력하는 글자는 반도체기기상으로는 모두 낱개자소인 알파벳자로라고 하여야 한다는 것이다.

이처럼 글자의 분류방법이 필기방법에서 선택방법으로 바꾸어지면서 즉, 글자는 종이위에 필기하는 글자에서 반도체기기의 자판에서 모니터로 실행하는 선택하는 방법이 도입되면서 글자의 분류방법도 달라졌다는 것을 알게 되고 정보 전달이라는 속도에서 반도체적성을 갖추지 못하면 글자의 사용이 불편하다는 것을 알 수가 있다.

## 5 초기단계의 글자사용 컴퓨터프로그램 개발

컴퓨터는 초기 개발단계에서는 계산기로 시작했다.

기록에 의하면 아스키코드가 1967년에 시작하여 1986년 마지막 개정이 있었고 컴퓨터세계화를 위한 줄기찬 개발로 1991년 8월 유니코드1.0.0으로 등장하여 1993년 6월 유니코드 1.1.0에 조합형 한글로서 처음으로 인코딩이 시작하고 1996년 6월 유니코드 2.0.0에 11,172자소의 현대한글로 인코딩되었다.

컴퓨터세계화를 위한 개발과정에서 한글을 '조합형 한글' 이니 '완성형 한글' 이니 하여 분류하고 취급하여 왔으니 한글학자들에게는 그동안 넋을 놓고 있었다는 것이 된다.

이는 극히 당연한 것으로 한글학자들이 반도체기기인 컴퓨터 프로그램세계를 이해할리 없으며 설사 이해한다고 하드라도 고급물리학의 세계이어서 쉽게 접근이 불가능하기 때문이다.

유니코드의 역사

1991년 8월: 유니코드 1.0.0 — 유니코드의 시작
1992년 6월: 유니코드 1.0.1 — 2만자가 넘는 한자가 포함되었다.
1993년 6월: 유니코드 1.1.0 — 조합형 한글을 위한 첫가끝 코드가 포함되었다.
1995년 6월: 유니코드 1.1.5
1996년 6월: 유니코드 2.0.0 — 11172자의 모든 현대 한글이 포함되었다.
1998년 5월: 유니코드 2.1.2
1998년 8월: 유니코드 2.1.5
1998년 12월: 유니코드 2.1.8
1999년 4월: 유니코드 2.1.9
1999년 9월: 유니코드 3.0.0
2000년 8월: 유니코드 3.0.1
2001년 3월: 유니코드 3.1.0 — 보조 평면에 처음으로 문자들이 포함되었다.
2001년 8월: 유니코드 3.1.1
2002년 3월: 유니코드 3.2.0
2003년 4월: 유니코드 4.0.0
2004년 3월: 유니코드 4.0.1
2005년 3월 31일: 유니코드 4.1
2006년 6월 14일: 유니코드 5.0 (문자 데이터베이스는 7월 18일에 발표되었으나 책은 11월 9일 출시)
2008년 4월 4일: 유니코드 5.1
2009년 10월 1일: 유니코드 5.2 — 확장 첫가끝 코드가 포함되었다.
2010년 10월 11일: 유니코드 6.0
2012년 2월 1일: 유니코드 6.1

## 한글 첫 가 끝 코드

첫소리-가운뎃소리-끝소리 글자부호화 방식(Syllable-Initial-Peak-Final Encoding Approach) 한글 부호계[1], 또는 줄여서 첫가끝 코드란 유니코드의 한글 자모 영역(U+1100~U+11FF)을 첫소리-가운뎃소리(-끝소리)(-방점) 순서로 배열하는 방법을 말한다.

컴퓨터 한글화개발이 곧 컴퓨터세계화 개발이라는 것으로 그 원인을 알려면 2진법을 알아야 하고 또 2진법체제에서만 설명되어진다.
우리가 일상생활에서 사용하는 글자를 프로그래머들은 문자열이라 하고 글자 값으로 처리하여 입력하고 실행하여 전달되기 위해서는 글자 값으로 수치화하고 계산과정으로 취급해야 한다.

모든 정보를 수치화하고 값으로 계산하여 따진다는 것이다.
컴퓨터는 글자를 값으로 계산하는 계산기로 보는 것이다
컴퓨터에서 모든 값은 2진법으로 처리된다.
문자열을 2진법체제 글자 값으로 취급한 첫 시도는 '7계단' 아스키코드이다.

2□ = 128

로서 128개의 코드를 배열할 수가 있다.
초기 컴퓨터 글자 값은 7계단 7비트정보 값으로 운용되었다.
영문 알파벳을 사용하는 대표적인 글자 인코딩이다. 아스키는 컴퓨터와 통신 장비를 비롯한 글자를 사용하는 많은 장치에서 사용되며, 대부분의 글자 인코딩이 아스키에 기초를 두고 있다.

아스키는 1967년에 표준으로 제정되어 1986년에 마지막으로 개정되었다. 아스키는 7비트 인코딩으로, 33개의 출력 불가능한 제어 문자들과 공백을 비롯한 95개의 출력 가능한 글자들로 이루어진다. 제어 문자들은 역사적인 이유로 남아 있으며 대부분은 더 이상 사용되지 않는다. 출력 가능한 글자들은 52개의 영문 알파벳 대소글자와, 10개의 숫자, 32개의 특수 문자, 그리고 하나의 공백 문자로 이루어진다.

아스키가 널리 사용되면서 다양한 아스키 기반의 확장 인코딩들이 등장했으며, 이들을 묶어서 아스키라고 부르기도 한다. 대표적으로 7비트 인코딩을 유지한 머리말과, 원래 아스키코드 앞에 비트 0을 넣어 8비트 인코딩을 만든 IBM 코드 페이지와 ISO 8859가 있다. 이 인코딩들은 언어 군에 따라 같은 숫자에 서로 다른 글자가 배당된 경우가 많다.
아스키코드에서는 7비트단위로 128개의 코드생성이 가능하여 반도체기기사용에 관한 운영체제가 개발된다.
이러한 반도체 개발과정을 이해하지 않고는 한글의 반도체글자로서의 속성을 알지 못한다.
2진법에서 '2' 라는 수는 사용할 수 없다.
오로지 ' 0 과 1'로 구성될 뿐이다.
그러하다면 아스카코드에서는 7계단의 수로서 가장 큰 수는 111 1111이 된다.
참고로 관련되는 용어로 비트(bit)는 컴퓨터용어로서는 두 가지의 값을 말하며
(1) 정보량의 최소단위.
(2) 2진법에서의 0 또는 1
2진법은 binary digit(=이진수(二進數))로서 binary는 컴퓨터용어로서 2진(법)의, 2진수의 체계이다.

1. 『컴』 정보량의 최소 기본 단위. 양자(兩者)택일의 꼴로 정리된 정보의 누적(累積) 단위. 8비트는 1바이트임.
2. 이진법에서 쓰는 숫자로, 0 과 1.

byte는 컴퓨터용어로서 8비트로 된 정보 단위

지금의 컴퓨터라는 반도체기기에서는 일부 문화권의 글자는 타자기를 사용하는 때와 같이 많은 글자로 인해서 7비터어로는 사용이 불가능한 때가 있었다.
이래서 초기 반도체기기개발 글자로 알파벳자소로 시작하였으며 한글로는 불가능한 이유이기도 한 것이다.
아래는 7비트 아스키코드이다.

| 이진법 | 팔진법 | 십진법 | 십육진법 | 약자 | 설명 | 한글설명 |
|---|---|---|---|---|---|---|
| 000 0000 | 000 | 0 | 00 | NUL | Null | 공백 문자 |
| 000 0001 | 001 | 1 | 01 | SOH | Start of Header | 헤더 시작 |
| 000 0010 | 002 | 2 | 02 | STX | Start of Text | 본문 시작, 헤더 종료 |
| 000 0011 | 003 | 3 | 03 | ETX | End of Text | 본문 종료 |
| 000 0100 | 004 | 4 | 04 | EOT | End of Transmission | 전송 종료, 데이터 링크 초기화 |
| 000 0101 | 005 | 5 | 05 | ENQ | Enquiry | 응답 요구 |
| 000 0110 | 006 | 6 | 06 | ACK | Acknowledgment | 긍정응답 |
| 000 0111 | 007 | 7 | 07 | BEL | Bell | 경고음 |
| 000 1000 | 010 | 8 | 08 | BS | Backspace | 백스페이스 |
| 000 1001 | 011 | 9 | 09 | HT | Horizontal Tab | 수평 탭 |
| 000 1010 | 012 | 10 | 0A | LF | Line feed | 개행 |
| 000 1011 | 013 | 11 | 0B | VT | Vertical Tab | 수직 탭 |
| 000 1100 | 014 | 12 | 0C | FF | Form feed | 다음 페이지 |
| 000 1101 | 015 | 13 | 0D | CR | Carriage return | 복귀 |
| 000 1110 | 016 | 14 | 0E | SO | Shift Out | 확장문자 시작 |
| 000 1111 | 017 | 15 | 0F | SI | Shift In | 확장문자 종료 |
| 001 0000 | 020 | 16 | 10 | DLE | Data Link Escape | 전송 제어 확장 |
| 001 0001 | 021 | 17 | 11 | DC1 | Device Control 1 | 장치 제어 1 |
| 001 0010 | 022 | 18 | 12 | DC2 | Device Control 2 | 장치 제어 2 |
| 001 0011 | 023 | 19 | 13 | DC3 | Device Control 3 | 장치 제어 3 |
| 001 0100 | 024 | 20 | 14 | DC4 | Device Control 4 | 장치 제어 4 |
| 001 0101 | 025 | 21 | 15 | NAK | Negative Acknowledgement | 부정응답 |
| 001 0110 | 026 | 22 | 16 | SYN | Synchronous idle | 동기 |
| 001 0111 | 027 | 23 | 17 | ETB | End of Transmission Block | 전송블록 종료 |
| 001 1000 | 030 | 24 | 18 | CAN | Cancel | 무시 |
| 001 1001 | 031 | 25 | 19 | EM | End of Medium | 매체 종료 |
| 001 1010 | 032 | 26 | 1A | SUB | Substitute | 치환 |
| 001 1011 | 033 | 27 | 1B | ESC | Escape | 제어기능 추가 |
| 001 1100 | 034 | 28 | 1C | FS | File Separator | 파일경계 할당 |
| 001 1101 | 035 | 29 | 1D | GS | Group Separator | 레코드 그룹경계 할당 |
| 001 1110 | 036 | 30 | 1E | RS | Record Separator | 레코드 경계 할당 |
| 001 1111 | 037 | 31 | 1F | US | Unit Separator | 장치 경계 할당 |
| 010 0000 | 040 | 32 | 20 | □ | | |
| 010 0001 | 041 | 33 | 21 | ! | | |
| 010 0010 | 042 | 34 | 22 | " | | |
| 010 0011 | 043 | 35 | 23 | # | | |

| | | | | |
|---|---|---|---|---|
| 010 0100 | 044 | 36 | 24 | $ |
| 010 0101 | 045 | 37 | 25 | % |
| 010 0110 | 046 | 38 | 26 | & |
| 010 0111 | 047 | 39 | 27 | ' |
| 010 1000 | 050 | 40 | 28 | ( |
| 010 1001 | 051 | 41 | 29 | ) |
| 010 1010 | 052 | 42 | 2A | * |
| 010 1011 | 053 | 43 | 2B | + |
| 010 1100 | 054 | 44 | 2C | , |
| 010 1101 | 055 | 45 | 2D | - |
| 010 1110 | 056 | 46 | 2E | . |
| 010 1111 | 057 | 47 | 2F | / |
| 011 0000 | 060 | 48 | 30 | 0 |
| 011 0001 | 061 | 49 | 31 | 1 |
| 011 0010 | 062 | 50 | 32 | 2 |
| 011 0011 | 063 | 51 | 33 | 3 |
| 011 0100 | 064 | 52 | 34 | 4 |
| 011 0101 | 065 | 53 | 35 | 5 |
| 011 0110 | 066 | 54 | 36 | 6 |
| 011 0111 | 067 | 55 | 37 | 7 |
| 011 1000 | 070 | 56 | 38 | 8 |
| 011 1001 | 071 | 57 | 39 | 9 |
| 011 1010 | 072 | 58 | 3A | : |
| 011 1011 | 073 | 59 | 3B | ; |
| 011 1100 | 074 | 60 | 3C | < |
| 011 1101 | 075 | 61 | 3D | = |
| 011 1110 | 076 | 62 | 3E | > |
| 011 1111 | 077 | 63 | 3F | ? |
| 100 0000 | 100 | 64 | 40 | @ |
| 100 0001 | 101 | 65 | 41 | A |
| 100 0010 | 102 | 66 | 42 | B |
| 100 0011 | 103 | 67 | 43 | C |
| 100 0100 | 104 | 68 | 44 | D |
| 100 0101 | 105 | 69 | 45 | E |
| 100 0110 | 106 | 70 | 46 | F |
| 100 0111 | 107 | 71 | 47 | G |
| 100 1000 | 110 | 72 | 48 | H |
| 100 1001 | 111 | 73 | 49 | I |
| 100 1010 | 112 | 74 | 4A | J |
| 100 1011 | 113 | 75 | 4B | K |
| 100 1100 | 114 | 76 | 4C | L |
| 100 1101 | 115 | 77 | 4D | M |
| 100 1110 | 116 | 78 | 4E | N |
| 100 1111 | 117 | 79 | 4F | O |
| 101 0000 | 120 | 80 | 50 | P |
| 101 0001 | 121 | 81 | 51 | Q |
| 101 0010 | 122 | 82 | 52 | R |
| 101 0011 | 123 | 83 | 53 | S |
| 101 0100 | 124 | 84 | 54 | T |
| 101 0101 | 125 | 85 | 55 | U |
| 101 0110 | 126 | 86 | 56 | V |
| 101 0111 | 127 | 87 | 57 | W |
| 101 1000 | 130 | 88 | 58 | X |

| | | | | | | |
|---|---|---|---|---|---|---|
| 101 1001 | 131 | 89 | 59 | Y | | |
| 101 1010 | 132 | 90 | 5A | Z | | |
| 101 1011 | 133 | 91 | 5B | [ | | |
| 101 1100 | 134 | 92 | 5C | ₩ | | |
| 101 1101 | 135 | 93 | 5D | ] | | |
| 101 1110 | 136 | 94 | 5E | ^ | | |
| 101 1111 | 137 | 95 | 5F | _ | | |
| 110 0000 | 140 | 96 | 60 | ` | | |
| 110 0001 | 141 | 97 | 61 | a | | |
| 110 0010 | 142 | 98 | 62 | b | | |
| 110 0011 | 143 | 99 | 63 | c | | |
| 110 0100 | 144 | 100 | 64 | d | | |
| 110 0101 | 145 | 101 | 65 | e | | |
| 110 0110 | 146 | 102 | 66 | f | | |
| 110 0111 | 147 | 103 | 67 | g | | |
| 110 1000 | 150 | 104 | 68 | h | | |
| 110 1001 | 151 | 105 | 69 | i | | |
| 110 1010 | 152 | 106 | 6A | j | | |
| 110 1011 | 153 | 107 | 6B | k | | |
| 110 1100 | 154 | 108 | 6C | l | | |
| 110 1101 | 155 | 109 | 6D | m | | |
| 110 1110 | 156 | 110 | 6E | n | | |
| 110 1111 | 157 | 111 | 6F | o | | |
| 111 0000 | 160 | 112 | 70 | p | | |
| 111 0001 | 161 | 113 | 71 | q | | |
| 111 0010 | 162 | 114 | 72 | r | | |
| 111 0011 | 163 | 115 | 73 | s | | |
| 111 0100 | 164 | 116 | 74 | t | | |
| 111 0101 | 165 | 117 | 75 | u | | |
| 111 0110 | 166 | 118 | 76 | v | | |
| 111 0111 | 167 | 119 | 77 | w | | |
| 111 1000 | 170 | 120 | 78 | x | | |
| 111 1001 | 171 | 121 | 79 | y | | |
| 111 1010 | 172 | 122 | 7A | z | | |
| 111 1011 | 173 | 123 | 7B | { | | |
| 111 1100 | 174 | 124 | 7C | \| | | |
| 111 1101 | 175 | 125 | 7D | } | | |
| 111 1110 | 176 | 126 | 7E | | | |
| 111 1111 | 177 | 127 | 7F | DEL | Delete | 삭제 |

**이상 살펴본 바와 같이 초기단계의 컴퓨터프로그램으로 개발되기 위한 자소의 자격으로는**

**2□ = 128**

**의 7비트 128개 글자 값을 가진 자소체계이어야 한다는 것을 알 수 있다.**

**즉, 낱개자소이며 바로자소이어야 하는 것으로 128개 자소 코드를 만족시켜야 한다는 것이다.**

**비록 한글은 알파벳의 글자소가 52개인데 반해 1개가 적은 51개의 글자소이나 알파벳은 낱개자소이고 한글은 조립자소이어서 자소 51개와 조립글자 11,172를 합한 11,223개**

의 글자코드를 128개의 자소코드인 2□ = 128개로 하는 2진법을 만족시킬 수가 없어서 초기개발글자가 아니라는 것을 알 수가 있다.

한글이 반도체기기에서 사용되기 위해서는 2진법을 $2^1$ □ = 65,536의 16진법으로 확장된 기술이 개발된 이후라야 비로소 사용된다는 것을 알 수가 있다.

기본 자소는 한글이 알파벳 26개보다 2개가 적은 24개이고 알파벳의 대소글자 52개보다 한글파생자소가 51개로 1개가 적지만 알파벳은 낱개자소이고 한글은 수직수평으로 조립되는 자소이어서 반도체기기의 개발 초기단계에서 한글의 호환성을 유지시키기 위해 무수히 노력했으나 근본적으로 11,223개의 글자코드 값이 없어 깨어지는 경우가 허다함을 경험하여온 것이다.

지금도 운영체제가 맞지 아니하여 깨어지는 글자를 경험하게 된다.

필자는 2013년 현재수준에서는 컴퓨터의 운영체제가 한글을 기준하여 개발하여 바꾸는 것이 세계반도체 미래를 위해 반드시 수행하여야 할 과정이라고 생각한다.

따라서 기계언어도 한글체제로 개발하여 사용하여야 한다고 본다.

현재의 대한민국의 반도체 수준이면 충분하게 개발할 수 있는 능력을 보유하고 있다고 생각되기 때문에 한국과학의 미래를 내다보며 아낌없는 투자가 일어 날 것으로 본다.

## 6 반도체기기의 세계화와 유니코드

한글이 자체적으로 프로그램과 운용체제를 개발할 수 없었던 것은 당시 기술로는 11,172개의 자소를 소화시킬 수 없었기 때문이다.

알파벳 52개의 자소로 개발하기 시작한 7비트 128개 코드로 시작하여 세계 다국어를 사용할 수 있도록 개발하기 시작하여 개발된 것이 8비트를 8진법, 16진법으로 확대하여 개발한 것이 요즈음 사용하는 유니코드로서 $2^1$ □ = 65,536개의 코드영역까지로 확대된 것이다. 지금도 16진법에서 32진법 64진법으로 확대하여 새로운 운용프로그램을 개발하고 있다.

지금의 컴퓨터와 같은 반도체기기는 8계단비트를 바이트단위 문자열을 정보단위로 하여 사용하기 때문에 초기 아스키코드는 7계단바이트단위로는 128개의 문자열정보가 만들어짐으로 그 이상의 문자열정보를 만들지 않았다.

또 8계단8비트 바이트단위 글자 값으로 확장하더라도 256개까지 문자열정보가 생성되어 그 이상의 문자열정보는 입력이 불가능하게 된다.

따라서 알파벳문자열이외 더 이상의 국가 글자를 사용하기에는 256개 글자 값으로는 불가능하게 된다.

타자기나 반도체기기가 나오기 전 일부 문화권의 글자들은 인쇄체와 필기체로 구분하여 사용하기도 하였다.

알파벳글자도 이중의 하나로서 26개 대소글자에 인쇄체와 필기체로 구분하면 104개의 글자가 되어있었다.

중국의 글자 같은 경우는 동일한 글자를 해서 행서 초서로 3가지의 글자체로 사용하기

도 하였다.

글자는 기록하여 사용하는 정보매체이기 때문에 반도체 이전에는 이렇게 인쇄체와 필기체로 2 원화하여 사용하는 이유로는 중국의 그자는 붓으로 알파벳글자는 깃털을 깎아 만든 펜이라는 필기구를 사용하여 종이위에 글자를 손으로 쓰기 위해서 인쇄체처럼 또박 또박 한글자한글자를 기록하다보면 기록속도가 너무 느리고 시간이 많이 걸리는 불편한 점이 있었기 때문에 필기체글자가 만들어지게 되었다.

인쇄술과 기계공업이 발달하면서 인쇄기가 아닌 글자를 찍어내는 기계의 개발이 시도되어 글자를 두들긴다는 타자기도 개발되었지만 중국의 글자는 원천글자가 50,000에 달하고 상용글자가 7,000여개에 달하는 글자를 277개의 부수로 글쇠장치타자기방식으로는 사용이 전혀 불가능하고 오로지 필기구류에 의한 손으로 일일이 글자를 써야하는 방법뿐이었다.

타자기 이후 알파벳문화권에서 아스키코드(ASCII, American Standard Code for Information Interchange, 미국 정보 교환 표준 부호)에 의한 반도체기기를 개발하여 사용하였으나 앞서 말한 8계단의 정보단위에는 256개 문자열이상을 사용할 수 없는 한계에 이러자 유니코드(Unicode)를 제작하여 세계다국어를 모두 사용할 수 있게 하였다.

반도체기기에서 글자를 입력하게 되면 굳이 필기체의 글자를 사용할 필요성이 사라지고 인쇄체글자만으로 글자단추를 두들기거나 선택하여 입력할 수 있으므로 동일한 글자를 2원화하여 사용하던 것이 일원화로 된다.

즉, 필기체가 사라진 것이다.

컴퓨터와 같은 반도체기기는 8계단단위의 바이트를 사용하나면 '0, 1' 로 된 2진법에서는 가장 큰 수가 '11111111' 로서 더 이상은 표현방법으로서는 불가능하여 고안한 것이 유니코드방식이다.

유니코드방식에는 2진법에서 8진법, 16진법, 32진법,64진법으로 확대하고 수의 사용방법도 '0, 1' 에서 알파벳글자를 사용하여 수치사용영역을 확대시키었다.

이러한 이유로 고전적 글자 사용방법과 분류방법이 반도체기기사용시대에는 아무런 효용가치가 없어져 버렸다는 것을 알 수가 있고 반도체기기에는 반도체적성에 맞는 글자가 중요하다는 것을 인정해야한다.

반도체기기에서는 글자나 부호 그림 그리고 소리를 자유롭게 입력하고 실행할 수 있기 때문에 글자의 분류방식을 입력방식에 따라 분류하는 방법이 훨씬 효과적이다.

컴퓨터를 중심으로 글자입력방식을 분류한다면

① 글자입력방법에 따른 바로입력방식과 전환입력방식

② 글자의 소에 따라 조립자소 와 낱개 자소(낱 자소)가 된다.

반도체개발속도는 초기 알파벳문화권에서 말하는 소위 '무어의 법칙' 에서 '황의 법칙' 으로 발전하면서 대한민국이 최첨단반도체국가로 등장하게 되었다.

필자의 조사고찰에 의하면 알파벳문화권이 아닌 대한민국이 초기부터 한글을 반도체기기글자로 사용방법을 개발하였다면 얼마나 편리했을까하는 생각을 하게 된다.

왜냐하면?

한글의 글자는 1446년에 설계 제작되어 반포된 글자이지만 철저하게 첨단반도체글자적성을 만족시키고 탄생했다는 것에 있다.

그러나 반도체의 근간은 2진법체계이어서 처음부터 한글을 사용할 수 있는 $2^{1}$ □ = 65,536개의 코드영역으로 확대할 수는 없지만 위의분류방식에 의하면 가장 이상적인 반도체글자는 첫 째로 글자입력방식이 전환방식이 아닌 바로입력방식이어야 하고 다음으로는 자소가 조립자소이어야 한다는 것이다.

이러한 글이 한글이면서도 알파벳문화권에서 개발당시 유니코드에 적용하여 프로그램화 하여 사용해 온 것을 대한민국인은 아무 여과도 없이 수동적자세로 그대로 받아들여 사용만 하고 있었다.

이 부분에 대해서 대한민국의 국민의 한사람이라면 매우 자존심이 상하는 것이지만 어쩔 수 없는 사실이다.

차세대문화는 첨단반도체국가에 의해서 주도되고 흘러가게 된다는 것에는 누구도 의심하지 못하며 이론이 있을 수 없다.

그렇다면 반 천 년 전 1446년 철저하게 첨단반도체글자적성을 만족시키고 탄생한 한글에게 제자리를 찾아주어야 한다는 것이다.

## 7. 반도체적성의 한글문제

이쯤 하여 왜 초기 반도체기기의 운영체계가 한글로서 개발되고 개발주체가 될 수 없었느냐에 대해 원인을 알 수 있어서 그 전망에 대해 알아보자.

한글이 첨단반도체국가 이면서 반도체기기의 운영체제의 주체가 될 수 없었든 가장 큰 원인을 단적으로 표현하면 한글언어학자의 부재에 있다.

여기서 언어학자의 부재라는 것은 반도체적성을 이해하는 언어학자를 의미하며 그간

언어학자들이 반도체기기에 필요한 2진법의 체계를 전혀 의식하지 못하고 있다는 것을 의미한다.

1945년 8월 15일 해방과 더불어 한글 연구가 활발하게 진행되고 표준이라는 어원의 한글체제를 확립했다고 하드라도 2진법의 반도체제에 근간을 둔 한글에 대해서는 모르고 있는 것이나 다름이 없고 특히 반도체매체를 도입하는 수준에서는 백지나 다름이 없다는 것에 있다.

단순히 글자를 표의문자 표음문자로 분류하여 다루는 체제는 이제 아무 소용이 없는 골동품임을 알아야 하는 시대가 되었다.

앞으로는 말과 글을 다루는 언어학도 첨단반도체체제로 바뀌어야 한다는 것이다.

지금까지의 언어학자들이 2진법, 8진법, 16진법, 32진법, 64진법의 고급물리학의 영역을 생각해 본 일도 없고 설사 있다고 하드라도 이해하기가 매우 힘들다는 것이다.

오늘날에 언어학은 이러한 고급물리학의 2진법을 이해하지 않고는 접근할 수가 없다는 것에 있다.

필자가 프로그래머로서 현대 다분야 과학에 접근하면서 한글체제의 프로그램개발 언어가 없다는 것에 착안하여 한글을 2진법적인 논리로 접근하면서 한글의 자소가 반도체적성글자로서 가장 이상적이라는 것을 발견하게 되고 가장 이상적인 반도체적성글자를 가지고도 반도체운영체제를 개발하는 열강국가들의 프로그래머에게 가장 천대받는 글자가 되었는가에 대한 분석을 하고 그 해답을 찾게 되었다.

대한민국의 국민으로서 1446년 한글이 반포된 이래 해방과 더불어 최근 한글의 표준화를 제정하면서도 지금까지 2진법체제의 반도체기기 사용을 위한 한글을 제대로 연구하지 못하고 있었다는 것은 정말 부끄러움으로 알아야 한다.

2013년 현재에도 한국인에 의해서가 아닌 열강국가들의 프로그래머에 의해 자기들대로의 편리성으로 분류한 한글의 체계 속성을

'완성형글자'
'조합형글자'

라고 분류한 것을 여과 없이 그대로 표현하고 받아들이고 있음에서 찾을 수 있다.

즉, 알파벳권의 열강구가들의 프로그래머들에 의해

① 1993년 6월 유니코드 1.1.0 — 조합형 한글을 위한 첫가끝 코드를 포함시키게 되고

② 2009년 10월 1일 유니코드 5.2 — 확장 첫가끝 코드가 포함된 것에서

③ 한글 인코딩의 이해 1편: 한글 인코딩의 역사와 유니코드 개발자( 2012.02.29 16:41)

④ 월간 "마이크로소프트웨어" 2012년 5월호에 "유니코드와 JAVA를 이용한 한글 처리"라는 제목으로 실린 글에서 알 수 있다.

※③, ④의 글은 NHN Business Platform 쇼핑서비스개발팀 오영은님께서 인터넷 사이트에 게제된 것에서 인용하여 온 것으로 말미에 실어둔다.

필자는 글자에 완성형이니 조합형이니 하는 분류체계는 이치에 맞지 않으며 완성형글자와 조합형글자는 존재하지 않는다는 것이다.

열강의 프로그래머들이 한글 때문에 한글을 취급하다 궁여지책으로 내놓은 분류방법임을 알 수 있다.

완성형글자라고 한다면 완성형에 대한 반대개념인 미완성형글자에는 어떤 것이 있느냐고 물으면 대답이 불가능하기 때문이다.

또 조합형글자라고 한다면 조합형에 대한 반대개념은 분해된 글자이며 분해된 글자에는 어떤 글자들이 있느냐고 물어도 대답은 불가능하기 때문이다.

언어와 글자에 있어서 완성언어이니 미완성언어 그리고 완성형글자와 미완성글자는 없으며, 조합언어니 분해언어니 하는 것도, 조합글자니 분해글자니 할 수도 없다.

말과 글은 조합되고 완성되는 것이 아닌 있는 그대로 일 뿐이다.

단지 글에는 소(素)가 있으며 소리에도 소(素)가 있을 뿐이다.

글자의 소는 자소(字素)이며 소리의 소는 성소(聲素)이다.

글자의 자소가 낱낱으로 된 "낱개자소"이냐 아니면 몇 개를 부속품처럼 짜 맞추어진 "조립자소"이냐 하는 것일 뿐이다

소리에도 동물의 소리 자연의 소리 기계나 악기의 소리가 있지만 사람의 인체 발성기관에서 나오는 목소리인 성대의 소리에 대한 소(素)를 가지고 언어학적으로 따지어야 한다고 본다.

지구를 이루고 있는 물질을 쪼개고 나누면 분자라고 하는 화합물로 구성되었고 우리가 일상생활에서 다루는 물건이나 물질 또한 이들과 같이 분자화합물이며 분자를 쪼개어 더 이상 쪼갤 수 없는 물질이 될 때 이 물질을 소(素) 또는 원소라고 한다.

소리나 글자를 이렇게 분해하고 쪼개면 본질적인 소가 남게 된다. 이러한 논리로 한다면 소리는 소리소인 성소(聲素)가 되고 글자는 글자소인 자소(字素)가 된다.

프로그래머들이 아스키코드라는 것으로 알파벳 글자를 2진법에 해당하는 반도기기프로그램을 개발하고 운영체제를 구축하였다.

일반적으로 알파벳은 26개의 글자라고 알고 있다.

그러하다면 알파벳의 자소는 26개란 말인가?

그 대답은 아니다.

알파벳은 본디는 104개의 자소로 구성되어 있었다.

인쇄체와 필기체로 나뉘었고 이를 각각 대소문자로 나누어 사용하고 있어서 반도기기로서는 104개의 자소로 인식하게 된다.

아스키코드에서는 알파벳을 104개가 아닌 기본적인 52개의 자소로 코드화하여 사용하고 있다. 프로그램 전문용어로 ‘52개의 문자인코딩’ 이라고 한다.

알파벳 52개의 낱개 자소는 초기 컴퓨터프로그램을 개발하는 글자로는 안성맞춤의 글자이다.

그러나 알파벳자소는 개발의 단계를 지나 성숙단계에서는 반도체적성자소가 아닌 것이어서 후기 개발과 운영주체가 될 수가 없다.

그 이유는 반도체정보체계에서는 속도와 편의가 생명이기 때문에 상대적으로 느린 속도 탓에 밀려나가게 되기 때문이다.

한글의 글자를 컴퓨터 프로그램에서 사용할 수 있도록 인코딩하려면 몇 개의 자소가 될 것이며 어떠한 방법으로 인코딩하고 사용하여야 하는 것일까?

반도체기기가 등장하기 이전까지는 한글의 자소를 24개로 알아왔다.

자성과 모성으로 초성 중성 종성으로 글자가 만들어지는 것으로 알고 분류 하였다.

인간의 성대인 발성기관에서 성소로서 분해하여 자소를 설계하고 제작하였다고는 하였어도 조립글자라는 것으로 분류한 학자는 아무도 없었다는 것이다.

“조립글자이론은 이 책의 필자 최 무식이 최조로 주장한 것이다.”

반도체기기 등장이전까지 글자는 종이에 붓 펜이나 연필 등의 필기구로서 글자를 글씨로서 써 왔다는 것이다.

지금과 같이 글자를 입력하는 자판에서 선택하여 LCD 모니터에 실행한다는 개념으로 이해하지 않았기 때문에 한글을 ‘조립 자소’ 로 취급할 수 있는 기회가 한 번도 없었다는 것이다.

한글의 24개 자소는 글자를 조립하는 글자의 부품으로 생각하지 못했다는 것이다.

이러한 발상과 사고는 개발과 발전방향을 정하는 지표로서 엄청나게 중요한 작용을 하는 인자가 된다는 것을 알아야 한다.

중국의 글자는 획수를 기준으로 277개의 부수를 ‘좌우상하’ 로 조립하여 50,000개의 글자를 조립하여 필기하여 사용해 온 지구상에서 가장 많은 글자를 보유한 국가이기도 하다.

중국의 글자는 획수에 기초를 둔 상형문자로 분류하고 있다.

한국과 일본은 중국글자를 병행하여 사용한다.

현대중국인들은 50,000개 글자를 상용으로 7,000자를 사용하고 있으며 좀 더 간소화하기 위하여 간자체를 개발하여 사용하며 병음이라 하여 반도체기기인 컴퓨터에서는 전환하는 방법을 채택하여 사용한다.

컴퓨터상에서는 277개의 부품으로 50,0000개 글자를 자판에서 직접방식이 아닌 전환방식으로 밖에는 사용할 수 없어서 고유의 277개 조립자소방식은 컴퓨터상에서는 아무런 의미가 없는 것이 사실이다.

한글은 어떨까?

한글은 1446년 반포 이래 다듬고 다듬어서 12,420개 이상의 자소를 가진 것이나 실제 사용하는 데에는 현대화한글에서 11,172개의 조립글자에 51개의 자소를 더한 11,223개 글자 값을 가진 글자이다.

51개 자소에는 14개의 기본자성과 파생자성 16개로 총 자성이 30개이고, 10개의 기본모성에 11개 파생모성이 합해지고 9개 2중모성과 2개의 3중 모성으로 총 모성이 21개가 된다.

이러한 51개자소로서 초성 + 중성으로 399개의 글자를 조립한다.

여기에 종성자음 27개로 10,773개의 글자를 조립한다.

이러함에도 우리는 한글 스물 넉자라고 한다.

이글에서 생소한 개념의 단어들이 많이 등장하게 된다.

2진법으로 운용되는 반도체글자를 이해하기 위해서는 새로운 용어가 필연적으로 따르게 마련이라는 것으로 이해해 주기 바란다.

좀 더 구체적인 내용으로 들어가 보자.

글자는 인간이 개발한 것으로 종이나 다른 평면에 기록하기 위해 쓴 것이다.

필기 해왔다는 것이다.

그것도 반도체가 등장하기까지의 일이다.

해서, 글자는 필기한다는 개념으로만 연구하였고 개발하였다.

그러나 타가기가 등장하면서 두들긴다는 개념으로 이해해야 했고 반도체가 등장 이후부터는 글자는 필기하는 것이 아니고 선택하는 것이 되었다는 것을 먼저 이해해야 한다.

일반적으로 알고 있는 한글의 자소가 24개라는 개념으로 반도체기기에 접근하였다가 된서리를 맞게 된 원인이 바로 이것이다.

더욱이 알파벳문화권에서 먼저 반도체기기를 생산하여 판매하려고 하였으나 한글의 사용체제를 구축하는 것에는 많은 제약이 따름을 알게 되었지만 이 시기에 한국에서는 넋을 놓고 보고만 있는 상태의 신세가 되었다.

진정 한글을 사용하는 한국인들은 손을 쓰지 못하고 알파벳문화권의 처분만 기다리고 있었다는 것이다.

이것은 솔직한 현상이고 부끄러운 현상이다.

이쯤해서 글의 속성을 다시 고찰하고 정리해 보자.

글은 소리와 말, 색깔, 그려진 그림 또는 부호와 함께 정보를 전달하는 전달매체로 개발된 것이다.

글은 자소로 되어있고 분자물질과 같이 낱말 단어가 되고 단어가 모여 물질과 같은 문장이 되어 온전한 정보 역할을 하게 된다.

여기서 다루는 자소에서는 낱말직전까지만 다루게 된다.

자소는 어떠한 종류로 분류 되며 사용방법에는 어떤 방법이 있는가를 알아보자.

자소에는 ‘하나의 기계를 조립하는 부품과 같이’ 자소끼리 조립되어 하나의 글자로 되는 조립자소와 낱낱개개자소의 단순한 자소가 있고 이들 두 가지 자소가 모여 낱말을 만드는 것이다.

알파벳은 낱개 자소에 해당하고 한글이나 중국의 글자는 조립자소에 해당하게 된다.

조립자소에는 수평조립자소와 수직조립자소 그리고 수직수평이 모두 포함하는 수직수평조립 자소로 나눌 수가 있다.

이렇게 분류하면 “한글은 수직수평조립자소”에 해당한다.

이제 반도체가 등장하면서 글자는 필기가 아닌 선택이고 글자는 조림되는 부품으로 인식해야하는 시대가 되었다.

부품과 같은 낱개 자소가 조립되어 글자를 만들고 단어를 만든다.

인간의 발성기관인 성대 목소리는 초성 + 중성 + 종성으로 분해되는 성소로서 한글은 이러한 기본원리에 기초하여 자소를 설계제작 된 것이다.

한글자소는 14개 자성소와 10개 모성소로 하는 24개의 기본자성소에서 초성에 필요한 파생자소 5개를 합한 19개 초성자소 와 기본모성 10개에 파생모성 11개를 합한 21개의 중성자소 그리고 종성에는 초성의 19개에 파생자소에서 종성 자소로는 소리 값이 없는 파생자소 3개를 제하고 다시 더해지는 종성파생자소 11개를 합하여 27개의 종성 자소가 구성된 24개 기본자소와 초 중 종성의 원리로 구성한다.

이에 초성 + 중성으로만 조립되는 399개의 조립자소와 여기에 종성을 27개가 더하여 조립되는 10,773개를 합하면 11,172개기 되고 51개 자소를 더하면 총 11,223개의 글자 값을 가진 글자가 된다.

2진법으로 7비트를 사용할 경우 128개의 자소 인코딩이 기능하다.

2진법을 이해하려면 2에 곱해지는 자승을 이해하면 되는 것으로 2진법은 8진법 16진법 32집법 64진법으로 확대하여 사용하면 환산이 쉬운 특징을 가지고 있어서 이러한 방법을 활용한다.

초기 아스키코드를 사용하여 반도체 운영체제를 구축하여 사용하다 국제화하기 위해 글자를 인코딩을 위하여 설계하고 프로그램 소프트웨어를 설계 제작하는 과정에서도 유니코드라는 코드체계를 채택하게 되고 65,536개를 글자인코딩능력을 확보하게 되었다.

65,536개의 문자인코딩의 기술개발단계까지 한국의 반도체 프로그래머들과 언어학자는 넋을 놓고 바라만 보고 있을 수밖에 없는 우리의 현실체제를 이해해야만 한다.

쉽게 이해하려면 현대화 한글의 반도체적성의 언어적 개발연구에 프로그래머가 아닌 언어학자로서는 아무런 기여가치가 없다는 것이다.

시대가 이만큼 바뀌고 있은데 우리의 문화정책에서는 낮잠을 자고 있었다는 아픈 과거를 어떻게 반추해야 하는 것일까 한다.

반도체이용기기 개발단계에서는 바로자소이며 낱개자소로서 2 × 10□ = 128개 글자 값을 소화시키는 자소이어야 하지만, 사이버공간인 웹에서 활용하기 위해서는 바로자소이며 조립자소로서의 자격을 갖춘 자소이어야 하는 것이다.

이러한 이유는 글자입력자소판의 구성 원리에서도 찾을 수 있다.

글자입력자소판의 구성 원리를 자세하게 정리하면

1). 2□ = 128개의 입력에 필요한 기능과 자소 값 보유와 충족

2). 자소 = 성소 = 발성기호

3). 고정 폭 글꼴

4) 글꼴과 부호의 확실한 구별

5) 한글의 굴림체에 버금가는 가독성

현재 지구상의 글자 중 위의 5가지를 만족시키는 자소는 한글뿐이다.

앞서 언급한 봐와 같이 조립자소는 프로그램개발에 필요한 자소로는 부적절함을 알 수 있고 사이버공간웹에서 사용에는 조립자소이어야 한다면 앞으로 한글자소만으로서 한글프로그램언어를 개발할 수 있을까?

이에 대한 대답은 2□ = 128개 코드 값으로는 불가능하다.

그 이유로는 조립자소수가 $2^{1}$ □ = 65,526개의 코드 값이 확보된 환경이어야 프로그램언어에서 부호화해야하는 기계언어들까지 소화 한다.

이러한 부호화기계어를 생성하는 데에는 알파벳과 같은 낱개자소가 유리하기 때문이다.

따라서 한글운용체제와 프로그램언어개발에는 알파벳 혼용부호와

조립자소 + 낱개자소 = 미래 반도체 컴퓨터운용 프로그램개발

이라는 등식이 성립함을 알 수 있다.

요즘 젊은 세대는 단순축약 언어를 생성하여 일상에 많이 쓰고 있다.

빠른 정보생성과 이해를 위해서는 많은 영역의 어휘가 필요하듯 프로그래밍언어도 단순축약 기계언어로 생성하여 구축되어야 하기 때문에 조립자소로서 부족한 부분은 낱개 자소로 보강할 필요성이 있다.

이제는 한글이 반도체프로그래밍언어로서 국제화되어야 하고 한글자소에 기반을 두고 국제적 호환성을 갖춘 운영체제를 개발하여 운영할 때라고 보아야 한다.

반도체최적자소를 보유한 한글을 사용하는 문화권에서 한글자소 컴퓨터운용체제와 프로그램언어가 개발되어 있지 않다는 것은 2013년 현시점에서는 심각한 문제로 받아들여야 한다.

한글운용체제와 프로그램언어개발의 시기가 도래했다는 것이기 때문인 것으로 다시 말해 인터넷사이버시대 웹상의 주도문화로서의 한글자소입지가 확보된 시대라는 것이다.

반도체기기는 과학의 발전과 경제, 문화 등 모든 분야의 근간이기 때문에 한글자소의 웹 운용체제와 프로그래밍기계언어가 개발되지 아니하고는 한국인의 우수한 두뇌를 활용할 기회를 제한하는 것이기 때문에 프로그램개발언어는 물론 운용체제를 한글자소체제에 바탕을 두고 새로운 반도체시대의 틀을 구축하여야 하는 것은 시대적인 사명으로 보아야 한다.

8 한글이 한국인에게 미친 영향

알파벳과 같은 낱개 자소는 사용방법에 있어서 수평으로 나열하여 사용하기는 편리하나 수직으로 사용하기에는 매우 부적절하다는 것을 알 수 있고, 조립자소는 낱자소의 단점을 모두 해결해 주는 장점이 있지만 조립되는 글자개수가 많아지는 관계로 글자인코딩이 되지 않은 한 운영체제를 구축하는 초기 자소로는 불리하다는 것을 알게 된다.

이러한 문제들은 뒤에 다시 정리하여 기술하기로 하고 최근 대한민국이 경제, 문화, 과학 분야에 두드러진 발전이 계속되어 선진대열을 넘어 최 일류 첨단국가로 향해가는 것을 피부로 느끼게 된다.

국가 근대화 사업은 영국이 400년 일본의 120년이 걸린 기간을 한국은 30년으로 최단기간에 근대국가로 기반을 구축한 것은 우리의 혼 빨리 빨리라는 것이 있었기 때문이

다,
브라운관이 등장하면서 한국은 보아의 춤과 노래가 지구촌에 퍼지고 비보이의 현란한 몸동작이 세계를 매료 시킬 수 있었고 강남스타일이 사이의 말 춤으로 한류의 열풍을 불게 한 것은 반도체 시대의 적성을 만족시키는 빨리 빨리라는 한국의 혼과 수직수평을 모두 만족시키는 반도체문자를 사용하면서 길들여진 수직수평사고의 덕택이라고 보아야 한다.
음식에서도 한류의 열풍이 일어나고, 노래와 춤의 오락문화에도 한류의 열풍이 일어나고, 영화계에서도 한류의 열풍이 일어나고, 축구월드컵경기에서 4강이 올림픽으로 이어져 동 매달을 차지했고 올림픽의 종합8위 성적을 냈고, 경제면에서도 반도체, 선박, 건설 등에서 추종을 불허할 정도로 발전하고 있다.

이러한 발전의 원동력 진원지를 어디에서 찾아야 할 것인가를 생각해 볼 때가 되었다.

2012년 영국에서 개최한 하계올림픽에서 양궁부문에서 연속 금 매달을 차지하자 한국이 쇠 젓가락을 사용하여 감각을 익혔기 때문이라고 세계가 극찬하였고, 사격에서도, 구기 종목에서도, 체조 등에서 놀라운 발전을 보이고 있다. 2012년 초중등학교 학생들의 창의력 올림피아드에서도 금상을 수상하였고, 매회 기능올림픽에서도 성적이 우수했던 것은 단순한 교육열만으로 설명할 수는 없다.

과거역사에서 찾아볼 수 없을 만큼 근자에 세계무대에서 한국인의 두각을 나타내고 있는 진원지를 어디에서 찾을 것인가?

여기에 필자는 이렇게 대답하고 싶다.

한국인이 사용하는 자소가 조립자소이기 때문이라고 당당하게 말하고 싶다.

빨리 빨리라는 정보특성의 글자인 한글을 사용하면서 빨리 빨리라는 한국인의 정서가 다듬어진 것이 한국발전의 원동력이라고도 보아야 한다.

한때는 '빨리빨리 서두는 것을 한국의 병' 이라 하고 양은냄비니 하여 비하하는 시대도 있었다.

빨리 빨리라는 빨라진 정보 확보만큼 기술과 능력에 있어서 시대를 앞서게 되는 것이다.

첨단반도체국가라는 국위에 오른 지금 아무도 그 누구도 '한국병' 이라고 말할 사람은 없다.

한국인이 한글이라는 자소를 사용하고 있어서 세계에서 가장 빠른 정보체계를 보유하고 있기 때문에 지피지기는 백전백승이라는 성어가 바로 맞아 떨어진 결과라고 할 수 있다.

앞으로 다가오는 세대에는 한국인이 한글을 사용하는 장점이 있는 한 더욱더 두드러지는 발전현상을 보게 될 것이다.

그리고 또 하나 한글을 사용하면서 얻어지는 효과는 낱 자소 알파벳 나열방식에서 얻어지지 못하는 공간사고방식이 길들여진다는 것을 알 수 있고 그 결과로 한국인의 두뇌가 우수하다는 것도 알 수 있다.

수직수평조립자소의 글자를 사용하는 문화권만이 서예라는 글자쓰기의 공간 구성의 예술분야가 있다.

수직수평조립자소의 글자를 사용하는 문화권만이 처음 학교에 들어가 글자를 익힐 때 반드시 수직수평으로 글자체의 수직수평 공간구성의 방법을 익히게 한다.

나열글자에도 서체는 있으나 극히 단조로운 필기체 인쇄체 수준이다.

한글에는 필기체와 인쇄체의 구분이 없는 글자이지만 수직수평으로 조립되어지는 글자이기 때문에 쓴 글자의 모양으로 그 사람의 인품으로 평가했으며 배운 사람의 덕목으로 여긴 것이 뇌 발달 근본이 되었다.

나열방식은 낱자소를 수평으로 나열만 하여 단어(낱말)를 입력되고 자소가 초중종의 소리소로 개발된 것이 아니라서 단어의 구성이 일정한 규칙 없이 사용되고 있음을 들 수 있다.

또 영문자를 자판으로 두들기다보면 일정한 규칙이나 리듬이 없다는 것을 느끼게 된다.

알파벳단어는 양손이 아닌 한 손의 손가락만으로 두들겨야 하는 경우가 많다.

양손을 번갈아 가면서 리듬에 맞추면 손의 피로도 줄어들고 일의 능률도 높아질 터인데 그러하지 못하다는 것을 알게 된다.

알파벳으로 사용되는 단어를 분석해 보면 묵음도 있고 한 자소에서 사용에 따라 여러 가지 소리를 내며 모음이 없는 자음에서 모음을 만들어 소리 내기도 하고 어렵고 복잡한 어미변화와 접두사 접미사를 붙여 사용하는 단어사용습관은 정보입력속도를 매우 느리게 하고, 자소를 입력하는 자판에 소리소의 구성인 초성중성종성별로 자음 모음으로 구별하여 자소배열을 할 수 없어 좌우 두 손의 손가락을 번갈아가면서 동시에 사용할 수 있는 자소배열이 불가능한 결점을 가지고 있다.

뒤에 설명하는 기회가 있겠지만 같은 내용의 정보를 담은 책을 만든다면 한글의 책 분량에 비해 알파벳은 1.3 ~ 1,5배의 분량이 된다는 것이 조사를 통해 나타났다.

다시 말해 같은 내용의 한글정보와 알파벳정보를 비교하면 전달소요시간과 정보를 담은 매체의 공간면적이 한글의 0.3 ~ 0.5배가 더 있어야 알파벳정보가 전달된다는 것이다.

알파벳의 언어습관에도 문제가 있는 것으로 그들은 문장의 첫머리 자소는 대문자라야 하고 단수 복수를 구분하는 단어를 만들어야 하고 'a' 와 특정 지칭어에는 'the' 를 사용해야 한다.

우리들의 언어습관으로는 매우불편하고 필요 없는 '것' 에 해당한다.

하여 정보입력과 전달해독속도에 많은 지장을 주고 있다.

낱자소의 결점은 또 하나 있다.

아래는 본 글에서 아스키를 설명하기 위한 부분을 이미지로 만든 것으로

타자기 이후 알파벳문화권에서 아스키코드(ASCII, American Standard Code for Information Interchange, 와

타자기 이후 등장한 것이 알파벳문화권에서 사용한 아스키코드로서 (ASCII, American Standard Code for Information Interchange,□

중에서 양 ASCII, American의 글자사이를 비교해 보자

단어사이의 간격이 많이 벌어졌다 좁아졌다 하는 것을 확인하게 된다.

낱자소인 알파벳은 "American" 이 글줄 말미에서 동강이 날 경우 조립자소는 동강이 나도 한 단어로 읽을 수 있지만 낱 자소는 둘로 나누어진 단어로 인식하게 되어 버린다.

“한글2007기능메뉴에는 앞뒤글줄정리 기능메뉴가 있으나 엑셀2007글줄메뉴에는 이것이 없다

이러한 이유 등으로 정보속도에 한글을 따라올 수 없고 한글속도의 0.5정도가 항상 뒤지는 속성이 있다는 것을 알 수 있다.

이제 이러한 엄청난 사실을 확인한 이상 한국인은 한글에 맞는 컴퓨터운영체제를 개발하고 사용해야 한다.

이걸 알게 되면 한국인의 학생이 영어를 배우려고 기를 쓴다는 것이 얼마나 어리석고 바보인가를 알게 될 것이다.

곧 “한국어가, 한글이 세계어의 지위를 차지할 수밖에 없다”라는 시대가 온다는 결론이 나오는 것이다.

또 한글은 수직 수평으로 사용할 수 있어서 정보효율을 높이기 위한 표현공간을 설계해야 하는 공간 활용 능력을 기르게 되는 것이 생활화 되고 있다.

그러나 알파벳은 세로로 단어나 문장을 표현할 수 없다.

이것이 낱자소의 나열표현방식과 자소조립으로 된 조립글자의 표현방법의 크나큰 차이이다.

그만큼 한국인은 알파벳문화권보다 뇌를 더 사용하면서 살아가고 있으니 뇌 발달이 되어 있는 우수한 민족이 되는 것이 된다.

언어습관과 의식구조에 있어서도 조립되는 자소들처럼 표현 한다.

한국인은 자기를 지칭할 때 ‘나라고 하기보다는 우리라고’표현한다.

형제자매가 없는 독신인 경우도 반드시 ‘우리 어머니, 우리 아버지’라고 하며 ‘내 어머니, 내 아버지’라고 하지 않는다.

혼자 살면서도 자기의 집을 ‘우리 집’이라고 한다.

한 울타리에서 대가족으로 구성하여 살아왔기 때문에 구성되고 맺어지는 환경을 항상 우선으로 생각한다.

자소들이 조립되어 조립글자를 구성하듯이 한국인의 생활은 조직을 중심으로 이루어지고 있다.

한국인들이 올림픽게임에서 단체로 구성하는 축구 배구 야구 핸드볼 양궁 탁구에서 좋은 성적을 내는 것은 어쩌면 당연한 것이다.

그 이유와 결과로 앞으로의 지구촌에서는 한국인이 지구촌의 경제는 물론 문화도 주도하면서 살아가야 하고 또 그렇게 되게 되어 있다.

지구촌에는 2013년 현재 70억 인구가 살아가고 있으며 대한민국의 인구가 차지하는 비율은 불과 0.7%에 이르면서 경제7대강국이고 과학7대강국이다.

올림픽게임에서는 8강에 위치하고 기능올림픽에서는 1위 초중등학교 학생들의 창작올림픽에서도 단연 대한민국의 어린이들이 1위를 차지하고 있다.

2000년대에 들어오면서 세계는 한류에 술렁인다.

세계인은 한류의 파고를 받아들이고 즐기고 움직인다.

이러한 원인은 우연에서 온 것이 아니다.

한국인이 한글을 사용하고 있기 때문이다.

언제부터인가 한국인에게는 '빨리빨리' 라는 정신과 혼이 자랐었다.

'빨리빨리' 일을 처리하고 발전시키려면 정보가 빠르게 유통되어야 하고 머릿속에서는 생각이 빠르게 돌아가야 하는 것이다.

과거 한 때 같은 유교권의 한중일의 동양 3국의 문화를 이렇게 표현 한 때도 있었다.

중국을 '돌다리도 두들기는' 문화라 했고 일본을 '버드나무' 문화라 했으며 한국을 '달리면서 생각하는 주마간산' 문화라 하여 '빨리빨리' 에 빗대는 시대가 있었다.

대전 이후 민주와 공산이라는 양 의념체계가 무너지고 소련이 붕괴어고 세계가 국가라는 울타리장벽이 걷히고 외국의 문물과 교류가 터이던 때까지 유교문화권이었던 한중일 3국이 외국문물을 받아들이는 과정에서 중국은 신중하고 보수적이며 외국의 문물은 그대로 바로 받아들이지 않고 철저하게 여과하는 문화권이라고 했다.

새로운 용어를 받아들이는 데에도 중국은 자기여과를 강하게 하는 편이라 당시 코카콜라(Coca-Cola)가 세계적인 음료가 되어 중국에도 들어가게 되자 중국인들은 코카콜라를 '가구가락(可口可樂)이라 하여 마실수록 즐겁다' 라는 브랜드의 '커크우커리' 로 받아들였다.

외국기업이 중문 브랜딩을 위한 7대 지침이 생겨날 정도이다.

1), 중국의 문화를 우선 이해하라

2), 뜻, 소리, 이미지를 함께 고려한 고도의 기법을 활용하라

3), 품목에 어울리는 한자와 이미지를 활용하라

4), 국내에서 쓰던 한자명을 중국시장에서 고집하지 말라

5), 중국에서 금기시하는 단어와 이미지사용에 유의하라

6), 중국시장에서 이미 강하게 인식된 브랜드는 개명하지 말라

7), 중국 현지인의 확인 작업을 반드시 거쳐라

중국이 사용하는 대표적 브랜드명(예)

인구밀집지역에서 토지이용효율을 높이기 위해 단층에서 복층으로 주거공간의 이용방법이 달라진 시대 아파트먼트(apartment house)가 들어서게 된다.

한국인은 이러한 아파트먼트층집을 줄여 아파트라고 하지만 중국인들은 층집이라는 루(樓)라는 용어를 사용한다.

일본은 외국의 것을 여과 없이 그대로 받아들인다.

따라서 일본이 서양의 근대문물을 앞서 받아들였다고 할 수가 있다.

버드나무는 잘라지거나 홍수로 파헤쳐져서 떠밀려 가드라도 습기와 토양이 있으면 가리지 않고 부착하여 새 뿌리를 내리고 정착한다.

주체의식이 부족하고 상황과 여건에 따라 유리하다고 생각된 쪽으로 쉽게 정착하여 살아가는 사람을 '버드나무 팔자' 라고도 한다.

한국인을 기마민족이라 한다.

말을 타고가면서 산을 보듯 빠르게 생각하고 판단하고 결정한다.

일처리를 주마간산(走馬看山)’ 처럼 한다하여 달리면서 생각하는 민족이라고 한다.
그러하기 때문에 중국인들처럼 일일이 여과하고 따질 여가가 없다.
그렇다고 일본인들처럼 전혀 여과 없이 액면 그대로 받아들이지는 아니한다.
반드시 우리 것을 가져다 붙여서 사용한다.
깡통(an empty tin-can)이라는 단어는 캔과 통이라는 어원을 조합하여 만들어 졌다.
can = 통은 같은 그릇과 같은 용기이다.

깡은 캔(can)에서 왔고 통은 우리말에서 온 것으로 깡통이란 ‘통 통’ 이라고 2중으로 표현하여 사용한다.
일본인들이 찹쌀가루로 속에 팥으로 소를 넣어 만든 떡을 ‘모찌(もち)□ 비슷한 말 모찌떡, 찹쌀떡)’ 라고 한다.
모찌 = 떡 이 같은 떡으로 모찌떡이란 모찌의 일본 고유의 음식 떡과 떡이라는 우리말을 합쳐서 모찌 + 떡을 2중으로 표현된 말이 된다.
서양식의 문을 ‘도어(door)문'이라고 한다.
도어(door)는 서양말이고 문은 우리말로 2중으로 된 단어를 만들더라도 꼭 우리말을 붙여서 사용한다.

반도체는 128개 아스키코드에 의해서 출발하여 개발되었고 65,536개 유니코드로 세계화를 완성시켜 사용하는 것이 발전경로상의 수순이라면 반도체문화의 꽃은 한글의 문화요 대한민국의 문화가 세계화문화 열매가 된다는 것에 의심의 소지는 없다.
이는 프로그래머들이라면 누구나 인정하는 사실이다.
대한민국 컴퓨터 보급률과 보급 속도가 세계최고이고, 인터넷보급과 활용 또한 최고가 되는 이유는 한국인이 조립되는 자소성질의 한글을 사용하고 있기 때문이다.
전화 기능만의 휴대 전화기에서 화상전화는 물론 무선인터넷기기로 개발하는 저력과 세계최고보급률의 고지를 점령한 것은 한국이고 한글에 의한 저력이라고 봐야 한다.
이처럼 반도체와 한글의 위력은 크다고 보아야 한다.

유행 아닌 첨단반도체 추구열풍
황의 법칙에 의하여 한국시장에는 새로운 첨단반도체기기가 계속 출시된다.
특히 pc는 물론 non-pc부문에서 두드러진 현상을 보인다.
아무리 좋은 제품이 시장에 출시된다고 해도 소비가 없고 고객이 없으면 실패작이 되고 개발자는 실패라는 부담으로 더 이상의 개발은 되지 않게 된다.
한국시장의 특징은 새로운 것을 항상 추구하고 있다는 것에 있다.
멀쩡한 것이라도 새롭고 편리한 것이 나오면 새것으로 바꾸어 주게 된다.
새롭게 바뀐 생산품의 목표가 국내위주로 생산되는 것은 아니다.
한국에서 신제품은 국외에서도 신제품이 되어 수출의 길이 열리고 외화벌이가 된다.
한국인에게는 자원이 부족한 것이 최대의 걸림돌이라는 것을 안다.

한국인은 수출을 위한 신제품 신상품을 소비시켜 줌으로 알게 모르게 개발자를 위한 노력과 협조를 하는 셈이 된다.

한국인에게는 한국의 발전을 위한 이러한 혼이 있다.

근검하고 절약하는 것만이 최상이 아니라는 것이다.

개발과 발전은 머무르는 순간부터 퇴보하는 것이고 생활의 향상과 삶의 질 또한 머무는 순간부터 다른 나라보다 떨어지는 것이다.

한국인은 새롭고 발전된 것을 추구하고 새로 개발된 신제품을 구입하여 사용하는 신제품의 사랑이 결구에는 해외 수출산업으로 이어진다는 것을 알고 있기 때문에 이렇게 하여 개발된 제품이 해외 수출시장에서 외화를 벌어들이고 국내의 자원 등 모든 열악한 분야를 이기는 길이라는 것을 알고 있다.

처음 한글이 작개언로 달사총으로 시작하여 반도체기기를 사용하는 현재시대 가장 아름다운 '정보화' 를 이룩하였고 계속될 것이다.

결과적으로 한글은 한국인에게 많은 영향을 미친 것으로 한국혼의 주체가 된 것이고 지구상에서 한국인이 살아갈 수 있게 한 것이다.

한국은 석유 없이 살아갈 수 없는 시대 석유 한 방울 나지 않고 자원이라고는 없는 나라이면서 첨단반도체시대가 되면서 국가경쟁력이 살아나고 산업과 문화 과학에 있어서 세계 속의 중심국가가 될 수 있는 것은 한국의 혼이 스스로가 정보화사업을 주도하고 있기 때문이다.

한국인이 한글을 사랑하는 세계 속의 한 일류 국가로서 살아남을 것이다.

## 9, 반도체와 알파벳자소 판

반도체기기에 필요한 기계언어에 대해서 알아보자.

반도체는 저장능력은 있어도 사고의 능력이 없는 기계이다.

컴퓨터는 인간이 개발하고 기능들을 내장시키고 입력시켜 인간이 필요한 일을 시킨다.

그래서 컴퓨터언어를 명령어라고 하고 기계어라고도 한다.

컴퓨터운용체제를 위한 프로그램언어를 저급언어라고 하고 인간언어를 고급언어라고 한다.

컴퓨터언어는 '0, 1' 로 된 2진법의 문자 값에 기초를 둔 언어이다.

많은 프로그래머들이 컴퓨터에 인간의 언어를 내장시키려고 연구하고 노력하여 만들어낸 언어가 고급과 저급의 중간인 중간언어를 준비해 왔다.

프로그래머들은 중간언어에 해당하는 어셈블리언어(assembly 言語, symbolize)라고 하는 프로그램언어로 점차적으로 고급언어로 개발되고 있다.

이는 초보자를 위한 목적도 있지만 반도체기기 상호간 호환성을 목적으로 개발하기도 한다.

어셈블리언어 ; 컴퓨터 언어로서 간단한 단어와 쉬운 기호로 이루어진 컴퓨터 프로그래밍 언어의 한 가지. 기계어와 일 대 일로 대응되는 명령어 체계를 가졌으며, 컴퓨터 기종에 따라 서로 통용이 될 수 없어 기계어에 가까운 저급 언어임. 기호 언어.

"반도체" 라는 어원은 전기의 성질을 설명하는 데서 찾게 된다.

전기를 통하게 하는 성질을 가진 물체를 전도체 혹은 도체라고 하고

그러하지 않은 물질을 부도체라고 한다,

오늘날 사용하고 있는 전기는 도체를 통하여 도선을 따라 발전소에서 우리 주면으로 흘러 온 것이다.

전기가 흐리지 못하게 하는 부도체도 전기를 사용하기 위해 도체와 같이 사용하게 되는데 이를 절연체라 하여 전선에 피복하여 감전으로 나타나는 사고를 피하게 되고 누전으로 오는 전기소모를 줄이고 효율적으로 이용하게 된다.

이러한 전기사용에서 도체와 부도체의 중간 성질을 가진 부도체가 발견되고 그것을 사용하는 방법을 발명하게 된 것이 반도체인 것으로 1900년대 후반기부터 반도체를 이용하는 산업이 급격한 발달을 하여온 것으로 이를 반도체산업이라 하고 모든 산업의 중추적 역할을 하는 것이라 하여 첨단반도체산업이라고까지 하게 되어 중요한 산업이 되었다.

2013년 현 시점에서 세계수준에서 볼 때 대한민국은 단연 첨단반도체산업의 선두주자임은 틀림없는 사실이나 초기반도체기기 개발시기에서 운영체제들은 알파벳문화권의 전유물이었다는 것은 숙명으로 받아들여야 하게 되고 있지만 성숙단계에서는 한국중심의 운용체제개발이 이루어져야 한다는 것이다.

초기 알파벳글자 입력기를 설계할 때에는 '7개 비트'로 구성된 128개의 아스키코드를 만들어 내었다.

여기서 잠깐 아스키코드의 발생에 대해 언급해 보자

아스키(ASCII, American Standard Code for Information Interchange, 미국 정보 교환 표준 부호)는 영문 알파벳을 사용하는 대표적인 문자 인코딩이다. 아스키는 컴퓨터와 통신 장비를 비롯한 문자를 사용하는 많은 장치에서 사용되며, 대부분의 문자 인코딩이 아스키에 기초를 두고 있다.

아스키는 1967년에 표준으로 제정되어 1986년에 마지막으로 개정되었다. 아스키는 7비트 인코딩으로, 33개의 출력 불가능한 제어 문자들과 공백을 비롯한 95개의 출력 가능한 문자들로 이루어진다. 제어 문자들은 역사적인 이유로 남아 있으며 대부분은 더 이상 사용되지 않는다. 출력 가능한 문자들은 52개의 영문 알파벳 대소문자와, 10개의 숫자, 32개의 특수 문자, 그리고 하나의 공백 문자로 이루어진다.

아스키가 널리 사용되면서 다양한 아스키 기반의 확장 인코딩들이 등장했으며, 이들을 묶어서 아스키라고 부르기도 한다. 대표적으로 7비트 인코딩을 유지한 ISO/IEC 646과, 원래 아스키코드 앞에 비트 0을 넣어 8비트 인코딩을 만든 IBM 코드 페이지와 ISO 8859가 있다. 이 인코딩들은 언어 군에 따라 같은 숫자에 서로 다른 문자가 배당된 경우가 많다.

제어할 때 쓰는 코드이다. 그러나 지금은 거의 사장된 코드로 역사적인 의미 때문에 남아 있다.

알파벳의 대소문자 52개가 아스키에서 7비트로 인코딩하는 과정에서

- 33개의 출력 불가능한 제어 문자들
- 영문대소 52개
- 숫자 10개

- 부호와 같은 특수문자 32개
- 공백문자 1개

를 모두 합하여 128개가 필요하다.

아스키(ASCII, American Standard Code for Information Interchange, 미국정보교환표준부호)

| 번호 | 이진법 | 팔진법 | 십진법 | 십육진법 | 약자 | 설명 | 한글설명 |
|---|---|---|---|---|---|---|---|
| 1 | 000 0000 | 000 | 0 | 00 | NUL | Null | 공백 문자 |
| 2 | 000 0001 | 001 | 1 | 01 | SOH | Start of Header | 헤더 시작 |
| 3 | 000 0010 | 002 | 2 | 02 | STX | Start of Text | 본문 시작, 헤더 종료 |
| 4 | 000 0011 | 003 | 3 | 03 | ETX | End of Text | 본문 종료 |
| 5 | 000 0100 | 004 | 4 | 04 | EOT | End of Transmission | 전송 종료, 데이터 링크 초기화 |
| 6 | 000 0101 | 005 | 5 | 05 | ENQ | Enquiry | 응답 요구 |
| 7 | 000 0110 | 006 | 6 | 06 | ACK | Acknowledgment | 긍정응답 |
| 8 | 000 0111 | 007 | 7 | 07 | BEL | Bell | 경고음 |
| 9 | 000 1000 | 010 | 8 | 08 | BS | Backspace | 백스페이스 |
| 10 | 000 1001 | 011 | 9 | 09 | HT | Horizontal Tab | 수평 탭 |
| 11 | 000 1010 | 012 | 10 | 0A | LF | Line feed | 개행 |
| 12 | 000 1011 | 013 | 11 | 0B | VT | Vertical Tab | 수직 탭 |
| 13 | 000 1100 | 014 | 12 | 0C | FF | Form feed | 다음 페이지 |
| 14 | 000 1101 | 015 | 13 | 0D | CR | Carriage return | 복귀 |
| 15 | 000 1110 | 016 | 14 | 0E | SO | Shift Out | 확장문자 시작 |
| 16 | 000 1111 | 017 | 15 | 0F | SI | Shift In | 확장문자 종료 |
| 17 | 001 0000 | 020 | 16 | 10 | DLE | Data Link Escape | 전송 제어 확장 |
| 18 | 001 0001 | 021 | 17 | 11 | DC1 | Device Control 1 | 장치 제어 1 |
| 19 | 001 0010 | 022 | 18 | 12 | DC2 | Device Control 2 | 장치 제어 2 |
| 20 | 001 0011 | 023 | 19 | 13 | DC3 | Device Control 3 | 장치 제어 3 |
| 21 | 001 0100 | 024 | 20 | 14 | DC4 | Device Control 4 | 장치 제어 4 |
| 22 | 001 0101 | 025 | 21 | 15 | NAK | Negative Acknowledgement | 부정응답 |
| 23 | 001 0110 | 026 | 22 | 16 | SYN | Synchronous idle | 동기 |
| 24 | 001 0111 | 027 | 23 | 17 | ETB | End of Transmission Block | 전송블록 종료 |
| 25 | 001 1000 | 030 | 24 | 18 | CAN | Cancel | 무시 |
| 26 | 001 1001 | 031 | 25 | 19 | EM | End of Medium | 매체 종료 |
| 27 | 001 1010 | 032 | 26 | 1A | SUB | Substitute | 치환 |
| 28 | 001 1011 | 033 | 27 | 1B | ESC | Escape | 제어기능 추가 |
| 29 | 001 1100 | 034 | 28 | 1C | FS | File Separator | 파일경계 할당 |
| 30 | 001 1101 | 035 | 29 | 1D | GS | Group Separator | 레코드 그룹경계 할당 |
| 31 | 001 1110 | 036 | 30 | 1E | RS | Record Separator | 레코드 경계 할당 |
| 32 | 001 1111 | 037 | 31 | 1F | US | Unit Separator | 장치 경계 할당 |
| 33 | 111 1111 | 177 | 127 | 7F | DEL | Delete | 삭제 |
| 34 | 010 0000 | 040 | 32 | 20 | □ | | |
| 35 | 010 0001 | 041 | 33 | 21 | ! | | |
| 36 | 010 0010 | 042 | 34 | 22 | " | | |
| 37 | 010 0011 | 043 | 35 | 23 | # | | |
| 38 | 010 0100 | 044 | 36 | 24 | $ | | |
| 39 | 010 0101 | 045 | 37 | 25 | % | | |
| 40 | 010 0110 | 046 | 38 | 26 | & | | |
| 41 | 010 0111 | 047 | 39 | 27 | ' | | |
| 42 | 010 1000 | 050 | 40 | 28 | ( | | |
| 43 | 010 1001 | 051 | 41 | 29 | ) | | |
| 44 | 010 1010 | 052 | 42 | 2A | * | | |

| | | | | | |
|---|---|---|---|---|---|
| 45 | 010 1011 | 053 | 43 | 2B | + |
| 46 | 010 1100 | 054 | 44 | 2C | , |
| 47 | 010 1101 | 055 | 45 | 2D | − |
| 48 | 010 1110 | 056 | 46 | 2E | . |
| 49 | 010 1111 | 057 | 47 | 2F | / |
| 50 | 011 0000 | 060 | 48 | 30 | 0 |
| 51 | 011 0001 | 061 | 49 | 31 | 1 |
| 52 | 011 0010 | 062 | 50 | 32 | 2 |
| 53 | 011 0011 | 063 | 51 | 33 | 3 |
| 54 | 011 0100 | 064 | 52 | 34 | 4 |
| 55 | 011 0101 | 065 | 53 | 35 | 5 |
| 56 | 011 0110 | 066 | 54 | 36 | 6 |
| 57 | 011 0111 | 067 | 55 | 37 | 7 |
| 58 | 011 1000 | 070 | 56 | 38 | 8 |
| 59 | 011 1001 | 071 | 57 | 39 | 9 |
| 60 | 011 1010 | 072 | 58 | 3A | : |
| 61 | 011 1011 | 073 | 59 | 3B | ; |
| 62 | 011 1100 | 074 | 60 | 3C | 〈 |
| 63 | 011 1101 | 075 | 61 | 3D | = |
| 64 | 011 1110 | 076 | 62 | 3E | 〉 |
| 65 | 011 1111 | 077 | 63 | 3F | ? |
| 66 | 100 0000 | 100 | 64 | 40 | @ |
| 67 | 100 0001 | 101 | 65 | 41 | A |
| 68 | 100 0010 | 102 | 66 | 42 | B |
| 69 | 100 0011 | 103 | 67 | 43 | C |
| 70 | 100 0100 | 104 | 68 | 44 | D |
| 71 | 100 0101 | 105 | 69 | 45 | E |
| 72 | 100 0110 | 106 | 70 | 46 | F |
| 73 | 100 0111 | 107 | 71 | 47 | G |
| 74 | 100 1000 | 110 | 72 | 48 | H |
| 75 | 100 1001 | 111 | 73 | 49 | I |
| 76 | 100 1010 | 112 | 74 | 4A | J |
| 77 | 100 1011 | 113 | 75 | 4B | K |
| 78 | 100 1100 | 114 | 76 | 4C | L |
| 79 | 100 1101 | 115 | 77 | 4D | M |
| 80 | 100 1110 | 116 | 78 | 4E | N |
| 81 | 100 1111 | 117 | 79 | 4F | O |
| 82 | 101 0000 | 120 | 80 | 50 | P |
| 83 | 101 0001 | 121 | 81 | 51 | Q |
| 84 | 101 0010 | 122 | 82 | 52 | R |
| 85 | 101 0011 | 123 | 83 | 53 | S |
| 86 | 101 0100 | 124 | 84 | 54 | T |
| 87 | 101 0101 | 125 | 85 | 55 | U |
| 88 | 101 0110 | 126 | 86 | 56 | V |
| 89 | 101 0111 | 127 | 87 | 57 | W |
| 90 | 101 1000 | 130 | 88 | 58 | X |
| 91 | 101 1001 | 131 | 89 | 59 | Y |
| 92 | 101 1010 | 132 | 90 | 5A | Z |
| 93 | 101 1011 | 133 | 91 | 5B | [ |
| 94 | 101 1100 | 134 | 92 | 5C | ₩ |
| 95 | 101 1101 | 135 | 93 | 5D | ] |
| 96 | 101 1110 | 136 | 94 | 5E | ^ |
| 97 | 101 1111 | 137 | 95 | 5F | _ |

| | | | | | |
|---|---|---|---|---|---|
| 98 | 110 0000 | 140 | 96 | 60 | ` |
| 99 | 110 0001 | 141 | 97 | 61 | a |
| 100 | 110 0010 | 142 | 98 | 62 | b |
| 101 | 110 0011 | 143 | 99 | 63 | c |
| 102 | 110 0100 | 144 | 100 | 64 | d |
| 103 | 110 0101 | 145 | 101 | 65 | e |
| 104 | 110 0110 | 146 | 102 | 66 | f |
| 105 | 110 0111 | 147 | 103 | 67 | g |
| 106 | 110 1000 | 150 | 104 | 68 | h |
| 107 | 110 1001 | 151 | 105 | 69 | i |
| 108 | 110 1010 | 152 | 106 | 6A | j |
| 109 | 110 1011 | 153 | 107 | 6B | k |
| 110 | 110 1100 | 154 | 108 | 6C | l |
| 111 | 110 1101 | 155 | 109 | 6D | m |
| 112 | 110 1110 | 156 | 110 | 6E | n |
| 113 | 110 1111 | 157 | 111 | 6F | o |
| 114 | 111 0000 | 160 | 112 | 70 | p |
| 115 | 111 0001 | 161 | 113 | 71 | q |
| 116 | 111 0010 | 162 | 114 | 72 | r |
| 117 | 111 0011 | 163 | 115 | 73 | s |
| 118 | 111 0100 | 164 | 116 | 74 | t |
| 119 | 111 0101 | 165 | 117 | 75 | u |
| 120 | 111 0110 | 166 | 118 | 76 | v |
| 121 | 111 0111 | 167 | 119 | 77 | w |
| 122 | 111 1000 | 170 | 120 | 78 | x |
| 123 | 111 1001 | 171 | 121 | 79 | y |
| 124 | 111 1010 | 172 | 122 | 7A | z |
| 125 | 111 1011 | 173 | 123 | 7B | { |
| 126 | 111 1100 | 174 | 124 | 7C | \| |
| 127 | 111 1101 | 175 | 125 | 7D | } |
| 128 | 111 1110 | 176 | 126 | 7E | ~ |

컴퓨터에서 글자는 수의 값인 체제진법으로 정하여 나타낸다.
초기 2진법의 숫자로 글자를 나타내고 있었다.
이러한 방법의 것이 아스키코드라는 것이다.
위의 표에서 CODE라는 알파벳의 글자는 찾아보면 컴퓨터에서는
C →1000011
O →1001111
D →1000100
E →1000101
로 나타내게 된다.
7비트 128개의 문자 값을 갖는 아스키코드에서 33개 제어문자 영문 알파벳 52개 숫자 10개 공백문자 1개를 문장부호로 사용할 수 있는 나머지 문자 값에는 32개가 남는다.
문서작성에 필요한 문장부호와 연산을 하기 위해서는 34개는 되어야 하는 것으로 아스

키코드에는 2개가 부족하다.

알파벳 자소는 가독성이 가장 높은 굴림체의 서체를 사용할 경우 연산부호인 × 를 사용할 수 없다,

알파벳 24번째 자소인 'x'와 연간부호 '× ' 는 같은 모양이어서 '*' 를 배열하여 사용하며, 나눗셈의 연산부호 '÷ ' 는 128개의 문자 값으로는 부족하여 사용할 수가 없어 '/' 를 대신 사용한다.

또한 참조부호 '※' 도 문자 값 부족으로 사용할 수가 없다.

이러한 사정 때문에 자소 판을 함부로 개발하여 사용할 수가 없다. 굴림체를 사용할 경우 '0' 의 수자와 알파벳 16번째'O'가 구분이 되지 못하며, 수 '1'과 알파벳 9 번째 대문자 'I', 알파벳 14 번째 소문자 'l'의 구별이 전혀 아니 된다고 본다.

이러한 관계로 수자배열은 계산기와 같이 할 수 없고 계산기처럼 입력하고 사용할 수도 없다.

따라서 계산을 자소 판에서 바로 할 수 없는 결점도 있다.

유니코드를 사용하는 지금도 알파벳의 자소 판은 더 이상 발전을 못하고 있다.

이것이 2013년 현재 $2^{1}$ □ = 65,536개의 문자 값이 확보되고 이어

$2^{3}$ $^{2}$ = 4,294,967,296,

$2$□$^{4}$ = 115,281,504,606,846,976

시대가 되면서 반도체산업에 있어서 알파벳문화권이 한국을 따라 잡을 수 없는 이유이기도 한 것이다.

반도체 산업은 절대적으로 속도산업이어서 알파벳자소 판은 입력속도가 늦어지고 입력공간이 늘어나고 가독성이 떨어져서 실행과 해독속도가 늦어져 개발의 발목을 잡는 요인이 되고 도태의 원인이 된다.

10 한글 자소 판

컴퓨터자소 판은 컴퓨터에 글자를 입력하기 위한 선택단추로 타자기의 글쇠와 같은 글자입력 기능을 하며 컴퓨터자소 판은 물리적자소판과 가상자소 판의 2가지로 구분되어 되어 있다.

자소 판을 키보드(keyboard)라고도 하며 가상 키보드는 키보드가 가지고 있는 것보다 더 적은 단추를 가지고 있는 컨트롤러를 위해 가상 확장 역할을 하는 컴퓨터 프로그램이다. 데스크톱 컴퓨터의 세상에서, 윈도 XP 프로그램 화상 키보드는 실제 키보드나 컴퓨터 마우스로 움직이는 가상 키보드처럼 동작한다.

스타일러스 기반의 PDA에서, 사용자가 운영 체제가 만들어 주는 화상 키보드를 두드리면서 텍스트를 입력하는 것이 보통이다. 이러한 키보드들은 자주 컴퓨터 키보드보다 버튼이 얼마 없는 시스템을 위해 가상 머신(에뮬레이터)의 기능으로 쓰인다.

가상 키보드는 또한 육체적 제한 때문에 일반 키보드를 쓰지 못하는 사람들에게도 쓰인다.

우분투 리눅스에 포함된 "OnBoard" 가상 키보드

가상 키보드는 물리 키보드인 물리 텍스트 입력 장치들과는 다르게 기능을 하므로, keystroke logging(키보드로 입력하는 키 정보를 외부의 다른 사람이 알 수 있게 함)에 대해 어느 정도 보안을 제공한다.

키보드 레이아웃이 일반적으로 실제 키보드와 비슷할지라도, 물리적 키보드들에 익숙

한 사람들은 자주 상당한 양의 시간을 가상 키보드로 빠른 입력을 배우는 데 몸을 바쳐야 한다. 물리적 키보드와 달리 손가락으로 키들을 느끼지 못해 키들의 위치에 친숙하지 못할 수도 있어서 사용자에게 압박을 줄 수 있다.
그러나 PDA의 외부 물리 키보드와 다른 전기 장치들을 개발하는 데 도움을 준다.
컴퓨터에는 자소 판과 같이 사용되어야 하는 것으로 자소 판이 나쁘면 불편함은 물론 입력속도가 현저히 떨어진다.
또한 사용하는 자소에 따라 자소 판은 달라짐으로 자소의 특성을 잘 살려야 한다.
필자는 한글에서 사용하던 기본모성소를 조립하여 파생모성소로 생성하던 과정을 생략하고 9개의 파생모성소를 바로 입력시키도록 단축하고 숫자와 연산부호를 오른쪽에 모아 배열하여 연산이 바로 일어나도록 설계하였다.
그동안 알파벳자소 판에 얹혀서 사용하던 방식이 매우 불편하여 독자적인방식으로 문장부호를 자소 판에 더 배열하여 문서작성 속도를 빠르게 한 것이다.

필자는 알파벳자소가 결점이 많다는 것을 오래전부터 느껴온 것으로 자소 판을 알파벳 방식에서 빨리 벗어나야 한글자소의 특성을 살릴 수 있다고 생각했다.

개발 자소 판은 72개의 선택단추만으로도 계산기에서와 같이 연산부호 '+, −, ×, ÷, ='를 완비하였고, 계산기의 숫자 배열과 같이 하였으며, 연산기능이 바로 실행되며, 2중모음 'ㅐ ㅔ ㅒ ㅖ ㅚ ㅘ ㅝ ㅟ ㅢ 와 3중모음 ㅙ ㅞ'를 하나로 된 선택단추로 배열했다.

또한 문서작성에 많이 사용되는 참조부호 '※'도 배열하여 입력속도는 한결 빨라지게 되었다.

한글 자소를 7비트로 인코딩할 경우

- 32개의 출력 불가능한 제어 문자들
- 자소 51개

기본 자음자소 14 ;ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ

파생자성자소 16 ;ㄲ ㄳ ㄵ ㄶ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅃ ㅄ ㅆ ㅉ

기본모성소 10 ; ㅏ ㅑ ㅓ ㅕ ㅗ ㅛ ㅜ ㅠ ㅡ ㅣ

파생모성소 11 ; ㅐ ㅔ ㅒ ㅖ ㅚ ㅘ ㅝ ㅟ ㅢ ㅙ ㅞ

- 숫자 ; 10개
- 부호와 같은 특수문자 34개
- 공백문자 1개

한글기호

| 순위 | 이진법 | 팔진법 | 십진법 | 십육진법 | 약자 | 설명 | 한글설명 |
|---|---|---|---|---|---|---|---|
| 1 | 000 0000 | 000 | 0 | 00 | 공백 | 공백I | 공백 문자 |
| 2 | 000 0001 | 001 | 1 | 01 | 머리시작 | 머리시작 | 헤더 시작 |
| 3 | 000 0010 | 002 | 2 | 02 | 본문시작 | 본문시작 | 본문 시작, 헤더 종료 |
| 4 | 000 0011 | 003 | 3 | 03 | 본문종료 | 본문종료 | 본문 종료 |
| 5 | 000 0100 | 004 | 4 | 04 | 전송종료 | 전송종료 | 전송 종료, 데이터 링크 초기화 |
| 6 | 000 0101 | 005 | 5 | 05 | 응답요구 | 응답 요구 | 응답 요구 |
| 7 | 000 0110 | 006 | 6 | 06 | 긍정응답 | 긍정응답 | 긍정응답 |
| 8 | 000 0111 | 007 | 7 | 07 | 경고음 | 경고음I | 경고음 |
| 9 | 000 1000 | 010 | 8 | 08 | 뒤 공백 | 뒤 공백 | 백스페이스 |
| 10 | 000 1001 | 011 | 9 | 09 | 수탭 | 수평 탭 | 수평 탭 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 11 | 000 1010 | 012 | 10 | 0A | 개행 | 개행 | 개행 |
| 12 | 000 1011 | 013 | 11 | 0B | 수직탭 | 수직 탭 | 수직 탭 tap |
| 13 | 000 1100 | 014 | 12 | 0C | 뒷장 | 뒷장 | 다음 페이지 |
| 14 | 000 1101 | 015 | 13 | 0D | 복귀 | 복귀 | 복귀 |
| 15 | 000 1110 | 016 | 14 | 0E | SO | 확장문자 시작 | 확장문자 시작 |
| 16 | 000 1111 | 017 | 15 | 0F | SI | Shift In | 확장문자 종료 |
| 17 | 001 0000 | 020 | 16 | 10 | DLE | Data Link Escape | 전송 제어 확장 |
| 18 | 001 0001 | 021 | 17 | 11 | DC1 | Device Control 1 | 장치 제어 1 |
| 19 | 001 0010 | 022 | 18 | 12 | DC2 | Device Control 2 | 장치 제어 2 |
| 20 | 001 0011 | 023 | 19 | 13 | DC3 | Device Control 3 | 장치 제어 3 |
| 21 | 001 0100 | 024 | 20 | 14 | DC4 | Device Control 4 | 장치 제어 4 |
| 22 | 001 0101 | 025 | 21 | 15 | NAK | Negative Acknowledgement | 부정응답 |
| 23 | 001 0110 | 026 | 22 | 16 | SYN | Synchronous idle | 동기 |
| 24 | 001 0111 | 027 | 23 | 17 | ETB | End of Transmission Block | 전송블록 종료 |
| 25 | 001 1000 | 030 | 24 | 18 | CAN | Cancel | 무시 |
| 26 | 001 1001 | 031 | 25 | 19 | EM | End of Medium | 매체 종료 |
| 27 | 001 1010 | 032 | 26 | 1A | SUB | Substitute | 치환 |
| 28 | 001 1011 | 033 | 27 | 1B | ESC | Escape | 제어기능 추가 |
| 29 | 001 1100 | 034 | 28 | 1C | FS | File Separator | 파일경계 할당 |
| 30 | 001 1101 | 035 | 29 | 1D | GS | Group Separator | 레코드 그룹경계 할당 |
| 31 | 001 1110 | 036 | 30 | 1E | RS | Record Separator | 레코드 경계 할당 |
| 32 | 001 1111 | 037 | 31 | 1F | US | Unit Separator | 장치 경계 할당 |
| 33 | 111 1111 | 177 | 127 | 7F | DEL | Delete | 삭제 |
| 34 | 010 0000 | 040 | 32 | 20 | □ | 공백 null | |
| 35 | 010 0001 | 041 | 33 | 21 | ! | | |
| 36 | 010 0010 | 042 | 34 | 22 | . | | |
| 37 | 010 0011 | 043 | 35 | 23 | : | | |
| 38 | 010 0100 | 044 | 36 | 24 | ; | | |
| 39 | 010 0101 | 045 | 37 | 25 | " | | |
| 40 | 010 0110 | 046 | 38 | 26 | ' | | |
| 41 | 010 0111 | 047 | 39 | 27 | \| | | |
| 42 | 010 1000 | 050 | 40 | 28 | ₩ | $ | |
| 43 | 010 1001 | 051 | 41 | 29 | / | | |
| 44 | 010 1010 | 052 | 42 | 2A | % | | |
| 45 | 010 1011 | 053 | 43 | 2B | + | | |
| 46 | 010 1100 | 054 | 44 | 2C | − | | |
| 47 | 010 1101 | 055 | 45 | 2D | × | | |
| 48 | 010 1110 | 056 | 46 | 2E | ÷ | | |
| 49 | 010 1111 | 057 | 47 | 2F | = | | |
| 50 | 011 0000 | 060 | 48 | 30 | ( | | |
| 51 | 011 0001 | 061 | 49 | 31 | ) | | |
| 52 | 011 0010 | 062 | 50 | 32 | 1 | | |
| 53 | 011 0011 | 063 | 51 | 33 | 2 | | |
| 54 | 011 0100 | 064 | 52 | 34 | 3 | | |
| 55 | 011 0101 | 065 | 53 | 35 | 4 | | |
| 56 | 011 0110 | 066 | 54 | 36 | 5 | | |
| 57 | 011 0111 | 067 | 55 | 37 | 6 | | |
| 58 | 011 1000 | 070 | 56 | 38 | 7 | | |
| 59 | 011 1001 | 071 | 57 | 39 | 8 | | |
| 60 | 011 1010 | 072 | 58 | 3A | 9 | | |
| 61 | 011 1011 | 073 | 59 | 3B | 0 | | |
| 62 | 011 1100 | 074 | 60 | 3C | @ | | |

| | | | | | |
|---|---|---|---|---|---|
| 63 | 011 1101 | 075 | 61 | 3D | ᯾ |
| 64 | 011 1110 | 076 | 62 | 3E | ᯿ |
| 65 | 011 1111 | 077 | 63 | 3F | ? |
| 66 | 100 0000 | 100 | 64 | 40 | _ |
| 67 | 100 0001 | 101 | 65 | 41 | ᄀ |
| 68 | 100 0010 | 102 | 66 | 42 | ᄂ |
| 69 | 100 0011 | 103 | 67 | 43 | ᄃ |
| 70 | 100 0100 | 104 | 68 | 44 | ᄅ |
| 71 | 100 0101 | 105 | 69 | 45 | ᄆ |
| 72 | 100 0110 | 106 | 70 | 46 | ᄇ |
| 73 | 100 0111 | 107 | 71 | 47 | ᄉ |
| 74 | 100 1000 | 110 | 72 | 48 | ᄋ |
| 75 | 100 1001 | 111 | 73 | 49 | ᄌ |
| 76 | 100 1010 | 112 | 74 | 4A | ᄎ |
| 77 | 100 1011 | 113 | 75 | 4B | ᄏ |
| 78 | 100 1100 | 114 | 76 | 4C | ᄐ |
| 79 | 100 1101 | 115 | 77 | 4D | ᄑ |
| 80 | 100 1110 | 116 | 78 | 4E | ᄒ |
| 81 | 100 1111 | 117 | 79 | 4F | ᄀᄀ |
| 82 | 101 0000 | 120 | 80 | 50 | ᄁ |
| 83 | 101 0001 | 121 | 81 | 51 | ᄈ |
| 84 | 101 0010 | 122 | 82 | 52 | ᄊ |
| 85 | 101 0011 | 123 | 83 | 53 | ᄍ |
| 86 | 101 0100 | 124 | 84 | 54 | ᄀᄉ |
| 87 | 101 0101 | 125 | 85 | 55 | ᄂᄌ |
| 88 | 101 0110 | 126 | 86 | 56 | ᄂᄒ |
| 89 | 101 0111 | 127 | 87 | 57 | ᄅᄀ |
| 90 | 101 1000 | 130 | 88 | 58 | ᄅᄆ |
| 91 | 101 1001 | 131 | 89 | 59 | ᄅᄇ |
| 92 | 101 1010 | 132 | 90 | 5A | ᄅᄉ |
| 93 | 101 1011 | 133 | 91 | 5B | ᄅᄐ |
| 94 | 101 1100 | 134 | 92 | 5C | ᄅᄒ |
| 95 | 101 1101 | 135 | 93 | 5D | ᄆᄒ |
| 96 | 101 1110 | 136 | 94 | 5E | ᄇᄊ |
| 97 | 101 1111 | 137 | 95 | 5F | ᅡ |
| 98 | 110 0000 | 140 | 96 | 60 | ᅢ |
| 99 | 110 0001 | 141 | 97 | 61 | ᅣ |
| 100 | 110 0010 | 142 | 98 | 62 | ᅤ |
| 101 | 110 0011 | 143 | 99 | 63 | ᅩ |
| 102 | 110 0100 | 144 | 100 | 64 | ᅪ |
| 103 | 110 0101 | 145 | 101 | 65 | ᅮ |
| 104 | 110 0110 | 146 | 102 | 66 | ᅯ |

| | | | | | | |
|---|---|---|---|---|---|---|
| 105 | 110 0111 | 147 | 103 | 67 | ㅡ | |
| 106 | 110 1000 | 150 | 104 | 68 | ㅣ | |
| 107 | 110 1001 | 151 | 105 | 69 | ㅐ | |
| 108 | 110 1010 | 152 | 106 | 6A | ㅒ | |
| 109 | 110 1011 | 153 | 107 | 6B | ㅔ | |
| 110 | 110 1100 | 154 | 108 | 6C | ㅖ | |
| 111 | 110 1101 | 155 | 109 | 6D | ㅚ | |
| 112 | 110 1110 | 156 | 110 | 6E | ㅘ | |
| 113 | 110 1111 | 157 | 111 | 6F | ㅝ | |
| 114 | 111 0000 | 160 | 112 | 70 | ㅟ | |
| 115 | 111 0001 | 161 | 113 | 71 | ㅢ | |
| 116 | 111 0010 | 162 | 114 | 72 | ㅙ | |
| 117 | 111 0011 | 163 | 115 | 73 | ㅞ | |
| 118 | 111 0100 | 164 | 116 | 74 | [ | |
| 119 | 111 0101 | 165 | 117 | 75 | ] | |
| 120 | 111 0110 | 166 | 118 | 76 | { | |
| 121 | 111 0111 | 167 | 119 | 77 | } | |
| 122 | 111 1000 | 170 | 120 | 78 | ( | |
| 123 | 111 1001 | 171 | 121 | 79 | ) | |
| 124 | 111 1010 | 172 | 122 | 7A | & | |
| 125 | 111 1011 | 173 | 123 | 7B | ^ | |
| 126 | 111 1100 | 174 | 124 | 7C | ` | |
| 127 | 111 1101 | 175 | 125 | 7D | ~ | |
| 128 | 111 1110 | 176 | 126 | 7E | 나가기 | 지우기 Delete |

대체로 같은 내용을 컴퓨터 실행 면에 실행시켜보면 알파벳글자로는 한글실행공간의 0.3 ~ 0.5배의 공간이 더 필요하게 된다.

왜 그러한 것인가는 알파벳과 같은 낱개 자소는 글줄은 양쪽정열방식으로 사용할 수 없다,

긴 단어가 글줄 말미에서 끝나지 못할 경우 다음의 새로운 줄로 바뀌기 때문에 앞줄의 글자사이에는 불필요한 넓은 공간이 생긴다.

낱개 자소는 구조상으로 낱말과 낱말사이 필요이상의 공백을 사용해야하게 되어 있다.

낱개 자소는 단어형성에서 단락이 생기면 다른 낱말로 갈라지게 되므로 줄의 말미에서 나열이 끝나지 못하면 부득불 다음 줄로 넘어가서 실행되는 필연성 때문에 낱말과 낱말의 사이를 구분하는 공백은 매줄 마다 달라질 수밖에 없어서 필요이상의 공백이 생기게 마련이다.

또 한글의 수직수평조립글자를 해부해보면 초성 중성 종성이 보통 2 ~ 5개의 자소들이 조립되어 하나의 완전한 글자를 생성하고 이들 글자가 모여 하나의 낱말이 이루어져

낱말공간을 조성하고 있으나 알파벳 낱개자소는 무조건 나열된 하나의 단락이 낱말을 만들기 때문에 단어공간도 길고 글자공간도 넓어졌다 좁아졌다 하는 것이다.

이러한 속성으로 인하여 한글의 글자공간보다는 항상 0.3 ~ 0.5배의 글자공간이 더 필요하게 된다.

입력속도에서도 현저한 차이가 일어난다.

알파벳글자를 사용하는 언어습관은 단수는 'a'로, 특정낱말은 낱말 앞에는 반드시 'the'의 사용해야하고 문장의 첫 글자에도 반드시 대자를 사용해야하는 언어습관은 한글언어습관에서 보면 매우 번거롭고 불필요한 글자를 더 입력해야하는 낭비가 일어난다.

글자입력 판에서 운용되는 손가락의 사용에 있어서도 양손의 손가락이 번갈아가면서 또는 동시에 입력할 경우 입력속도는 빨라지나 알파벳글자입력에서는 때로는 왼손의 손가락으로만 사용하거나 오른손의 손가락만을 사용하게 되는 경우가 빈번하게 발생하는 것이다.

예를 들면 'extends' 와 'earth' 와 같은 글자의 배열이다.

'extends' 와 'earth' 의 낱말이 문장의 첫 글자에 해당한다면 대자로 사용해야 하므로 모두가 왼손 편에 있어서 매우불편하고 입력속도를 현저하게 저하시키는 요인이 된다.

최고능력의 타자수가 알파벳의 경우 분당500타를 치는데 비해 한글은 700타를 넘는다고 하였고 새로 개발된 자소 판은 900타 효율이 있다.

한국인의 빨리 빨리 혼이 여기에서 길들여 진 것이라고 봐야 한다.

한국인은 일상생활에서 조립공간을 생각하면서 살아가고 있다.

한국인의 뇌가 우수한 것은 한글을 사용하여 다듬어진 결과라고 생각된다.

이렇게 우수한 글자를 사용하는 첨단반도체국가가 자존심을 잃은 체 한글에 의한 컴퓨터운영체제를 개발하지 못하고 알파벳문화권에서 그들의 필요에 의해 개발된 프로그램을 그대로 이용하고 있는 것은 매우 유감스런 일이다.

한글은 기본자소가 알파벳에 52개비해 51개로 1개가 적으며 숫자나 문장부호, 연산부호와 확실하게 구분되고 자소 또한 모성소와 자성소로 구분되는 데에다 초성자성 + 중성모성 + 종성자성으로 조립되도록 설계되어 있어서 입력기 자소배열이 반드시 양손을 번갈아 동시에 사용하도록 되어 있어서 글자입력속도는 매우 빠른 반면 LCD모니터 실행공간을 적게 차지하는 것은 정보생성과 전달속도를 최적화하고 있다.

무엇보다 중요한 것은 가감승제의 계산식이 바로 입력되고 계산기 기능까지 겸하고 있어서 사무기기로서는 이보다 더 발전 할 수는 없다.

또 하나 계산기에서 자리 수는 작업공간에서 크게 제한을 받아 33자리의 수 등으로 그 이상 입력이 불가능하지만 LCD모니터에서는 무한대로 입력이 가능하다.

종합적으로 환산하면 한글과 자소 판은 여타 세계 각국의 자소보다 정보생성속도가 30% ~ 50%가량 빠른 반면 정보생성 공간 또한 30% ~ 50% 적게 소요된다.

즉 한글은 정보생성의 속도와 공간이 30% ~ 50% 적어도 되는 반면 전달속도가 빨라지게 된다.

낱말인 단어에 있어서도 아래와 같이 한글의 2 ~ 3배의 길이가 요구되는 것을 알 수가 있다.

saccharification

당화

alcoholfermentation
알코올발효

반도체에 글자입력에 대한 이해를 돕기 위해 반도체속성에 대해 더 언급하고자 한다.

많이 사용하고 있는 반도체기기는 사무기기인 컴퓨터로서 글자입력기에서 좌우 두 손의 손가락 10개를 동시에 사용할 수 있어야 편리하고 모든 문장부호와 계산식으로 바로 입력되고 계산기 기능도 바로 되어야 한다.

컴퓨터는 하드웨어에 Windows와 Linux의 양대 운영체제(operating system)가 있고 이들 양 운영체제에 프로그램을 내장하여 사용하고 있다.

컴퓨터에 프로그램을 내장하는 것은 컴퓨터는 인간과 같이 스스로 창의적인 생각을 하지 못하기 때문이다.

2진법에 의한 상상을 초월할 정도의 무한한 자료저장 공간이 개발 될 수는 있지만 입출력에 있어서는 모두 기계적 명령으로 이루어 진 자소 판을 사용해야만 한다.

인간은 10진법으로 생각하고 스스로 처리하지만 반도체기기는 2진법에 의한 기계어의 명령에 의해 이루어진다.

인간만이 스스로 생각하고 표현하는 언어를 가지고 있다.

이러한 인간의 언어를 고급언어라고 하고 컴퓨터에 필요한 언어를 저급언어라고 한다.

저급언어에는 기계어가 있고 고급화한 어셈블리어가 이다.

반도체기기를 위한 언어가 여러 형태로 개발되어 있다.

그러나 대한민국이 첨단반도체국가이지만 한글로 기계어를 개발하지 못 하였다.

또 운영체제에 있어서도 그저 알파벳문화권에서 그들의 필요에 의해서 만들어 놓은 체제를 그대로 사용할 뿐이다.

원인과 이유는 Windows와 Linux의 양대 운영체제(operating system)이외의 새로운 운영체제나 기계어를 개발한다고 하드라도 대중성은 없다는 것으로 들 수 있다.

그러나 절대적인 것은 없다고 보아야 한다.

운용체제이든 개발언어이든 컴퓨터의 사용방법은 날로 발전하고 달라지는 추세가 계속되어 왔고 앞으로도 계속된다면 머지 않는 날에 한글에 의한 컴퓨터사용주도시대가 오기 마련이다.

컴퓨터 언어는 B언어에서 시작하여 C언어가 1972년경 시스템 PDP-11에서 운용되는 운영체계 Unix를 개발하기 위한 언어로 지금도 전 세계적으로 많이 활용되고 있다.

컴퓨터를 위해 2진법의 글자 값을 배열하면 7비트로는 128개의 문자열 값을 배열할 수 있게 된다.

이러한 특성을 만족시키는 글자로는 바로자소이고 낱자소인 알파벳만이 가능하다.

한글자소는 바로자소이나 11,172개 조립글자 된 조립자소이므로 지구상에서 중국 다음으로 많은 글자를 보유하고 있다.

다시 말해 11,172개의 글자 값을 배열할 수 있는 체계의 프로그램

이 내장된 컴퓨터가 아닌 이상 사용할 수가 없다.

아스키코드수준으로는 128개가 아닌 한글 11,172개의 글자 값을 소화할 수가 없다는 결론이 쉽게 나온다.

반도체기기는 2진법에 기초를 두고 7비트가 아닌 8비트 정보단위인 바이트를 사용할 수 있는 8진법 16진법 32진법 64진법으로 쉽게 환산되는 진법체계로 확대 개발되고 자료의 저장능력공간도 확대 개발되고 있어서 상상을 초월할 정도로 발전가능성이 크다 할 것이다.

한글자소는 이렇게 무한가능의 반도체시대에 가장 최적화된 자소이다.

한글자소는 반도체 사용기술이 16진법이상체계의 수준에 진입하면 서 반도체기기로서 온전하게 사용될 수 있는 소지를 발휘하게 될 수가 있어서 비록 반도체기술개발의 후발주자이드라도 앞으로의 반도체기술개발은 한글자소에 의해서만이 발전가능성이 있다는 결론이 나온다.

알파벳 자소는 바로자소이며 낱 자소에 해당하지만 글자를 입력하는 자소 판에서는 한글자소 판보다 2개의 단추가 더 있어야 한다.

한글자소가 51개인데 반해 알파벳자소는 52개라서 1개가 더 있어야 하고, 대소문자를 선택해야하는 문자선택의 기능단추가 1개 더 있어야 한다.

또 숫자와 글자구분이 되지 못하는 결점이 있고 가감승제(+,-.x,÷ ) 의 연산부호가 자소 판에 배열될 수도 없다.

낱개 자소의 나열방식의 단어는 컴퓨터에서 실행공간을 넓게 하는 결점이 있다.

단수 복수 구분과 지정관사를 사용 하는 언어습관은 정보생성과 생성 공간 그리고 전달속도를 느리게 한다.

이러한 이유 때문에 한글을 반도체기기개발을 위한 프로그램언어로 사용할 수 있도록 개발해야 한다는 것이다.

가감승제 연산(+, −, ÷, ×, = )부호의 사용 역사는 연산부호에 따라 10세기에서 1600년대로 400년 가까운 역사가 되며 인간문화는 셈을 하는 문화로서 생활 속에 셈은 필수적인 것이고 컴퓨터는 사무용기기임에도 문서 작성에서 수식과 계산 기능을 바로 실행되도록 개발해온 것이 없다.

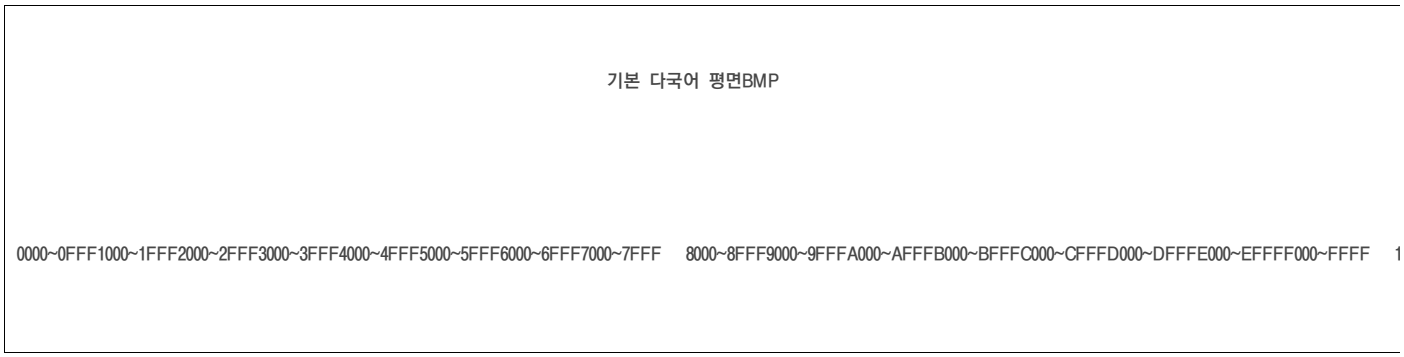
모든 컴퓨터가 수식과 계산 기능이 분리 되어있어서 따로 전환과정을 거치는 불편이 있다.

알파벳 자소로는 구조상문제가 있기 때문이다.

우리가 너무도 익숙하게 사용하고 있는 사칙 연산과 연산기호도 반드시 입력자판에서 바로 실행이 시켜야만 한다.

## 11 발전하는 진법체계

아래 표는 16진법으로 확장되는 글자 값으로 인코딩된 유니코드의 목록범위이다. 16진법으로 확장하여 개발한 코드이기 때문에 65,536개까지 글자 값을 배열할 수 있다.




표에서 260,000여개의 문자를 값을 배당할 유니코드목록범위를 BMP, SMP, SIP, TIP, SSP 5개로 나누어 문자 값을 인코딩하고 있다.

중국의 상형글자가 50,000개를 할당하고 있으며 한글이 12,000여개 할당되어 있다.

기본다국어평면(BMP)에서 65,536개의 글자 값 배당공간에 중국글자 20,000 한글 12,000으로 50%를 차지하고 그 밖에 여타 다국어로 배당되고 많은 공간이 비어 있다.

중국의 50,000개의 글자는 보조 상형 문자 평면(SIP)에 따로 배당되어 있다.

반도체에서 글자의 사용방법은 계속 발전하고 있으며 그 대표적인 예로서 Java는 프로그램운용체제에 있어서도 Java virtual machine 이라는 가상의 기계를 제작하여 모든 자료들을 원활하게 실행되도록 체제를 갖추고 있다.

또 인터프리터 (interpreter)라는 기계언어상의 고급언어해석기를 사용하여 기존의 컴

파일러(compiler) 방식이 아닌 방식으로도 실행이 되는 방법이 개발되어 사용되고 있다

여기서 말하는 고급언어라는 것은 인간이 사용하는 수준의 고급언어가 아니고 저급언어인 기계언어수준상의 고급을 말하는 것으로 고급언어를 인간언어로 이해하는 오류가 없기를 바란다.

고급언어해석기인 인터프리터는 고급언어로 작성된 원시 코드 명령문을 한 번에 한 줄씩 읽어서 실행하는 프로그램이며 고급언어로 작성된 프로그램을 실행하는 데에는 2가지 방법이 있다. 가장 일반적인 방법은 프로그램을 컴파일(compile)하는 것이고 다른 하나는 인터프리터에 통과시키는 방법이다. 인터프리터는 명령문을 중간 형태로 번역한 뒤 실행하고 컴파일러는 고급 명령문을 기계어로 직접 번역한다.

컴파일 된 프로그램은 인터프리터를 이용해 실행시키는 것보다 빨리 실행되지만 프로그램의 크기가 큰 경우 상당한 시간이 걸릴 수 있다. 이와 달리 인터프리터는 기계어 명령문이 만들어지는 컴파일 단계를 거칠 필요가 없으며 작은 크기의 고급 프로그램을 즉시 실행시킬 수 있다. 베이직(BASIC)이나 LISP(list processing) 같은 언어는 인터프리터에 의해서만 실행되도록 설계되었다

이러한 방법으로 호환성 프로그램이 개발되어 내장되지 않으면 사용할 수가 없다.

특히 많은 수를 가진 글자는 전환 자소로 취급하는 전환프로그램이 있어야 한다.

한글의 경우는 전환방법이 아닌 바로 자소이다.

그러나 워낙 많은 조립글자를 생성하고 있어서 이에 필요한 환경이 구축되어야 하고 이러한 환경프로그램이 환경프로그램 간에 호환성이 제대로 이루어지지 못해 깨어지는 글자로 실행되는 사례가 많았다.

아직까지 완벽하다고 장담하기에는 이르다.

위의 표는 유니코드 E000~EFFF의 기본 다국어 평면(BMP) E800에서 이미지로 스캔한 것이다.

이것을 복사하면 아래와 같이 된다.

| U+ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E800 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| E810 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| E820 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| E830 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| E840 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| E850 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| E860 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |

옛한글인지 중국글자 부수인지 국적불명인 것이다.

이러한 원인은 사용상에 서로 다른 환경에 있다.

웹과 한글에 있어서 자주 깨어지는 글자발생으로 한글은 조립자소로서 웹과의 처리는 무척 까다롭다. 그 이유는 사용자의 환경이 매우 다르다는 데 있다. 웹 프로그래밍을 하려면, 운영체제의 기본 인코딩, Java 소스 코드의 인코딩, JSP 파일의 인코딩,

HTTP 요청의 인코딩, HTTP 응답의 인코딩, 데이터베이스의 인코딩, 파일의 인코딩 등 이렇게 많은 인코딩과 마주하게 된다.

여기 웹에서 한글이 왜 깨지는 것은 브라우저 인코딩 값과 서버 인코딩 값이 다르기 때문이다.

필자의 견해로는 우선 한글의 정의가 확실해야 한다고 본다.

조합형한글’로 할 경우와 ‘완성형한글’로 할 경우 모두가 궁여지책으로 관리한 방법으로 과학적인 올바른 방법으로 볼 수는 없다.

필자는 한글을 애초부터 ‘조립 자소, 바로 자소’로 보았다.

중국의 글자는 반도체기기상이 아닌 상태는 획수를 기본으로 하는 조립 자소’이지만 조립 자소로 운용할 수 없다

자전에서 277개 부수로 나누어 조립되기 때문에 반도체기기상에서 277개의 입력단추를 배열하여 사용할 수는 없다.

50,000개 중국글자를 277개의 부수로 분류하여 수록한 것이다.

특이한 것은 중국글자에 ‘0’로 된 부수나 글자는 없다.

반도체기기인 컴퓨터에서는 277개의 입력자소단추로 직접 입력할 수 없는 글자이고 설사 조립 자소로 입력된다고 할지라도 컴퓨터는 문자 값으로 다루어지게 되므로 결과적으로는 낱개 자소로 다루어진다.

따라서 중국글자는 50,000개의 낱개자소라는 것이 된다.

낱개 자소는 전환방식을 사용하게 된다.

따라서 입력속도가 느리다는 결점이외에 사용상에는 문제발생이 없다고 본다.

한글 11,223개 글지도 중국의 50,000개 글자 값과 마찬가지로 글자 값으로 다루어진다.

52개의 알파벳이나 글자 값으로 취급되는 현상에는 다를 바 없다는 것이다.

즉,

글자 값 = 중국글자 50,000개 값 = 한글 11,223개 값= 알파벳 26개 값과 같이 동등한 글자 값의 등식으로 취급된다는 것이다.

모든 글자는 입력이후 하나하나의 글자 값 수치로 다루어지기 때문에 반도체기기에서 “완성형이니 조합형이니 일본글자 중국글자 영어”로 분류하는 것은 전혀 의미가 없는 것으로 이해하여야 한다.

한글의 경우도 자소 판에서 바로 입력할 수 있지만 입력이후부터 사이버 웹상에서는 글자 값으로 취급되기 때문에 사용 환경을 일치시키든가 호환성이 있도록 하는 것만이 중요하며 이 원칙이 지켜지지 않으면 글자가 깨어지는 현상으로 아무런 소용이 없다는 것이다.

한글을 표준화하는 과정에서 ‘조합형한글’과 ‘완성형한글’로 2원화하는 다른 환경은 호환성이 없어 가독성이 없는 깨어진 글자가 되어 버린다.

한글은 600년간 사용하면서 해방이후 현대화에 노력했고 당초의 옛한글과는 사용방법

이 많이 바뀌었다.

옛한글을 반도체기기의 글자로 사용하기에는 많은 문제가 있다.

따라서 반도체시대에는 반도체시대적성에 맞추어야 하며 2진법, 8진법, 16진법, 32진법, 64진법으로 발전하기 위해서는 반도체적성으로 사용 환경을 모두 정비해야 한다.

중국글자는 50,000개의 문자 값을 가져야 하고 전환과정을 거쳐야 하는 사용상의 문제가 있어서 조립 자소로서는 의미가 없어 방대한 낱개 자소로 취급되어야 하므로 굳이 새로운 사용방법의 프로그램을 로 개발할 필요성이 없고 개발의 진전성도 없다.

한글은 반도체기기의 초기단계가 아닌 성숙단계에서는 조립자소로서 개발할 가치가 매우 큰 것이므로 미래를 위해 프로그램과 운영체제를 현시점에서 확실한 정비가 필요하다.

앞서 중국글자에서 언급 하였듯이 반도체기기의 웹상에서는 어느 글자이든 글자 값을 가지는 낱개글자로 취급되어야 하는 특성을 이해하고 사용 환경에서만은 조립 자소도 동일한 환경으로 구축하여야 한다.

좀 더 구체적으로 언급한다면 한글은 조립 자소로서 기본자소 24개만 고집할 수 없고 기본자음 자소에 14개에 근거하여 파생자음 자소 16개가 합쳐지고 기본모음 자소 10개에 파생모성자소 11개가 합해져서 자소가 51개 글자 값으로 배당되고 초성 + 중성으로만 조립되는 399개 글자에도 글자 값이 배당되어지고 종성 27개로 다시 더 하여 조립되는 10,773개의 글자에도 글자 값이 배당되어지므로 이를 정리하면 한글은

기본자음 14개

파생자음 16개

기본모음 10개

파생모음 21개

총 조립 자소 51개

총 조립자소 51개

초성 + 중성 399개

초성 + 중성+ 종성 10,773개

조립글자 11,172개

한글합계 11,223개

의 글자 값이 배당된다.

이렇게 글자 값이 배당 되면 알파벳의 52개 대소글자나 중국의 50,000개 글자나 한글의 11,223개의 글자가 반도체기기상에서 모두 같은 자격으로 낱개글자로 취급될 수 있다.

인코딩에서 글자 값을 가진 동일한 자격의 글자로 취급되는 웹 환경을 부여하면 글자 값으로 충돌이 일어나는 사고는 없어진다.

앞으로 프로그램언어가 점차 고급화 하게 되고 호환성이 갖추어지므로 웹상에서의 충돌은 사라지게 된다.

이렇게 반도체기기를 통한 글자이용방법이 바뀌게 되면 고전적인 글자분류방식이나 글

자 사용방법은 사실상 의미가 없는 것으로 되어 버린다.

글자가 인간의 의사표시 수단으로 필기구류에 의존하는 시대에는 표음문자니 표의 문자니 하는 분류방법을 사용하였고 글자를 빠르게 필기할 수 있도록 하기 위한 필기체가 발달하게 되었고 두들기는 타자기 시대도 있었고 필기구류가 필요하지 않은 반도체 기기의 가상공간에서는 실시간으로 실행되고 출력되고 있어서 글자의 입력방식과 입력속도 그리고 출력속도 전달매체가 오늘날의 반도체 문화발전 속도에 매우 중요한 것이 될 수밖에 없다.

한국인이 한글에 맞는 프로그램개발 언어가 없고, 프로그램을 개발하기 위해 알파벳글자방식 프로그램개발 언어만을 배우는 것은 '한국과학의 발달을 저해하는 요인'에 해당한다.

초등학생 수준에서도 프로그램 소프트웨어를 개발할 수 있는 한글의 글자로 프로그램을 개발할 수 있는 언어와 개발운영체제가 장치되어 있다면 첨단반도체 대한민국의 앞날의 튼튼한 기초가 될 것이다.

그리하여야 만 대한민국의 미래가 밝아질 것이다.

황의 법칙에 의하여 반도체 메모리칩의 용량은 6개월마다 배로 증가하고 그 크기는 반으로 줄어드는 발전을 한다는 것이 2002년도의 일로서 삼성전자는 2012.05.17 16:58 세계 최초로 20나노급(1나노: 10억분의 1미터) LPDDR2(Low Power Double Data Rate 2) 4기가비트(Gb) 모바일 D램을 공급하며 4기가비트 메모리 시장 확대에 나선다. 라고 보도자료 통신사 뉴스와이어(www.newswire.co.kr) 배포와 같이 오늘날은 초박형에 고성능 모바일 기기들이 등장하며 중요한 것은 얼마만큼 입력속도가 빨라지는가 하는 것이기 때문에 글자가 반도체적성에 최적화되어야한다.

일반적으로 컴퓨터는 데이터를 8개의 비트 단위로 묶어 한 번에 처리한다. 비트는 2진법의 0 과 1 가운데 하나를 나타내는 단위이다. 즉, 1비트는 '0' 이 될 수도 있고, '1' 이 될 수도 있다. 비트 8개를 모아 놓은 것을 바이트(byte)라고 부른다. 그러므로 1바이트로 표시할 수 있는 최대 글자의 수는 256조합이 된다.따라서 컴퓨터에서는 8비트씩을 묶어 처리하는 것이 가장 효율적이다. 예컨대 7개 비트 이하로 묶을 경우에는 표현 가능한 수가 128이 된다. 그러나 이 숫자로는 세계 여러 나라에서 사용하는 모든 숫자 · 국가언어 기호 등을 수용하는 것은 절대적으로 불가능하다.

2진법의 체계에서 반도체기기에 사용하는 수의 단위자리는 8계단이며 이때 배열할 수 있는 문자열은 256개가 된다.

$2^{\square} = 256$

따라서 256개의 글자 값을 가지고는 세계 각국의 글자를 배당시킨다는 것도 불가능하므로 세계 각국의 글자를 수용할 수 있는 새로운 방법을 찾아야 하는 것이 필요했다.

반도체기기에는 2진법의 수 체계이어야 하는 특수성으로 다른 진법의 수 체계를 사용할 수 없어 2진법으로 환산의 호환성이 있는 8진법, 16진법, 32진법, 64진법으로 확장하는 것은 가능했다.

이렇게 8,16,32,64로 확장하는 이유는 8계단의 자리 수에 있다.

8진법에서 0,1,2,3,4,5,6,7의 자리수를 넘어서면 더 이상 확장이 불가능하기 때문에 다른 문자를 사용하여 확장시킬 수가 있다는 것이다.

유니코드에서 사용하는 수의 자리는 8진법을 넘었기 때문에

즉, 7777 7777의 8자리수를 넘게 되면 '&#x(Y)YYYX'로 표시하며

한글 '가'는 유니코드 문자표에서 'U+AC000'열에 있다.

이러한 방법으로 현재 64진법으로 까지 발전하고 있다.

진법을 확장하여 표현 가능한 모든 숫자 · 글자 · 특수글자를 수용할 수 있도록 표현 가능한 것을 하나씩 마다 글자 값을 정해 사용할 수 있도록 한 것이 곧 기계언어코드이다.

아스키코드만으로 지구촌 다국어를 수용할 수 없으며 확장된 엡시딕코드로도 다국어를 사용할 수 없어 현재는 유니코드를 사용하고 있다.

유니코드에서는 전 세계의 문자를 모두 표현하기 위한 16진법으로 확장된 코드 체계를 사용하게 된다.

$2^{1}$ □ = 65,536

65,536개의 글자 값을 배당할 수 있는 능력이 발생한다.

참고로 32진법 64진법으로 확장하면 글자 값은 다음과 같이 된다.

$2^{3}$ $^{2}$ =4,294,967,296개 글자 값

2□$^{4}$ =115,281,504,606,846,976개 글자 값

수의 진법체계에서 10진법은 우리 인간이 일상생활에서 사용되고 있는 기본적이 수치이지만 기계어로 사용하기에는 부적합하여 2진법의 가장 단순하고 가장 논리적인 수치로 논리적인 기계장비를 생산할 때 공학적으로 접목 할 수 있어 설계가 원활한 수치이다.

이러한 2진법을 사용할 경우 단위수가 커질 수밖에 없어 상대적으로 단위수를 줄여 언제라도 2진수 치환이 원활한 8진수와 16진수 32진수 64진수를 사용하는 방법이 필요하게 된다.

그러나 16진수의 단점은 단 단위 수치문자인 인간이 사용하는 1~9이외의 문자를 수치문자로 표현되어야 하는 부담이 발생합니다.

이러한 수치 표현 원리를 살펴보면

1, 진법체계

가 10진법체계의 기본

① 10개로 구성된 숫자(0,1,2,3,4,5,6,7,8,9)의 조합으로 모든 수치를 처리

② 소수점 기준으로 10의 누승단위로 자리올림, 마이너스 누승단위로 자리내림

나 2진법의 기본체계

① 2개로 구성된 숫자(0,1)의 조합으로 모든 수치를 처리

② 소수점 기준으로 2의 누승단위로 자리 올림, 마이너스 누승단위로 자리내림

다 8진법의 기본체계

① 8개로 구성된 숫자(0,1,2,3,4,5,6,7,)의 조합으로 모든 수치를 처리

② 소수점 기준으로 8의 누승단위로 자리 올림, 마이너스 누승단위로 자리내림

라 16진법의 기본체계

① 16개로 구성된 숫자(0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)의 조합으로 모든 수치를 처리

② 소수점 기준으로 16의 누승단위로 자리 올림, 마이너스 누승단위로 자리내림

2 진법의 진행방법

가 10진수를 2진수, 8진수, 16진수로 변환

▶정수부 ; 해당진수로 소인수 분해하여 나머지 값을 역순으로 나열

▶소수부 ; 해당진수로 곱하여 정수부 단 단위 값을 정수로 나열

(예) 10진수 (74.375)10를 2진수, 8진수,16진수로 변환

| 10진수 → 2진수 | 10진수 → 8진수 | 10진수 → 16진수 |
|---|---|---|
| | | |

한글을 배우게 되면 자음과 모음 그리고 조립되는 기본글자의 개수에 따라 크기를 수평수직으로 설계하는 글자습관을 터득해야 한다.

그래야 조립된 글자의 모양과 균형이 이루어진다.

이러한 글자사용의 습관과 생활화는 무의식적인 생활에서 자동적으로 수직과 수평 공간의 짜임새를 늘 설계하게 된다.

한글을 사용하는 한국인의 뇌는 늘 조립구조의 전체를 생각하게 된다.

그러하기 때문에 한국인의 사고는 '나' 보다는 '우리라는 조립된 무리들' 의 사고를 하게 된다.

형제가 없는 '독자' 의 경우도 자기를 낳아준 엄마를 '나의 엄마' 라고 하지 않고 '우리엄마' 라고 한다.

나보다는 '우리' 라는 낱말을 즐겨 사용하게 된다.

한국인은 개인보다 우리라는 조직체를 중요하게 생각한다.

오랜 생활 대가족제도와 부락공동체에서 길들여진 우리라는 것이 늘 생활화 되어 있다.
한국인에게 '우리라는 힘' 은 한국의 역사 속에서 큰 힘으로 작용해 왔다는 것을 쉽게 찾아 볼 수 있다.

지난 88올림픽 때 세계는 한국의 저력을 보고 놀랐다.
외국으로부터 찾아오는 손님에게 보이기 위해 거리며 질서이며 자원봉사는 세계인이 아닌 한국인인 우리들도 놀랐든 사실이다.

1997년 11월 21일 정부가 IMF에 구제 금융을 공식 신청 발표하고 2001년 8월 23일 IMF 관리 체제에서 졸업(IMF 구제금융 195억 달러 전액 상환)하는 3년 9개월 동안 경제적으로는 아주 어려웠지만 전 국민이 금모으기 운동 등으로 일치·단결 하여 과거의 무사안일주의가 타파되었다는 점에서 긍정적인 평가를 받기도 한 때도 있다.

더욱이 놀라운 사실은 1950년 6월 25일 전란이 발생하고 1953년 7월 27일 휴전협정 체결이후 UN에서 한국전쟁복구를 위해 많은 원조가 있었다. 전쟁복구를 위해 무엇보다 먼저 지원한 것이 정신과의사 파견이었다고 한다.
동서고금을 막론하고 전후에는 미망인과 부랑고아로 정신질환자가 넘쳐나기 때문에 이를 위한 정신과 의사파견이 필요한 것으로 판단했기 때문이다.
그러나 정신과 의사들이 한국에 파견되어 맨 처음 보고 느낀 것이 한국의 전쟁미망인이 정신과 질환을 앓고 있는 줄로 짐작한 것은 기우에 불과했다는 것이었다.
한국의 길거리에는 미망인들이 아이를 들쳐 업고 길거리에서 생필품을 늘이어 좌판을 펴고 호객을 하고 있었고 우는 아이에게 젖을 물리고는 살아남기 위해 억척을 보인 것이다.
부끄러움은 사라진지 오래고 어떻게 하든 가족과 가정을 꾸리고 있었고 길거리에서 떼묻은 돈으로 자식을 공부시키고 있었다.
필자도 이 시기에 태어났다.
정확하게 말해 1950년 6월 25일 전쟁이 일어나기 38일 전인 5월 18일 생으로 학교생활에서 아버지를 잃은 유가족이라는 친구들이 많았다.

즉, 미망인의 자식이라는 것으로 모두가 착실하게 같이 자랐다.
한국인은 이랬다.
나보다는 조립 자소로 조립되어진 글자처럼 구성체인 '우리를' 먼저 생각했다.
지구상에 한국처럼 많은 위기를 겪은 국가들도 드물 것이다.
그러하면서도 한국이라는 문화를 유지하면서 이어왔고 그것도 현 세기에서는 문화를 이끌어가는 선진대열에서 중추적 역할을 하고 있다.

## 12 중국 50,000 한글 26,000이 합한 76,000자소의 해결방법

산 아래서 혹은 산 속에서 전체의 큰 산을 말하라고 하면 산이 어떠한지를 설명할 수가 없다.

그러나 산 정상에 올라서서 사방을 둘러보고 나서면 그 때야 그 산이 동쪽에는 어떠하고 서쪽은 어떠하며 계곡과 산등성이가 무엇으로 조성되고 채광은 어떠하며 바람과 물의 흐름을 설명할 수가 있다.

이와 같이 2진법체계도 한 나라 글자 값이 50,000을 넘고 30,000을 넘으면 128개의 글자 값으로 시작한 초기 컴퓨터에서 충돌 없이 사용한다는 것은 사실상 불가능한 것이나 다름없고 근본적인 해결방법은 그 진법을 확장하는 방법뿐이다.

반도체기기 프로그램개발의 초기 8진법체제에서 중국, 일본, 한국(CJK, Chinese-Japanese-Korean)에서 사용하는 글자 집합인 한글, 한자, 가나, 병음, 주음부호 등은 그 개수가 많아 ASCII 코드로는 이를 처리한다는 것은 불가능한 것으로 CJK 글자를 처리하기 위한 별도의 방안이 요구되었다.

원초적으로 글자 값을 많이 할당해야 하는 CJK글자 중 중국 글은 조립자소이나 부수가 277개나 되어 전환자소로서만 입력이 가능하고 한글은 글자를 표현하기 위해서는 가장 먼저 '글자 집합'을 정의해야 한다. 글자 집합은 표현해야 할 글자를 정하고 순서를 지정한 것이다. 영어라면 'A', 'B', 'C'에서 'Z'까지(소문자 a에서 z), 한글이라면 '가', '각', '간'에서 '힣'까지다. 물론 숫자나 특수 문자뿐만 아니라 인쇄와 통신을 제어하기 위한 제어 문자도 문자 집합에 포함되어야 한다. 이러한 글자 집합을 코드 형태(일반적으로 행렬)로 표기한 것을 코드화된 문자 집합(CCS, coded character set)이라고 한다. 예를 들어 '가'에는 10001, '각'에는 10002와 같이 코드를 할당하는 방식 말이다. 그리고 문자 집합을 컴퓨터에 저장하기 위해서 옥텟(octet, 8비트 단위) 형태로 표현한 것을 인코딩 방식(CES, character encoding scheme)이라고 한다.

2013년 현재의 확장된 방법으로는 확보된 글자 값이 상상을 초월할 정도로 되어 있다

2진법으로 쉽게 사용할 수 있는 7 또는 8비트 8계단 이하의 수 값은

$2^{\square}$ = 128개

$2^{\square}$ = 256개

$2^{1\ \square}$ = 65,536개

$2^{3^{2}}$ = 4,294,967,296개

$2\square^{4}$ =115,281,504,606,846,976개

의 글자 값으로 확장하고 나면 지구상의 글자와 부호가 100.000개가 체 되지 아니하므로 16진법체계에서부터 쉽게 해결 된다는 답이 보이게 된다.

따라서 세계다국어는 100,000개 글자 값들이 하나하나 구별이 지어짐으로 충돌이 일어날리 없고 같은 사용 환경으로 단장시키면 호환성에는 문제가 발생할 수 없는 동질성의 경로를 찾게 된다.

큰 통속에서는 소스가 많아도 또 어떤 소스이건 담아서 편하게 사용할 수 있듯이 8피트 8진법으로 확장한다고 해야 고작 256개의 글자 값이 되어 알파벳 권에서 갖은 방법을 동원하여 운용방법을 찾았으나 결국은 16진법에서 글자 값 표현방법을 다르게 하거나 32진법, 64진법으로 확장하는 방법을 개발 하게 되고 그런 후에야 비로소
100,000개에 가까운 글자 값이 제대로 처리되고 해결 되었다.

글자들 중 낱개 글자로 취급하고 전환방법으로 사용 환경을 바꾸게 되면 모든 충돌이 사라지기 때문이다.

이러한 혜택을 크게 받은 글자는 중국으로서 중국의 50,000의 글자는 알파벳자소에 의한 병음에서 전환방법이 개발되어 처리 되고 있다.

1950년대 이후 중국도 50,000개나 되는 자국의 글자의 사용이 불편함을 해소시키기 위해 국가정책으로 많은 노력을 하게 된 것을 아래를 보면 알 수가 있다.,

그리고 중국정부는 번자체를 간화자하는 정책으로

① 17획 이상인 글자는 반듯이 그 획을 줄이며

② 이미 통용되는 간편한 글자는 그대로 사용하며

③ 서법의 규칙과 특징에 맞게 간소화 한다는 것이다.

어찌하든 중국의 글자는 사용하기에는 나아졌다고는 하나 중국의 전통성을 사라졌고 정보속도전에도 문제가 있다.

중국은 조립자소를 사용하는 나라이나 반도체적성글자로는 낱개자소취급을 받는 글자이다.

이처럼 많은 글자를 가진 경우라도 글자를 수용할 수 있는 진법체계를 확장하고 전환의 방법으로 사용하면 무리 없이 사용할 수는 있다.

그러나 한글은 전환방법인 아닌 바로 자소로 다루어야 하기 때문에 지난 수 십 년간 프로그래머들에게는 어렵고 많은 숙제를 풀게 한 글자이었든 것이다.

진법체계사용기술이 2진법에서 16진법인 65,536개의 글자 값에서

65,536개 글자 값 표현방법을 바꾸거나

32진법($2^{32}$) = 4,294,967,296개 글자 값

64진법($2^{64}$) = 115,281,504,606,846,976개 글자 값까지 확장할 수 있는 기술이 개발되어서 현대한글 11,223개 처리는 쉬워졌다.

또 컴퓨터의 진법체계에서는 글자 값에 대해 음절문자, 첫-가-끝, n바이트, 조합형, 완성형, 조합완성형이니 하는 방법으로 굳이 다룰 필요성도 없어졌고 사실상 의미도 없는 것이나 다를 바 없다.

개발 당시 큰 글자 통이 없어 작은 글자 통에서 많은 글자 값을 소화시키려고 하다 보니 여러 방법이 동원된 것은 인정해야 하고 충돌이 자주 일어나는 많은 부작용은 따랐다.

반도체에 있어서 글자 값은 어떤 형태의 글자이든 입력 경로에서는 낱개자소와 조립자소 2가지로만 나누어지고 있을 뿐이고 입력경로 그 이후부터는 모두가 같은 하나하나 독립된 글자 값을 가지고 다루어지게 됨으로 문제가 발생할 소지가 없어진다.

컴퓨터의 진법체계에서 글자 값의 이해는 마치 산 아래 혹은 산속에서의 경로로 생각하지 말고 산 정상에서 생각 하자는 것으로 “어느 나라의 글자이든지 글자 하나하나 독립된 글자 값으로 동등한 사용 환경을 만들면 웹상에서 충돌이 사라진다는 것이다.

같은 글자를 사용하는 중국이지만 대만은 번자체에 주음부호를 사용하고 공화국에서는 병음에 간자화 하여 사용한다.

중국어 주음부호에는 자음 스물한 개와 모음 열여섯 개로 되어 있다

결과적으로 컴퓨터상에서 입력경로를 병음 처리하는 것은 사실상 중국글자는 사라져 없어진 것이나 마찬가지이다.

반도체기기인 상용컴퓨터에서 조립자소는 대단이 중요하다.

그러나 자소수가 많아 입력자판기로 사용할 수 없을 때에는 낱개 자소로 취급되어야 한다.

중국 글 50,000개 글자 값과 한글 26,000개 글자 값은 같은 전형적인 조립자소의 글이지만 조립자소의 수가 많다는 이유로 중국 글은 낱개 자소로 전락하여 알파벳자소를 빌려다 써야하는 주체성을 상실한 글자가 되었다는 것이다.

반도체기기 사용방법이 세계의 글자 값을 정상적으로 운영되는 체계 하에서는 한글이 입력경로이든 실행경로이든 웹상에서 가장 이상적인 자소이어서 전환방법으로 사용하는 글자들은 모두 한글자소방법을 택하게 되어 있다.

따라서 문화의 주도권이 한글로 넘어오는 시대가 임박했다는 결론이 나온다.

## 13 프로그래머들에 의한 한글코드의 발전사

앞서 한글이 반도체적성최적글자라는 것은 이미 이야기를 했다.

지금부터는 컴퓨터를 위한 한글이 실제 겪어온 발전사를 알아보자.

반도체적성최적글자 한글 발전사를 이해하면 미래사를 그려볼 수 있게 되어 미래반도체산업의 발전 방향을 찾게 될 것이다.

지금은 유니코드에서 중국의 50,000개 글자 값을 따로 보조다국어 평면(SMP)에 담아두었고 옛한글을 포함한 한글 26,000개 글자 값은 기본다국어 평면(BMP)에 담아두었다.

이렇게 본다면 세계 각국 글자와 기호 문장부호 등을 합한 100,000개가 글자 값 대상이 된다고 할 수 있고 16진법으로까지 확장한다고 하여도 65,536개의 글자 값밖에 확보되지 않아 다른 방법이 찾지 못하는 한 웹상에서 충돌은 당연한 것으로 봐야 한다.

현재 유니코드에서는16진법의 글자 값 표현 방법은 5개 분야로 달리하여 사용하는 것으로 즉,

평면다국어 BMP, 65,536개 글자 값

보조다국어 SMP, 65,536개 글자 값

보조상형글자 SIP, 65,536개 글자 값

3차 상형 문자 평면 TIP, 65,536개 글자 값

보조특수목적평면 SSP 65,536개 글자 값

합계 327,680개 글자 값

을 가질 수 있도록 확보하여 두고 있다.

여기에 32진법 64진법으로 확장하면 실로 글자 값은 상상을 초월하게 된다.

이쯤 설명하면 무슨 말인가를 충분하게 이해가 될 것이다.

산을 이해하는 방법으로 산 아래 혹은 산중에서는 절대 불가능하고 산 정상에서 사방을 둘러 보고나서야 산을 설명할 수가 있다.

충분하게 큰 규모의 글자 값 통이 마련되면 그 통 안에 많은 글자 값의 집어넣었을 수 있고 이때에는 충돌 없이 원만한 사용관리가 이루어진다고 이해가 될 수 있다.

때문에 조립자소에 의한 한글11,172개의 조립글자는 큰 통의 틀로서 다시 정리해야할 필요성 있으며 32진법 64진법으로 처리할 수 있는 기술이 확보된 지금의 수준에서야 비로소 “조립자소이며 바로자소인 반도체최적글자소로서 웹상에서 충돌의 사고 없이 제대로 사용가능한 방법을 찾을 수 있다는 것은 물론 반도체글자로서는 최적화된 한글 사용에 대한 운용체제가 새롭게 개발되어야 한다는 결론도 나온다.

16진법으로 글자 값을 확보하기 전까지 한글은 글자가 아닌 그림으로 취급하여 실행하여온 과거를 되새기며 변천사를 살펴보게 되면 왜 한글이 여러 우여곡절을 겪어야만 하였는가를 빠르게 이해하게 되고 반도체한글 변천사가 곧 반도체세계변천사라는 것으로 착각하게 된다.

아래의 반도체한글 변천사에 대한 설명은 사이트에서 발췌한 것을 적어본 것이다.

초기 컴퓨터 프로그래머들은 아스키코드로 글자 값 배당을 7비트 혹은 8비트를 사용하면서

2□ = 128개 글자 값,

2□ = 256개 글자 값이 생성되고,

2진법의 8개 단위(1111 1111)수로 사용하기 위한 글자 값이 256개 생성되면 알파벳의 128개의 아스키코드 글자 값 배당에서 반가량이 차지해 버리고 나머지 반인 128개만이 여유 값으로 세계다국어를 수용해야 하는 처지가 된다.

여기에 동양권(CJK)의 글자 값 배당이 턱 없이 부족하다는 것을 알 수 있게 되고 따

라서 궁여지책으로 여러 방법을 동원하여 시행하였으나 그때마다 충돌로 일관했다.

이러한 방법은 결국 궁여지책일 뿐 온전 할리는 없었기 때문인 것으로 보아야 한다.

쉽게 말해 256개 글자 값 그릇에 한글이 11,172개가 담아질리 없고 중국글자 50,000개가 담아질리 없다.

설사 16진법으로 확장한다 해도 2 × $10^1$ □ = 65,536개 글자 값이 생성되지만 이들 글자 값에 50,000개 중국글자와 26,000개 옛한글을 포함한 76,000개 글자 값을 어떻게 수용처리 할 것인가?

이해가 가리라 생각한다.

컴퓨터에서 한글코드는 취급과 변천사는 대체로 다음과 같이 여러 방식들이 제안 되는 가운데 변천하였다.

1). n 바이트 한글코드

2). 3 바이트 한글코드

3). 상용 조합형 한글코드

4). KS 완성형 한글코드

5). 유니코드의 한글코드

1). n 바이트 한글코드

n 바이트 한글코드 는 운영체제 상에서 한글 지원이 없던 시기에 한글 입출력이 가능하도록 지원하기 위한 방법으로 단순히 자음과 모음을 아스키 문자에 대응시킨 것이다.

ex) n 바이트 한글 코드 표 -- 풀 코드표는 책 또는 인터넷을 참조하시길...

ㄱ - A

ㄲ - B

ㄴ - D

아래 글은 컴퓨터를 위한 한글프로그램개발자의 글이다.

## 한글 이 컴을 만났을 때| 자유 게시판

후니 | 조회 9 |추천 0 | 2001.11.03. 06:26

이 글은 도스 프로그래밍을 한창 배울 때 써본 것입니다. 지금 같은 윈도우즈 환경에서는 한글문제가 OS 자체로 거의 해결된 상태이지만 윈도우즈는 완성형 한글을 사용하기 때문에 완전한 한글을 표기하려면 아무래도 조합형 한글이 필요합니다. 아무튼 컴퓨터에서의 한글 표기방식을 간략하게나마 구조체와 공용체와 관련지어 써본 것입니다.〈컴퓨터와 한글의 만남〉 -구조체, 공용체, 비트필드 요즘 잘나간다 싶은 프로그램에서는 거의 한글이 지원된다. 원래 컴퓨터란 게 물 건너 온 물건이라서 영어는 잘하는데 한글과는 사이가 꽤 있었다. 그러던 것이 몇몇 분들의 노력에 의해 컴과 한글이 만나게 되었던 거다. 이글에서는 컴과 한글의 궁합에 대해 잡다하게 써 보기로 한다. 애플의 매킨토시에는 그래픽 모드만 존재하는데 비해, IBM 계열의 컴에는 텍스트 모드와 그래픽 모드의 두 가지 모드가 있다. 텍스트 모드란 그냥 텍스트만 출력하는 상태이고, 그래픽 모드란 그래픽을 출력할 수 있는 상태이다. 그러니까 그래픽 모드는 말 그대로 점단위로 모든 걸 그려주는 거다. 위에서도 말했다시피 컴이 물 건너 온 것이라서 텍스트 모드에서는 한글을 출력할 수가 없다. 그러므로 그래픽 모드에서 한글을 그려 주어야 한다. 한글의 코드방식에는 과거 여러 방식이 있었으나 현재는 2바이트 완성형과 2바이트 조합형이 국내 표준으로 자리 잡고 있다. 완성형이란 이미 완성되어 있는 한글을 출력하는 방식이고, 조합형이란 초성, 중성, 종성으로 쓰일 한글 자모들을 몇 벌씩 마련해 두고 이를 알맞게 조합하여 한글을 출력하는 방식이다. [컴퓨터] 어떤 기종용(機種用)의 프로그램을 특수한 기법이나 기구를 써서 딴 기종에게 해독 · 실행시키는 장치 또는 프로그램한글을 처리하는 방식으로 나누자면 크게 에뮬레이터 방식과 내장한글 방식으로 나눌 수 있는데 애뮬레이터 방식은 nkp 나 한글 도깨비 같은 프로그램을 램에 상주시켜 사용자의 키보드입력을 가로채어 한글인지 아닌지 판별하여 화면상에 문자를 출력한다. 물론 비디오 모드이지만 겉으로 보기에는 텍스트 모드처럼 보이는데 이것은 텍스트 모드를 흉내 낸 것이다. IBM PC에는 1초에 18.2번의 주기로 타임 인터럽트(time interrupt, INT 08H)라는 루틴이 실행되는데 에뮬레이터 방식에서는 바로 이 타임 인터럽트가 실행될 때마다 비디오 램을 검사하여 내용이 바뀌어 있으면 그 바뀐 내용대로 화면에 그래픽으로 출력하는 것이다. 입력 때도 이와 비슷하게 키보드가 눌릴 때마다 키보드 인터럽트가 실행되어 한글인지 아닌지 판별하게 되는 것이다. 그렇게 한글을 출력하기 위해서 애뮬레이터도 항상 램에 상주되어 있어야 하지만, 폰트 문제도 만만치 않다. 한글 도깨비처럼 아예 프로그램에 포함되어 같이 램에 상주될 수도 있고, 태백한글처럼 폰트화일을 따로 둘 수도 있다. 하지만 결국 빠른 입출력을 위해서 사용하는 폰트 자체는 어떠하든 램에 상주되어 있어야 하므로 램도 많이 차지하게 되고 속도도 느려지게 된다. 그 해결책으로 나온 것이 한글 카드이다. 이것은 한글 폰트를 롬에다 저장하여 하드웨어적으로 한글을 처리하게 한 것인데 속도는 조금 더 빠르지만 이 또한 애뮬레이터가 램에 떠있어야 하는 것은 마찬가지이다. 또 폰트 또한 롬에서 직접 읽어올 수도 있지만 빠른 입출력을 원한다면 램에다 올려놓아야 한다. 이것은 주로 기본메모리가 아닌 EXPANDED MEMORY(EMS)를 이용하는 수가 많다. expander 증량제, 확장기 □컴퓨터□ 기억장치, 메모리내장한글 방식은 한글 라이브러리를 사용하여 프로그램 내부에 한글출력 루틴을 내장시키는 것이다. 애뮬레이터 방식이 한글 상태를 제어하기 어렵고, 인터럽트를 항상 확인해야 하기 때문에 속도가 느린데 비하여, 내장한글 방식은 라이브러리

이므로 속도에 영향을 주지도 않고, 제어하기도 편하며, 여러 가지 효과도 줄 수 있어서 여러모로 편리하다. 그러나 한글이 전혀 지원되지 않는 프로그램에서나 내장한글 프로그램 소스에 한글을 입력할 때는 애뮬레이터 한글을 사용할 수밖에 없다. 우리나라에서 쓰이고 있는 한글 라이브러리로는 허르미 한글 라이브러리, 한라 프로, 하나리 C 라이브러리, 한글 라이브러리 〈한〉 등이 있는데, '허르미'는 다양한 출력 방법을 보여주기 위해 처음부터 공개로 만들어진 라이브이고, '한라프로'는 SVGA및 마우스까지 지원되고, 함수 종류가 다양하며, '한'은 소스가 공개되어 있어 참조하거나 고치기 편하다. 그 외 '미니 한글 라이브'라는 것을 본적이 있는데, 이것은 2벌식 한글 오토마타는 물론 폰트, 상하좌우 스크롤, 한글 입출력 함수 등이 전부 C소스로 제공되어 간단히 한글 입출력만 하는 프로그램에서 작고 빠르게 한글을 지원할 수 있게 되어 있었다. 이렇듯 시중에 여러 가지 한글 라이브가 있으니 자기가 계획한 프로그램에 알맞은 한글 라이브를 입맛대로 골라 쓰면 되겠다. 한글에서 표현가능한 문자의 개수는 모두 11172 가지나 되기 때문에 그 많은 수를 1바이트로는 감당할 수가 없고, 2바이트 이상이 되어야만 한다. 그래서 표준이 정해지기 전까지는 2바이트 한글, 3바이트 한글, n바이트 한글, 7비트 한글(청계천 한글) 등이 있었는데 지금 표준은 2바이트 한글 체계로 정해졌다. 처음에 표준으로 정해진 완성형은 표현 가능한 11,172 가지의 한글 가운데 자주 이용되지 않는 글자는 빼버리고 사용 빈도수 별로 2350 자의 한글을 선정하여 통신시에 충돌하지 않는 코드 영역에 가나다순으로 넣은 것이다. 2바이트에 넣을 수 있는 문자는 $2^{1}$ □ (=65536) 가지이므로 2350 자의 한글이외에도 256 자의 영문자, 4888 자의 한자, 1128 자의 특수문자를 제공한다. 오래 동안의 논의 끝에 보조 표준으로 정해진 조합형은 초성 19개, 중성 21개, 종성 27개가 조합되어 하나의 문자를 이룬다. 조합형에서 한글을 어떻게 구현하는지를 알기 위해서 C 프로그램 상에서 어떻게 조합형 한글을 구현하게 되는지 간단하게나마 살펴보기로 한다. 그러기 위해서는 우선 구조체, 공용체, 비트필드 등에 대해서 알아야만 한다. 구조체와 공용체는 비슷한 면이 많다. 큰 방이 하나 있다고 생각해 보자. 그 방을 A, B, C 라는 세 사람이 사용하는 데에는 두 가지 방법이 있다. 세 사람이 각자 그 방에다 칸막이로 삼등분을 해서 서로 관계없이 그 방을 사용하는 경우가 있고 또 세 사람이 각자 그 방을 사용하는 시간이 다른 경우, 각자 필요한 만큼 칸막이로 공간을 만들어 쓰는 경우가 있을 것이다. 전자의 경우가 구조체, 후자의 경우가 공용체에 대한 개념 설명이다. 이제 좀 더 구체적으로 설명을 해 보기로 한다. 〈구조체〉〈공용체〉 struct score { union number { char name[30]; double dd; int kor; float ff; int eng; int ii; int math; char cc; } person; } num; 위 왼쪽은 구조체, 오른쪽은 공용체의 예를 든 것이다. 내용 뿐 아니라 형식도 비슷하다. 구조체는 몇몇 개의 관련 있는 변수들을 한데 모아 놓은 것이다. 이렇게 해 놓으면 나중에 변수를 참조할 때 편리하게 사용할 수 있다. 예를 들어 학교에서 각 학생들의 성적을 개인별로 입력한다고 생각해 보자. 과목도 여러 가지이고 학생도 여러 명이라, 과목과 학생이름을 저장하는 변수를 따로따로 쓴다면 상당히 혼란스러울 것이다. 이럴 경우 여러 개의 디렉토리가 각기 거기에 관련되는 여러 개의 서브디렉토리를 가지고 있을 경우, 그 구조를 정리 및 이해하기가 쉬운 것처럼, 구조체를 사용하면 각 학생들의 이름과 점수가 혼동되지 않고 쉽게 연결되어 정리될 것이다. 위 구초체의 예에서 국어(kor)점수를 나타내는 변수는 person.kor 이고, 이런 식으로 구조체 변수(person)와 그 구조체 속의 한 변수(kor) 사이에 점(.)을 찍어 나타낸다. 공용체는 그 공용체 속에서 메모리를 가장 크게 차지하는 변수를 기준으로 메모리를 할당받아서 그 공간을 각 구성요소 (공용체 속의 변수들) 들이 자기들의 데이터형으로

사용한다. 즉 같은 메모리를 서로 다른 데이터형으로 사용한다는 뜻이다. 그 기본개념 외의 나머지는 구조체와 같다. 비트필드란 방금 설명한 구조체와 공용체의 조합에서 약간의 내용이 추가된 것뿐이다. 비트필드란 그 이름처럼 각 데이터를 비트단위로 다루는 것이 가능하다. 위에서 한글 표준으로 2바이트 완성형과 조합형이 있다고 한 적이 있는데, 한글을 표시하기 위해서는 초성, 중성, 종성을 표시하는 부분과 한글인지 아닌지 표시하는 부분이 필요하다. 그러기 위해서는 2바이트(=16비트)를 비트단위로 가를 수밖에 없다. 그래서 비트필드가 한글 입출력에 유용하게 사용되는 것이다. +-----------------------------------------------------------------+ | | 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | | --++--+--+--+--+--+--+--++--+--+--+--+--+--+--+--++-- | | 메모리 || | | | | | | || | | | | | | | || | | --++--+--+--+--+--+--+--+--++--+--+--+--+--+--+--+--++-- | | | | | | | | | | | | +------+-----+ +------+------+ +----+-------+ | | | | | +----- 종성 (5비트) | | | | +---------------- 중성 (5비트) | | | +---------------------------- 초성 (5비트) | | | | | +--- flag (flag=1 이면 한글, 0 이면 영어로 판단) | | | +-----------------------------------------------------------------+ 〈조합형 한글코드의 체계도〉 한글 한 자를 출력하기 위해서는 2바이트가 필요하고, 그 2바이트를 위의 그림과 같이 비트 단위로 쪼개어 사용하는 것이다. 다음은 조합형 한글 코드를 비트필드로 나타내어 본 것이다. union han { unsigned in; /* 완성된 한글 코드로 인식 */ struct { unsigned last : 5; /* 종성 */ unsigned middle : 5; /* 중성 */ unsigned first : 5; /* 초성 */ unsigned flag : 1; /* flag */ } bit_field; } h; 위와 같이 공용체를 쓴 이유는 한글을 하나의 완성된 코드로 입력할 수도 있고, 분리된 각각의 음소 단위로도 입력할 수 있게 하기 위해서이다. 지금까지의 설명을 참고로 조합형 한글을 출력하는 방법을 알아보자. routine 『컴』 프로그램의 일부분으로, 특정한 기능을 수행할 수 있도록 마련된 일련(一連)의 명령군(命令群). 입력 루틴 · 출력 루틴 · 진단(診斷) 루틴 등이 있음.

한글은 모아쓰기 방식이므로 한글을 생성하기 위해서 '한글 입력 오토마타' 라는 루틴을 이용하여 입력받은 코드와 현재 입력되어 있는 코드로부터 새로운 코드를 생성하게 된다. 그 코드에서 초성, 중성, 종성을 구분해 내어 각각의 자모를 메모리에 저장시켜 그것을 겹쳐 화면에 출력하여 한 글자의 한글을 출력하게 된다. 그래서 조합형 한글은 글자의 표현에 제한이 없다. 다만 조합형은 경우에 따라 하위 바이트의 값이 통신 제어 코드와 충돌한다는 큰 단점이 있어 국내 대부분의 통신 BBS에서는 완성형 한글을 사용하고 있다. 지금까지 살펴본 바와 같이 한글은 컴퓨터와 잘 어울리는 체계를 가지고 있다. 이 정도만 가지고서도 충분히 한글의 우수성을 알 수 있을 것이다.

한글과 영어를 사용하기위해 제어문자 SI(Shift-In), SO(Shift-Out)를 사용하며, SO 문자 이후의 문자들은 한글로 인식하는 방법이다.

ex)

"〈SO〉AsA〈SI〉" 는 한글로 "국"을 나타낸다.

또한 "〈SO〉A〈SI〉 〈SO〉sA〈SI〉"

"〈SO〉A〈SI〉 〈SO〉s〈SI〉 〈SO〉A〈SI〉" 로 나타낼 수 있다.

위의 예처럼 한글로 "국"을 나타내기 위해 다양한 방법이 존재 한다.또한 'ㅘ/ㅟ'등 복모음은 두개의 아스키 문자로 표현되고 종성이 없는 음절도 존재하므로...

결국 한글을 표현함에 있어 가변 길이의 형태로 제공되어 문자열 정렬 또는 탐색에 어려움이 있다.

2). 3 바이트 한글코드

3 바이트 한글코드는 n 바이트 한글 코드에서의 가변적인 길이 문제를 해결하기 위해 제시된 방법이다. 초성, 중성, 종성을 각각 하나의 아스키 문자로 대응 시켰다.

중성을 표현하는데 있어 'ㅘ/ㅝ'등의 문자를 1개의 아스키 문자로 표현하는 것에 있어 n 바이트의 가변 길이의 문제점을 해결하였다.

ex) 상용 조합형 및 3 바이트 한글코드 표 -- 풀 코드표는 책 또는 인터넷을 참조하시길...

2진수 / 초성 / 중성 / 종성
00000 / 채움 / / 채움
00001 / ㄱ A / 채움 / ㄱ A
00002 / ㄲ B / ㅏ b / ㄲ B
00003 / ㄴ D / ㅐ c / ㄳ C
....
....
01111 / ㅉ Y / ㅙ n / ㅥ O
....
....
11111

3). 상용 조합형 한글코드

위의 n 바이트, 3 바이트 한글코드는 한글과 영어 표현에 대하여 같은 바이트가 사용되었다. 이는 한글과 영문자를 구문하는데 있어 표준 코드로써 부적합한 점이 있다.

상용 조합형 한글코드는 한글과 영어문자를 구분하기 위하여 최상위 비트를 이용하여 구별한다. 최상위 비트가 1이면 한글, 0이면 영문자이다. 그리고 초성, 중성, 종성 각각 5 bit씩 코드를 할당하여 초성, 중성, 종성의 조합으로 한글을 표현하도록 하였다.

ex)

'가' : 1 00010 00011 00001 -〉 1000100001100001

참고로 한글은 초성 19자, 중성 21자, 종성 27로써 조합 가능한 음절수는(여기서 종성이 없는 음절도 생각해야 됨) 19x21x(27+1) = 11,172자이다.

4). KS 완성형 한글코드

기존의 방법들(n바이트, 3바이트, 상용 조합형)은 한글에 대해서만 정의하였다. 하지만

한글 문서에는 한글 이외에도 다수의 한자가 등장하기 때문에 한글코드와 한자 코드도 정의되어야 한다. 또한 상용 조합형 한글코드는 국제 표준 협회(ISO)에서 제시하고 있는 코드 체계를 따르지 않기 때문에 표준 이라고는 할 수 없다.(제어문자 0~31 영역과 128~159까지의 영역을 문자 코드로 사용하지 못한다. 또한 최소한 아스키 코드영역과 겹치지 말아야한다.)

KS 완성형 한글코드(KS C5601-1987)는 2바이트 영역에서 0~31, 128~129영역을 제외한 공간에 한글 및 한자 코드를 부여한 코드이다.

한글의 경우 11,172자의 문자 중에 빈도수가 높은 2,350자를 가나다순으로 코드를 부여하였으며, 한자 코드는 4,888자를 한자의 음(한글 발음)에 따라 가나다순으로 코드를 부여하였다. (한자의 경우 음(한글 발음)으로 배치하였기 때문에 같은 한자에 대하여 다른 발음이 존재하는 경우가 있으므로 같은 한자가 두개 나타나는 경우가 있음) 또한 한글의 경우 자음만으로 이루어진 초성, 중성 코드와 기호 등은 한글, 한자 코드 이전의 영역에 배치하였다.

2바이트 영문자, 숫자, 일본어 문자, 기호 등
- 상위 바이트 : 0xA1 ~ 0xAC
- 하위 바이트 : 0xA1 ~ 0xFE

한글 자음
- 초성 영역 : 0xA4A1('ㄱ') ~ 0xA4BE('ㅎ')
- 종성 영역 : 0xA4BF('ㅏ') ~ 0xA4D3('ㅣ')
- 엣 한글 자모 : 0xA4D5 ~ 0xA4FE

한글 2,350
- 상위 바이트 : 0xB0 ~ 0xC8
- 하위 바이트 : 0xA1 ~ 0xFE

한자 4,888
- 상위 바이트 : 0xCA ~ 0xFD
- 하위 바이트 : 0xA1 ~ 0xFE

5). 유니코드의 한글코드

KS 완성형은 아스키코드와 겹치지 않게 구성하였지만, 다른 언어(일본어, 중국어)의 코드 체계와 중복되는 부분이 있다. 국제적으로 이런 겹침을 방지하기 위하여 국제 표준으로 제시한 것이 유니코드이다.

유니코드의 한글코드는 한글 음절 11,172자를 0xAC00('가') ~ 0xD7A3('힣')까지 순서대로 정의하였다. 상용 조합형 한글코드처럼 초성, 중성, 종성을 분리 할 수 있지만 따로 계산식이 필요하다.

14 ; ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ

5 ; ㄲ ㄸ ㅃ ㅆ ㅉ(쌍자음 ㄸ ㅃ ㅉ은 종성으로 사용 않음)
11 ; ㄳ ㄵ ㄶ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅄ 삵 삻 삾 삿
10 ; ㅏ ㅑ ㅓ ㅕ ㅗ ㅛ ㅜ ㅠ ㅡ ㅣ
9 ; ㅐ ㅒ ㅔ ㅖ ㅚ ㅘ ㅝ ㅟ ㅢ
2 ; ㅙ ㅞ
유니코드에서의 초성(19), 중성(21), 종성(27+1) 분리 식은 다음과 같다.

```
unsigned unicode; // 한글 유니코드 음절
unsigned uni_num, cho, jung, jong;

uni_num = unicode - 0xAC00; // 0 ~ 11,172 숫자 값으로 변환
cho = uni_num / (21 x 28); //초성 0 ~ 18
jung = (uni_num % (21 x 28)) / 28; //중성 0 ~ 20
jong = (uni_num % (21 x 28)) % 28; //종성 0 ~ 27
```

이외의 코드 설명은 아래의 주소 참조.
http://blog.naver.com/pin0156/100147496224

14 첨단반도체문자가 세계를 지배한다.
새천년의 시대가 되면서 지구촌의 일상생활은 반도체에 의한 정보로 이루어지게 되었다.
인간의 생활은 정보의 수집과 전달로 이루어지고 있으며 이 정보의 전달매체는 반도체 기기이고 기능이 좋고 나쁨과 빠르고 느림의 정도는 그 사회의 문화수준을 평가는 척도가 될 수 있다.
인간이 개발한 정보수집과 전달기능의 매체 수단 중 가장 발달된 것은 반도체이다.
반도체는 1948년 6월 30일 , 미국 벨 전화 연구소에 수십 명의 기자들이 몰려들면서 빛을 보기 시작하였다.
이날 벨 전화연구소는 조그만 원통에서 다리 세 개가 나와 있는 손톱만한 제품을 "트랜지스터라고 소개하면서 이 자그마한 물체가 앞으로 세상을 바꿀 겁니다."

이렇게 소개된 트랜지스터라는 반도체가 세상에 처음 나온 것은 한해 전인 1947년 11월 17일부터 12월 16일 사이에 트랜지스터를 개발하는 데 성공 하면서 이다. 하지만 반도체 기술의 발달에 앞서 전자 기술은 진공관에 의해 꾸준한 발전을 이룩해 왔다. 진공관은 원주형 유리 속을 진공상태로 만들어 놓고 그 안에 필라멘트를 넣어 전기 신호 증폭 등의 기능을 갖게 한 것 이다. 현재는 최고급 오디오를 제외하고는 쓰이는 곳이 거의 없다.진공관의 시작은 1884년 에디슨이 발견한 '에디슨 효과'를 그 출발점으로 한다. 에디슨은 그가 발명한 백열전구의 필라멘트에 또 하나의 전극을 넣으면 여기에도 전류가 흐른다는 사실을 발견했다. 그러나 이 현상을 응용한 상업성을 예견하지 못하고 그냥 지나쳤다. 그 후 미국의 발명가인 드 포레스트가 무선전신기에 진공관을 설치하는 데 성공했고 이 기술을 AT & T (미국 전신전화회사) 에 팔았다. 에디슨이 휴지통에 버린 기술을 살려낸 것이다. 진공관의 발명은 때마침 개발되기 시작한 무선통신 , 무선방송의 발전을 불러일으켜 라디오 전성시대를 가져왔고 1940년 전후부터는 TV전성시대가 개막됐다. 이를 이용한 세계 최초의 컴퓨터 ' 에니악' (ENIAC)도 개발된다.

진공관은 매우 유용하고 편리한 장치이기는 했지만 여러 가지 불편함을 동시에 가지고 있었다. 크기가 너무 커서 전자제품의 사이즈를 줄이는 데 한계가 있었으며 진공관 속의 필라멘트를 가열하기 위해서는 엄청난 전력이 필요했다. 에니악 컴퓨터는 길이가 30미터에 무게가 30톤 , 소비전력이 1백 40킬로와트 이었다. 여기에 냉각장치까지 덧붙이면 그야말로 웬만한 빌딩보다도 큰 '초대형 기계'가 된 것이다. 진공관은 수명이 짧아 수시로 기계의 고장을 유발한다. 에니악은 툭하면 고장을 일으켜 이를 사용하는 시간보다 고치는 시간이 4배 정도 많았다고 전해진다. 그러나 1948년 반도체의 발명으로 진공관 시대는 종지부를 찍게 된다.초기의 트랜지스터는 진공관의 기능을 대치하는 점접촉 형 수준이었으나 몇 개의 트랜지스터를 한 개의 기판에 모아놓는 집적회로가 개발되면서 컴퓨터를 비롯한 전자제품의 소형화 , 경량화 , 다기능화가 급속히 이루어졌다.오늘날의 반도체 집적 회로는 수억 개의 트랜지스터를 하나의 칩 위에 담고 있어 반도체공학의 절정인 집적회로는 전자회로의 소형화 , 고성능화 , 고정밀화 , 저가격화를 가져다주었고 반도체 기술의 진보를 단적으로 보여주는 나타났으니 이것이 컴퓨터다. 컴퓨터 기술은 크기는 줄면서 성능을 향상시키는 경향으로 발전되어 왔다.

반도체는 흔히 단어 그대로 전기가 반쯤 통하는 물질로 알려져 있지만 실제로는 좀 더 복잡한 내용을 포함하고 있다. 일반적으로 고체 상태의 물질은 도전율 (Electrical Conductives) 에 의해 구분된다. 도전율이라는 것은 쉽게 말해서 '전기의 흐르는

양'이다. 이 도전율에 따라 도체 ,절연체 (부도체) , 반도체를 구분한다. 반도체는 평상시에는 부도체처럼 전류가 통하지 않지만 주위 환경을 변화시키면 도체로 변해 전류가 통한다. 에를 들면 반도체에 열을 가하거나 빛을 쪼이는 등 주위 환경을 변화시키면 반도체가 전류가 잘 통하게 되는 것이다. 반도체는 온도 , 광학적인 효과 , 자기장 및 소량의 불순물 원자에 대해 민감한 반응(도핑, doping)을 보인다. 반도체는 이러한 전기적 특성은 다음과 같다. 첫째, 반도체는 온도에 의해 도전율이 현저하게 변한다. 불순물을 포함하지 않은 순수한 반도체의 온도를 높이면 전류가 통한다.둘째, 반도체에 빛을 쪼이면 전류가 잘 통하며 전기저항이 작아지는데 , 이것이 바로 광전효과다.셋째, 반도체에 미량의 불순물을 첨가하면 반도체는 그 불순물의 종류와 농도에 따라 전류의 흐름을 변화시킨다. 반도체 제조공정에서 불순물을 주입하는 것은 이 때문이다. 넷째, 반도체에 금속 등을 접촉시킴으로써 교류를 직류로 바꾸는 작용을 할 수 있다. 이를 정류현상이라고 한다.마지막으로, 도체나 반도체 같은 물질에 전류를 흘리고 이것과 직각 방향으로 자계를 작용시키면 전류를 일으키는 작용이 생겨난다. 이러한 현상을 홀(Hall)효과라고 한다.

최초의 MOSFET 트랜지스터 (1959 )

반도체는 메모리 반도체와 비메모리 반도체로 나누어지며 메모리 반도체는 기억과 저장의 주 기능이 있으며 메모리 공간을 갖고 있는 반도체로 대표적으로 RAM, ROM, 플레쉬 메모리 등의 소자를 예로 들 수 있다.

비메모리 반도체는 반도체의 물리적 성질에 따른 기타 소자들(LED소자 등이 여기에 속합니다)을 제외하면 대부분 시스템 반도체 이다.

시스템 반도체는 디지털 반도체와 아날로그 반도체로 나누고 디지털 반도체는 우리가 흔히 알고 있는 프로세서인 cpu가 여기에 속한다.

아날로그 반도체는 빛, 소리, 압력, 전기 등의 아날로그 신호를 디지털 신호로 전환 및 관리하는 반도체이다.

우리가 사는 세상의 실제 데이터는 아날로그에 속하므로 전자기기 및 방송통신에서는 아날로그의 실제 세상을 적절히 디지털로 변화하여 다루고 있다. 따라서 디지털 시대에서는 아날로그를 디지털로 바꿔주는 아날로그 반도체와 이렇게 변화된 디지털 데이터를 가공 처리를 하는 전자 기기(기타 프로세서 및 메모리 반도체 사용)를 통하여 전자제품의 동작 방송 및 통신, 정보 및 데이터의 처리 및 분석이 이루어지게 된다.

세계 반도체 시장 규모는 비메모리가 80%를 차지하며 메모리는 20%를 차지한다.

이러한 시장 특성은 비메모리 반도체의 특성에 기인하는데 비메모리는 상대적으로 고부가가치 산업이고 수익률이 높다.(메모리는 시장 문턱이 낮고 가격변동이 커서 가격의 안정성이 낮다.)

때문에 인텔은 애초에 메모리 자체를 포기하고 비메모리에만 올인 했고 한국이 비메모리를 육성하지 못했던 이유는 메모리는 소품종 대량생산 이지만, 비메모리는 다품종 소량생산 이기 때문이다.

비메모리는 종류가 2만 가지가 넘고, 고도의 설계기술을 요구하기 때문에 기술 장벽이 높아서 우리나라가 쉽게 접근하지 못했다.

반도체의 발전 속도는 “인텔의 창업자 고든 무어가 1965년 반도체의 집적도가 1년 6개월마다 2배씩 증가 하지만 가격은 변하지 않는다.”고 하는 무어의 법칙으로 발전하였으나 2002년 한국의 황창규 삼성전자 총괄사장이 세계 3대반도체학회 중의 하나인 ISSCC에서 메모리 신성장론을 발표한 “반도체 집적도는 1년에 2배씩 증가하며 이를 주도하는 것은 모바일기기와 디지털 가전 등 이른바 Non-pc분야가 될 것”이라는 주장에 맞아 가면서 “황의법칙”으로 발전하고 있다.

2000년대 인간의 모든 문화는 정보의 수집과 전달속도에 비례하여 발달하게 된다.
그 정보의 수집과 전달매체는 바로 반도체이며 누가 첨단반도체고지를 유지하느냐에 따라 세계를 정복하고 지배하느냐가 달려있다.
인간의 정보 수집전달의 매체는 언어와 글자를 사용하면서 시작하였고 글자와 언어는 기록하여 전달하는 방법으로 종이와 연필이라는 필기구를 통하여 우편제도로 발전하였고 또 글자는 전신부호로 바꾸어 통신선을 통하여 전보형식으로 발전하기도 하였고, 음성으로는 전화기를 통하여 정보전달이 이루어진다.
반도체가 도입되면서 음성 글 그림이 유선이 아닌 무선으로 우주공간의 통신위성을 경유하여 실시간으로 정보를 수집하고 전달할 수 있는 반도체정보문화를 중심으로 인간의 일상생활이 이루어지고 있다.
과학과 예술이 반도체정보매체를 통하여 생성되고 발전하고 있다는 것이다.
반도체정보전달매체의 구성요소에는 글자이외에 소리 그림의 3대 구성요소로 이루어져 있다.
지구촌에는 인종과 민족 국가에 따라 문화권이 이루어져 있고 언어와 글자를 달리하여 사용하고 있다.
반도체정보전달매체에 사용하는 구성요소 중 소리와 그림은 인종이나 민족 국가문화가 달라도 동일한 것이나 글자는 다르게 되어 있다.
반도체정보전달매체에 사용될 수 있는 글자의 특성은 처리장치와 입력장치에 글자를 인식할 수 있도록 내장(install)되어 있어야 한다.
다시 말해 입력장치의 글자단추(button)와 처리장치의 글자실행이 일치하도록 하여 입력된 정보가 수집정보와 일치하도록 인식되어야 한다는 것이다.
입력장치의 개수는 일반글자단추에 해당하는 수단추 연산단추 부호단추 포함하여 46개의 입력단추가 한정되어 있고 기능단추를 포함하면 90개 이내로 된다.
입력단추를 한글전용으로 할 경우 입력단추 1개를 변환단추를 활용하면 알파벳문자입력단추가 26개의 기능을 14개의 단추수로 대신할 수가 있어 글자단추수를 반으로 줄일 수 있다.
인간이 글자를 입력하기 위해서는 양 손의 10개의 손가락을 효율적으로 활용하는 과학적 구조는 단추의 수를 40개로 줄임에 있다.
지구촌에 인간이 개발하여 사용하는 글자 중에 입력장치의 적성을 가장 만족시키는 글자는 두 종류이다.
하나하나의 낱개 자소로 된 알파벳(alphabet) 52개와 35개의 조립 자소된 한글이다.
어떠한 형태이건 글자 수가 적어야 한다는 것은 반도체적성의 제1요건이다.
이러한 원리로 본다면 반도체글자는 자소가 적은 조립자소이어야 한다는 것이다.
반도체가 본격적으로 사용되기 전 1940년대 대전종말이 다가온 시점에는 알파벳글자를 사용하는 문화권에서 세계를 지배했다.

알파벳은 낱개 자소로서 당시 수준으로 정보전달이 빨랐기 때문이다.
정보는 곧 무기요 황금인 것은 예나 지금이나 앞으로 불변의 진리이다.
반도체이전에 글자생성과 전달이 가장 빠른 방법이 타자기요 전보였다.
전선을 타고 전달되는 전신부호로는 26개 낱개 자소인 알파벳이었고 인쇄글자처럼 글자를 찍어내는 타자기 적성글자 역시 알파벳이었다.
중국의 글자나 한글이 이를 따르지 못하기 때문이다.
이러한 시대가 지나고 반도체시대가 온 것이다.
타자기시대와 전보시대에는 낱개 자소 시대이지만 반도체시대에는 달라 자소가 적고 조립되는 글자가 정보시대의 주역이 된다.
이러한 시대는 단연 한글의 조립자소임을 쉽게 알 수가 있다.

낱개 자소인 경우 LCD 모니터에 문자를 실행하려면 수평실행에는 문제가 없으나 수직실행은 불가능한 것이 흠이다.
조립되어 하나의 글자를 이루고 그 글자들이 하나의 단어를 이루는데 있어서 조립자소는 가로로 실행하거나 세로로 실행하는 데에는 전혀 문제가 없는 것이다.
낱개 자소에 있어서 일본어의 경우 중국어와 병행하여 세로로 사용해온 **습관으로 무리는 없어나 알파벳의 경우는 불가능하다.**
실재로는 일본의 글자를 '준 조립자소'라고 해야 할 것이다.
일본 글자는 같은 글자에 부호를 덧대어 사용하기 때문이다.
이러한 부류에는 알파벳 문화권에서도 많이 나타난다.
알파벳에서도 b, d, q, p는 뒤집고 돌려서 만들어 졌기 때문에 근본은 같다고 보아야 한다.
여기에 위아래에 점이나 선을 그어 사용하고 있어서 준 조립자소라고 할 수가 있다.
한글자소는 조립자소로서 수자 부호 등과 확연하게 구분이 가고 자소자체에도 확연히 구별이 된다는 것이 가장 큰 장점으로 작용한 다.
조립문자에는 수직조립방식과 수평조립방식 그리고 수직수평조립방식이 있다.
최적화된 반도체글자로는 수직수평조립자소이라 할 수 있고 다음이 수직 또는 수평조립자소이며 낱개자소이다.
조립자소와 낱개자소의 차이는 조립자소는 실행공간이 가로이든 세로이든 다 만족시킬 수기 있으나 낱개자소는 가로로는 가능하지만 세로로는 불가능한 흠이 있다.
인체에서 정보전달의 중추는 눈인 것으로 눈은 상하와 좌우로 움직여서 정보를 받아들이게 된다.
따라서 때로는 수직인 세로로 실행할 필요성도 있다.

세계가 문화권별로 자소의 형태와 사회구조를 분류하면 조립자소를 사용하는 수직사회와 낱개자소를 사용하는 수평사회로 구분할 수도 있다.
조립자소를 사용하는 수직사회는 전체를 구심점으로 하는 공동사회 발달하였고 낱개자소를 사용하는 사회는 개인을 중히 하고 개인주의가 발달하였다.
조립자소이든 낱개자소이든 정보를 글자로 나타내기 위해서는 낱개로는 불가능하고 하나 또는 둘 이상의 다수가 합하여야 비로소 정보가 만들어진다.
중국의 글자는 수직수평으로 조립되어진 조립글자라고 한다면 알파벳은 단순하게 나열한 낱개자소라는 것이 된다.
일본글자도 알파벳과 같은 수준의 글자이다.

그러나 인체공학적으로 인간의 손가락은 좌우 각 5개로서 반도체라는 매체에 입력하기 위해서는 단추로 하여야 하고 두 손의 손가락으로 입력시키기 위해서는 글자입력단추와 기능단추를 합하여 그 수는 제한될 수밖에 없다.
수를 입력하는 단추 10개, 문장부로를 입력하는 단추 34개, 기능단추 40개를 합한 84의 기본단추와 글자단추를 합하고 1개의 단추로서 두 가지를 입력하도록 제한한다면 88개의 단추로 입력장치를 구성할 수가 있다.
따라서 반도체정보글자의 입력장치는 글자가 25내외일 때이다.
단추의 수를 줄이면 입력방법이 쉬어지고 입력이 빨라진다.
알파벳의 글자습관은 글자를 대소로 구분해야하는 것 때문에 단추의 수가 1개 이상 늘어나는 불편한 점이 있고 복수와 단수를 구분해야하는 까다로운 언어습관은 'a'의 사용으로 글자입력속도를 더디게 하고 상하좌우로 조립할 수 없어 길게 나열해야하기 때문에 글자가 실행되는 공간이 길어야 하는 결점이 있다.
조립글자에서 기본글자가 많은 한자는 반도체정보글자로서는 글자입력 단추 수에 제약을 받아 부적격이라 할 수가 있다.
PC의 글자입력장치로 개발된 것은 알파벳과 한글이 호환하도록 하여 사용하는 것에는 88개의 단추로 되어 있다.
한글로만 사용하는 입력장치라면 88개의 단추에서 9개 또는 16개의 단추를 줄여도 되어 총 72개의 단추로도 입력이 원활할 수 있게 된다.
지금까지 발달된 반도체글자로서 실행 장치에서 사용되는 수직공간으로 사용하는 글자는 없으며 모두가 수평나열글자로 되어 있다.
실행 장치에서 수직으로 사용할 수 있는 글자로는 한글이외에는 없다고도 할 수가 있다.
알파벳문자로는 수직공간으로 사용할 수는 없다.
한글은 14개의 기본자음소와 16개 파생자음소로 30개 자음소, 10개기본모음소에 파생모음소 11개를 합한 21개 모음소가 되어 총 51개로 된 수직수평조립자소이다.
한글은 초성과 중성 그리고 종성으로 구성하여 초성에 중성을 더하여 수평으로 혹은 수직으로 399개의 글자를 조립할 수 있고, 여기에 종성을 더하여 10,773개의 글자를 조립한다.
이렇게 조립된 글자는 그 차체가 발음부호와 동일한 것이 특징이다.
즉 한글은 소리글자로서 조립되어진 그 자체가 발음부호와 동일한 것이지만 영어권의 알파벳은 발음부호를 따로 사용하여야 한다.
때문에 한글의 어법을 익히면 들은 소리만으로도 정확한 정보입력과 실행공간에서 수집한 정보가 정확하게 해석되는 일치를 하게 된다.
반도체글자로서 한글과 영어를 서로비교 해 보면 한글이 최적반도체글자로 제작되어 있다는 것을 알 수가 있다.
한글에서는 낱말의 발음을 둘 이상으로 발음하지 않으며 발음 부호를 따로 사용하지 않으며 발음의 강약을 사용하지 않은 것으로 되어 있어서 사용하고 이해하기 가 매우 쉽다.
같은 내용을 반도체라는 매체공간을 사용하여 전달하면 알파벳글자는 단순하게 나열하는 수단만으로 되어 있어서 한글보다 공간영역을 50%더 사용하여야 한다는 것도 알 수가 있다.
한글은 낱말을 구성하는 글자가 수직수평으로 조립되어지므로 공간을 매우 효율적으로 적게 사용할 수 있는 글자이기도하다.

한글에는 묵음이 없다.
한글에는 같은 자소가 다르게 발음되어지는 경우가 없는 것이 영어와의 차이이다.
한국인이 영어를 쉽게 배울 수 없는 이유 중의 하나가 우리의 언어습관이 그 들과 다르기 때문이다.
앞서 말한 묵음과 동일한 낱말을 다르게 발음하는 것과 발음에 강약이 들어 있다는 것이다.
한글 글자를 사용하는 언어습관에 없는 이러한 언어습관은 반도체시대발전의 발목을 잡는 저해요소로 작용하고 있다.
인간의 사고능력은 사용하는 글자에 많은 영향을 받게 되어 있다.
사용하는 글자가 낱개자소이냐 조립자소이냐에 따라 다르고 사회의 구조도 사용하는 글자에 따라 수직구조를 이루든가 수평구조를 이룬다.
가로인 수평으로만 사용되는 낱개글자는 개인주의로 발전하였고 세로인 수직으로 사용해온 글자는 수직집단주의로 발전하였다.
사고의 능력도 개인주의와 집단주의에 따라 다르게 된다.
집단주의는 따로따로 분산되었을 때 당황하게 되고 개인주의는 뭉쳐 있을 때 당황하게 된다.
이는 오랜 기간에 걸쳐 길들여진 문화습관이기 때문이다.
한국은 1446년 훈민정음이 창제 되고서도 한자를 병행하여 사용하는 것에 길들여진 수직구조사회에서 1945년 8월 15일 세계대전 종전과 함께 서구문화가 들어오고 글자 사용에 있어서도 가로의 수평 글을 사용하면서 많은 혼란을 겪었다.
2000년대가 들어오면서 한글을 사용하는 한국인은 한글의 특성인 수직수평의 특성을 이해하고 자리 잡게 되면서 수직사고와 수평 사고를 동시에 이해하게 되었고 생활화하게 되었다.
한글을 사용하는 한국인의 사회구조는 수직수평사회구조이다.
한글을 사용하는 한국인의 사고능력도 수직수평사고능력을 갖추고 있다.
외국인들이 보는 한국인의 사고는 빨리빨리 이라고 한다.
이제 한국인의 사고는 반도체 사고방식으로 발전하게 되었다.
70억 지구촌인구 중에서 0.7%에 불과한 한국인이 2000년의 현대에 들어와서 세계문화와 경제 발전에 많은 변화를 가져온 것은 우연이 아닌 필연으로 해석해야 한다.
한국인이 발달시킨 빨리빨리 문화가 가져온 결과라고 보아야 한다.
수직사회에서 강국은 인구가 많아야 한다.
소수민족으로서 강국이 될 수는 없는 사회구조가 수직사회이다.
처음의 인간사회구조는 수직사회로 시작하였다.
문화가 발달하면서 점점 수평사회화 하였고 완전한 수평사회는 존재할 수가 없는 사회구조 특성상 이상적인 사회는 수직수평사회일 때라고 할 수 있다.
사회구조는 능력에 따라 강국으로도 발전하고 약소국으로 발전하게 된다.
능력은 수직수평사고를 동시에 할 수 있는 사고체제를 요구하고 있다.
인간사회는 능력의 사회고 능력은 사회를 주도한다.
현대사회는 정보사회이고 빠른 정보능력이 주도하게 된다.
정보능력은 반도체글자에서 시작된다.
반도체적성글자를 사용하지 않으면 정보에 밀리게 되고 정보에 밀리면 주도국이 될 수 없다.

한글은 반도체글자로서 가장이상적인 입력글자이며 실행글자이다.
알파벳의 26자 문자에서 대소를 구분하는 입력장치 1개가 더 있어서 사실상 알파벳문자는 27개의 문자입력단추가 있어야 한다.
알파벳대소글자를 구분하여 입력하는 입력단추는 한글입력장치에는 사실상 필요 없는 입력단추이다.
알파벳은 발성기호 없이 소리전달이 불가능하고 묵음이나 같은 자소로서 2 ~ 3개의 발성기호로 표기하는 경우도 있다.
한글의 글자는 대소를 구분할 필요가 없고 발성기호 없이 사용되는 글자로서 묵음이나 다르게 발음되는 경우는 더더욱 없다.
또 한글은 실행에서 초성 중성 종성으로 나누어 사용할 수 있고 초성이나 종성이 중첩으로 겹쳐서 사용할 수 있고 중성도 중성이 2 ~ 3 중첩으로 겹쳐 사용할 수 있어서 따로 발음부호를 사용할 필요가 없는 글자이며 수직수평조립글자는 실행공간에 수직으로 사용하여도 불편함이 없는 것이 특징이다.
입력장치를 통하여 입력된 정보가 실행공간에 차지하는 면적도 같은 내용의 정보를 입력할 경우 실행공간면적소요가 한글보다는 알파벳글자에서 30% ~ 50%를 더 소비하게 되어 있다.

영어의 결점은 소리문자이지만 완벽하지는 못하다는 것이다
Dermatitis[d□□ː rm□ta□iti]에서 같은 'ti' 를 tai와 ti로 발음해야 하고 있어서 소리로서 글자를 실행시키는데 혼동이 온다.
영어는 문자와 소리가 일치하지 못하고 때로는 묵음처리 된다.
같은 문자를 i와 ai로도 발음하여 2중으로 나타나면 다르게 또는 전혀 전달이 불가능한 경우도 있는 것이 반도체의 특성이다.
또한 영어는 수직으로 나열되지 못하여 공간면적이 효율적으로 이용될 수가 없다.
낱말은 글자가 모여서 이루어지고 낱말과 낱말의 구별은 공백으로 표시되므로 컴퓨터는 글자사이에서 공백이 생기면 다른 낱말로 인식한다.

낱개자소는 글줄의 정열에서 낱말에 글자가 많고 길면 글줄말미공간이 부족하면 다음줄로 글줄을 바꾸어 실행된다.

낱글자인 알파벳의 글줄정열메뉴에는 3개가 있다. 이것은 Excel 2007 홈 메뉴에서 따온 것이다. 한글 2007 글줄정열메뉴에는 와 같이 4개로 되어 있다.

알파벳과 같은 낱글자에는 한글에 있는 와 같은 앞뒤 양쪽정열을 을 할 수 없어 글줄의 정열에 문제가 일어난다.

따라서 실행공간이 0.3 ~ 0.5배 더 필요한 요인이 된다.

같은 내용의 글을 입력한 실행공간면적을 비교하여도 엄청난 차이가 난다.
아래 글에서와 같이 한글이 9줄을 채우지 못하는 내용을 영어로 옮길 경우 12줄에 해당하여 영어는 한글에 비해 1/2~3 더 필요하다
차세대는 정보전달 속도가 문화권을 지배하게 된다.
알파벳과 한글만이 반도체적성글자라고 볼 때 전달속도가 알파벳이 한글을 절대적으로 따라올 수가 없을 뿐만 아니라 알파벳의 문자를 사용하는 경우 수평문화가 발달하게 되고 수직문자를 사용할 경우 수직문화가 발달하게 되어 있다.
그렇다면 수직수평문자를 사용하는 경우는 어떠할 것인가?
인간의 사고와 사회구조는 뇌의 발달과도 연관이 있다고 보아야 한다.

한국, 중국, 일본의 3국은 한자글자를 사용하면서 고래로터 붓으로 세로로 문자를 입력하였고 한지의 두루마리에 말아서 휴대하게 되었다.
병사들의 전령을 적은 전문이나 상소문을 입력하여 전달하는 장치로서는 한지두루마리에 세로로 문자를 입력하고 두루마리를 가로로 펴면서 읽기에는 수직수평조립문자기록방식이 적당하다.
수직수평조립문자를 사용하면서 조직사회 구성도 수직사회 피라미드구조로 이루어지게 되고 집단주의가 발달하였다.
알파벳과 같은 수평나열글자는 수평문화가 발달하였고 개인주의가 발달하였다.
반도체산업이 발달되기 전의 지구촌의 문화는 힘의 논리로 영토와 자원이 크고 풍부하거나 인구가 많은 국가나 민족이 지배하였으나 반도체가 날로 첨단으로 발전하게 되면 영토나 자원 인구에 관계없이 첨단반도체적성글자로서의 적성을 갖춘 글자를 사용하는 조직이 중추가 되어 사회를 지배하고 이끌어가게 된다.
양차 대전 후 이스라엘의 유대민족이 대전에서 겪은 고통을 이기기 위해 탈무드의 교육에 힘을 쓴 것이 세계발전의 중추로서 발전하기도 하였고 패전 일본이 경제동물(economics animal)이라고 할 정도로 세계경제시장에서 두각을 나타내던 것이 소강상태로 주저앉게 된 이유는 반도체문화로 접어들면서 나타난 현상이다.
필자가 50세 후반의 늦은 나이에 식물생리학을 연구하면서 양차대전과 열강의 소용돌이에서 반반한 과학자를 양성하지 못했고 한글로 된 학술자료라고는 찾아 볼 수 없는 열악한 과학기술환경에서도 컴퓨터에 의해 우주천체에서 태양광 발생소면시각변화량의 vector와 생체시계진자vector를 찾아내고 지구자전축의 흔들림의 실측하고 그 크기를 구할 수 있었다는 것은 첨단반도체를 이용한 혜택이라 할 수 있다.

## 15, 구개언로 달사총(口開言路 達四聰)과 빨리 빨리 민족혼

한국사회에 한글이 사용되기 시작한 것은 1446년 한글반포시기에서부터 시작하였다.
한글의 시작이 작개언로 달사총(作開言路 達四聰)에서 비롯하였고 인체발성구조에 기초하여 제작된 글자이며 반도체최적 자소요건인 입력과 실행이 일원화하는 “바로 자소”로서 입력단추수가 72 ~ 87개로 한정하는 요건인 72개 단추에 해당하는 위대한 정보글자이다.
인체발성원리로 초성 중성 종성으로 되어 있어서 따로 발음기호가 필요하지 않아 배우는 속도가 빠르고 전달속도가 빠르고 판독속도가 빠르면서 또 정확하다.
한글은 반도체기기에서 가장 적은 72개 단추로도 가독성이 가장 높은 글자이다.
따라서 한글을 사용하게 되면 사고력과 일상의 생활이 빨라지게 된다. 한국에는 “황의 법칙”이 통한다.
한국인의 피 속에는 빨리 빨리가 통한다.
한국인의 머리에는 빨리 빨리라는 민족혼이 자라고 있다.
이것이 한국인의 자산이고 유산으로 이어지고 있다.
한국인에게는 작개언로 달사총(作開言路 達四聰)으로 이어온 빨리 빨리 문화는 위대한

문화가 있다.
첨단반도체문화가 정착하면서 지구촌의 문화 중 가장 빠른 변화와 발전을 가져온 문화는 한국인들의 빨리 빨리 문화이다.
2000년대 초기 지구촌의 인구는 70억이며 그 중 한국인은 5천만으로 6.7%이지만 경제 7강과 올림픽 성적 8강 기능올림픽 1위이다.
1970년대 후반 가장 한국적인 것이 세계적이라고 선언한 정책이 있었다.
이것으로 세계는 한류의 바람이 일게 되었다.
음식도 건강식으로 한식의 우수성을 인정하고 선호 하고 있다.
이것의 근원은 빠르게 변화시키고 발전하게 하는 빨리 빨리라는 한국문화가 주효했기 때문이다.
첨단반도체는 정밀지향에 확대지향의 과학성을 근거로 발전하고 있다.
1947년 미국에서 시작한 반도체는 초기 일본의 소니에 의해 불붙기 시작하였고 한국이 반도체산업이 빛을 보기 시작한 시기는 1990년 전후로서 불과 20년 역사를 조금 넘었지만 첨단을 달리고 있는 것은 빨리 빨리 문화의 덕택이라고 해야 한다.
한국인의 머릿속에는 빨리 빨리라는 혼이 들어 있다.
한 때 사회일각에서 양은냄비라는 비난도 있었지만 결코 비난할 것은 아니며 장점으로 승화 발전시킨 결과라고 하여야 한다,
이것이 “황의법칙”으로 발전한 원동력이다.
메모리칩의 능력이 테라로 발전시키고 정밀도가 펨토수준으로 발전하게 된다.
빨리 빨리라는 한국의 혼이 아니면 불가능한 것이라 할 수 있다.

첨단반도체글자 한글이 제작될 당시 한국을 비롯한 중국 일본은 견고한 수직사회 구조였다.
수직사회는 구성원의 개성이나 능력은 철저하게 배제되는 사회 특성으로 발전의 속도는 기대 할 수 없다.
구성사회가 공정성을 잃고 부정과 비리가 난무 한 것이 고질적인 병폐로서 한글의 창제자는 한글창제의 목표를 “작개언로 달사총(作開言路 達四聰)”에 두었다.
당시 백성의 소리를 들을 수 있도록 사간원이라는 관리를 두었으나 관리 구실이 제대로 되지못한 병폐를 바로 잡기 위해서는 왕이 바로 백성의 소리를 듣고자 하지만 백성은 글을 몰라서 관리를 통하게 되고 관리는 개관적으로 일을 처리하지 않고 유리한 자기주장으로 처리하여 백성사회가 피폐하게 되었다.
한국인의 빨리 빨리 문화는 여기에서 싹트기 시작한 것으로 볼 수 있다.
2차 세계대전과 6 25전쟁으로 한국사회는 수직문화의 고리에서 수평문화의 고리로 발전하는 계기가 되었다.
전쟁을 통하여 UN의 외국군이 들어오게 되어 여러 나라문화를 접하게 되는 계기가 되어 갑자기 들이닥친 수평사회구조에 적응하기가 힘들었고 혼란사회를 청산하기 위해 한국인의 의식구조는 혁명이 필요했고 첨단반도체적성글자인 한글의 혼을 찾을 필요성이 있었다.
이렇게 하여 시작된 것이 429㎞ 대장정의 경부고속도로 건설을 위한 토목사업이었다.
당시 기술로는 상상을 초월하는 3년 기간의 공기에서 빨리 빨리 혼을 찾을 수 있다.
경부고속도로는 이렇게 하여 세계역사에서 찾을 수 없는 최단 3년의 건설공기를 통하여 개통이 된 것이다.
국가 근대화 사업이 영국의 400년 독일이 120년 일본의 90년의 기간을 한국은 30년으

로 최단기간에 근대국가로 기반을 구축한 것은 우리의 혼 빨리 빨리라는 것이 있었기 때문이다,
브라운관이 등장하면서 한국은 보아의 춤과 노래가 지구촌에 퍼지고 비보이의 현란한 몸동작이 세계를 매료 시킬 수 있었고 한류의 열풍을 불게 한 것은 반도체 시대의 적성을 만족시키는 빨리 빨리라는 한국의 혼과 수직수평을 모두 만족시키는 반도체문자를 사용하면서 길들여진 수직수평사고의 덕택이라고 보아야 한다.

16, 아토액사(atto,$10^{-18}$,exa,$10^{18}$)과학시대 한글

아토액사(atto, exa)라는 수를 반도체상으로는 나타내면

$2^{-4}$ =115,281,504,606,846,976)

필자는 이 글은 통하여 겁 없이 한글학자의 부재라는 표현을 했다.

사실 부재라는 표현이 맞는 것은 결과론으로 따져서 하는 말이다.

2000년 현시대 학자는 어느 학문이고를 따지기 전에 반도체시대 학자로서 반도체원리인 2진법의 진법체계를 모른다면 학자로서의 자격이 없다고 보아야 하기 때문이다.

컴퓨터를 모르고 컴퓨터를 사용하지 않고 자기 학문의 세계를 표현할 수 없기 때문이다.

이러한 상황에서 반도체기기에서 글을 다루어야하는 한글학자가 2진법의 체계를 모른다면 전체를 모르는 것이나 다름이 없다고 보아지기 때문이다.

좋든 싫든 반도체시대 반도체기기의 이해를 위한 진법체계파악은 필수라는 것이 된다.

유니코드가 16진법으로 확장 될 때가지만 해도 프로그래머들이 한글부문에 26,000개 한글코드 값으로 배정하기를 꺼려했다.
원인은 16진법으로는 $2 \times 10^{1}$ □ = 65,536개의 글자 값이 확보 될 뿐이므로 라는 이유에서 이다.
중국의 50,000개 옛한글 포함 26,000개를 합한 76,000개 글자 값을 배당할 수도 없는 형편이라 쪼개고 쪼개어 배당하다보니 한글은 어쩔 수 없이 갖은 방법을 동원하게 되었으나 결과적으로는 뾰족한 해결은 찾을 수 없었다.
글자 값이 원활하게 배당 되려면 적어도 100,000개 글자 값이 필요한 것이어서 이들 글자 값을 모두 수용할 수 있는 방법으로는 32진법으로 더 많은 글자 값을 확보할 수 있는 기술개발이 있어야 했다.

$2^{3^{2}}$ =4,294,967,296개

2□⁴ =115,281,504,606,846,976개 글자 값

이렇게 글자 값이 충분하게 확보되면 한글자소에 대한 짜깁기 식으로 배당하는 일은 사라지게 되고 안심하고 중국글자 값과 한글 글자 값을 배당할 수 있게 된다.

홉으로는 절대로 말박이나 멱서리 섬박을 될 수 없듯이 그간 아스키코드수준으로 한글과 중국글자를 처리하려고 무진 애를 썼으나 모두가 허사였지만 지금은 눈치 볼 필요 없이 100,000개 글자 값을 빠짐없이 모두 배정되어 웹상으로 일어나는 글자 값 사이의 충돌에서 벗어나게는 되는 것이다.

이 글에서 마무리 결어를 맺는다면 지난과거야 어찌 하였든지 간에 앞으로 한글자소 발전 방향은 반도체기기사용의 전성시대를 준비하여야 한다는 것이어야 한다.

32진법이상으로 발전하면 반도체는 무성하게 가지치고 꽃피어 열매 맺는 시대가 전개될 것이고 이는 모두 한글자소의 몫일 수밖에 없다.

이를 대비하여 필자는 자소 판과 코드표를 개발하였다.

이를 사용하여 한글운용체제가 연이어 개발될 것으로 본다.

아래 표를 보면 프로그램언어는 1950년대 말에서 현재까지 20개가량 개발 되어 있다.

한국은 프로그램개발에 모두 외국의 것을 빌어다 사용했다.

한글로 된 프로그램개발언어가 없다는 것은 프로그램을 개발하기 위하여 외국부터 배워야 하고 외국어 배우는데 시간을 소비하고 나면 개발할 시간적 여유가 없어 한글프로그램언어개발은 요원한 것이 된다.

어떤 방법으로든 한국인에 의한 한국의 한글프로그램언어로 프로그램 소프트웨어가 개발되어야 한다.

| 프로그램명 | 생산년도 | 원산지 | 용 도 |
|---|---|---|---|
| 베이직 | 1964 | 미국 | 절차형 언어 |
| C | 196년<br>1973 | 미국 | C++는 C에서 객체 지향형으로 발전 |
| C# | 2003 | 미국 | 마이크로소프트에서 개발한 객체 지향 프로그래밍 언어 |
| C++ | 1983 | 미국 | 명령형 프로그래밍 언어,객체 지향 프로그래밍 언어 |
| D | | 미국 | 객체 지향 명령형 프로그래밍 언어<br>C++의 후손 |
| F# | | 미국 | 마이크로소프트에서 함수형, 명령형, 객체지향형 프로그래밍 언어 |
| 파이썬 | | 미국 | 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어 |
| 루비 | 1995 | 일본 | 동적 객체지향 스크립트 프로그래밍언어 |
| 자바 | 1995 | 미국 | 가전제품 내에 탑재해 동작하는 프로그램을 위해 개발했지만 현재 웹 애플리케이션 개발에 가장 많이 사용하는 언어 가운데 하나이고, 모바일 기기용 소프트웨어 개발에도 널리 사용하고 있다. 현재 버전 7까지 출시했다. |
| 파스칼 | 1969 | 스위스 | 명령형 프로그래밍 언어 절차적 프로그래밍 언어 구조적 프로그래밍 언어 |
| 프롤로그 | 1973 | 프랑스 | 논리형 프로그래밍 언어이다 |
| 포트란 | 1954 | 미국 | 명령형 프로그래밍 언어 절차적 프로그래밍 언어 과학계산기용 산술 기호(+, |

| | | | |
|---|---|---|---|
| [illegible] | [illegible] | IBM | -, *, / 등)를 그대로 사용할 수 있음 |
| 코볼 | 1959 | 미국 | 객체 지향 프로그래밍, 사무처리 |
| 리스프 | 1958 | 미국 | 함수형 프로그래밍 언어, 수학 표기법 |
| 펄 | | 미국 | 인터프리터 방식의 프로그래밍 언어 |
| R | | 뉴질랜드 | 통계 계산과 그래픽을 위한 프로그래밍 언어, 자유 소프트웨어 통계 소프트웨어 |
| 그루비 | | 영국 | 자바에 파이썬, 루비, 스몰토크 등의 특징을 더한 동적 객체 지향 프로그래밍 언어 |
| 스칼라 | | 미국 | 객체 지향 프로그래밍 언어와 함수형 프로그래밍의 요소가 결합된 다중패러다임 프로그래밍 언어, 자바의 자바 가상 머신 에서 실행 |
| occam | | 영국 | 명령형 프로그래밍 언어, 절차적 프로그래밍 언어, 파스칼과 같이 명령형절차적 |

**한국인은 한글이 탄생하기 전부터 중국의 글을 빌어다 썼고 지금도 중국한자로 된 말을 사용하고 있다.**

**한편에서는 한글전용이냐 한문혼용이냐를 두고 오랜 줄다리기를 했다.**

**필자는 이런 줄다리기는 매우 어리석은 짓으로만 보인다.**

**지금은 영어를 많이 쓰고 있다.**

**한글전용이냐 영어혼용을 따질 것이냐 ?**

**필자는 어떠한 방법으로건 즉, 자국어이든 외국어이든 풍부한 언어를 사용할 수 있다는 것은 매우 환영해야 할 것으로 본다.**

**현대는 국경의 턱이 낮아지고 지구촌이라고 할 정도로 문화권의 벽이 허물어진 글로벌 시대라고 한다.**

**언어는 연장이나 도구와 같은 것으로 연장이 많고 도구가 많으면 그만큼 일의 능률이 올라가고 발전적일 수 있기 때문이다.**

**프로그램언어개발과 사용에 있어서도 굳이 한글이어야만 한다고 고집할 수는 없다.**

**왜냐하면 반도체적성으로 보면 낱개자소와 조립자소 중에서 프로그램언어로 사용하기에는 낱개자소가 유리한 점도 많다고 본다.**

**지금까지 개발한 프로그램언어에서 낱개자소로만 사용하여 개발된 언어 즉, 알파벳이나 알파벳 파생 언어로 된 프로그램에서 구조상으로 부족한 점들도 찾을 수 있다.**

**즉, 조립자소와 낱개자소를 섞어서 개발하였다면 더욱 나아졌을 것이다.**

**따라서 한글 프로그램언어개발은 한영을 혼용한 개발이 되어야 할 것이다.**

**반도체에 있어서 2진법비트의 확장은 글자 값에만 한정된 것이 아니고 IP(인터넷 프로토콜,Internet Protocol) 주소의 확장에도 있다.**

**최근 IP주소는** IPv4(Internet Protocol version 4)의 주소체계는 총 12자리 IPv4 주소는 32비트($2^{3}$ $^{2}$ )가 고갈됨에 따라 IPv6(Internet Protocol version 6)주소체계 주소의 길이가 128비트($2^{128}$인 $3.4x10^{38}$) 로 확장된 것에서 찾을 수 있다

IPv4는 32비트의 주소공간을 제공함에 반해, IPv6는 128비트의 주소공간을 제공한다.

32비트 주소공간이란, 32 비트로 표현할 수 있는 주소영역을 지칭한다. 32 비트에 의해 생성할 수 있는 모든 IPv4 주소는 $2^{32}$ 인 4,294,967,296개이다.

IPv6의 128비트 주소공간은 128 비트로 표현할 수 있는 2128개인 약3.4x1038개(340,282,366,920,938,463,463,374,607,431,768,211,456개)의 주소를 갖고 있어 거의 무한대로 쓸 수 있다.

IPv4 주소에 비해 IPv6 주소는 그 표현 비트 수가 128비트로 IPv4의 32비트에 비해 4배가 되었지만, 생성되는 IPv6 주소 공간 영역은 IPv4 주소공간에 비해 296배의 크기를 갖는다.

IPv6 주소공간은 향후 인터넷에 등장할 대량의 유비쿼터스 통신 장치들이 상호 통신을 할 수 있는 주소공간을 제공할 수 있다. 냉장고, TV, AV 스피커, DVD 플레이어, 홈 보안장치, 전화기 등 각 요소 장비들이 지능화하면서 동시에 무선 인터넷 등을 통해 상호 통신할 수 있도록 각 장치(device)에 IPv6 주소를 제공할 수 있다.

128비트의 주소공간은 지표면의 모든 공간에 10m2당 1개씩의 IPv6/48 네트워크를 제공할 수 있을 만큼의 많은 개수를 갖는다. 어떤 사람들은 이 주소가 지나치게 많고, 그렇게 많은 네트워크가 필요하게 될 일은 영원히 없을 거라고 주장하지만, IPv6의 128비트 주소공간은 주소가 바닥나는 것을 막는 것 외에도 네트워크가 여러 개의 작은 단위로 조각나는 것을 막아 라우팅을 빠르게 만들기 위한 목적도 갖는다.

위와 같이 인간이 사용하는 수의 체계가 확대되어가고 있다.

그런가 하면 정밀하게는 펨토(10□¹ □, femto)수준으로 정밀해져서 펨토과학이라고 한다.

레이저를 이용하여 펨토초의 순간을 촬영할 정도로 발전해 있다.

2002년 월드컵 축구 상암 운동장에 발생한 UFO사건이 있었다.

한 제보자에 의하여 카메라에 비행물체가 잡힌 것이 발견되어 이 비행물체의 실체가 무엇이냐 하는 사건이 있었다.

동영상 카메라에서는 1 초에 30 컷의 화상이 만들어지는 것으로 1초에 30 컷의 화상이 이어지면 인간의 눈은 움직이는 것으로 느껴지기 때문이다.

한참 축구경기가 진행되고 쉴 새 없이 카메라가 돌아가고 있는데 난데없이 날 파리 한 마리가 카메라 앞을 날아가게 되었다.

카메라에 날 파리의 비행이 포착되었으나 날 파리의 날개 짓은 초당100회하고 한다.

이를 수직으로 정리하면

날 파리 날개 짓 = 100회/초

날 파리의 동영상 카메라 컷 수 = 30컷/초

초당 30컷 카메라로는 초당 100회 날 파리 날개 짓을 찍어 낼 수가 없어 UFO 라는 비행물체로 볼 수밖에 없다.

2002년 월드컵 상암 경기장의 UFO사전은

곤충학자들의 해명으로 날 파리 비행으로 끝났다.

**앞으로 생물을 다루는 자연과학에는 펨토과학과 같이 좁은 면적 짧은 순간을 반도체기 기카메라와 현미경으로 포착하는 기술이 개발되어야 발전하게 된다.**

| 국제 수의 접두어 체계 SI | | | | | | | |
|---|---|---|---|---|---|---|---|
| 접두어 | 기호 | 배수 | $10^n$ | 접두어 | 기호 | 배수 | $10^n$ |
| 요타 yotta | Y | 자 | $10^{24}$ | 욕토 yocto | y | 일자분의 일 | $10^{-24}$ |
| 제타 zetta | Z | 십해 | $10^{21}$ | 잽토 zepto | z | 십해분의 일 | $10^{-21}$ |
| 엑사 exa | E | 백경 | $10^{18}$ | 아토 atto | a | 백경분의 일 | $10^{-18}$ |
| 페타 peta | P | 천조 | $10^{15}$ | 펨토 femto | f | 천조분의 일 | $10^{-15}$ |
| 테라 tera | T | 조 | $10^{12}$ | 피코 pico | p | 일조분의 이 | $10^{-12}$ |
| 메가 mega | G | 십억 | $10^{9}$ | 나노 nano | n | 십억분의 일 | $10^{-9}$ |
| 기가 giga | G | 백만 | $10^{6}$ | 마이크로 micro | ■ | 백만분의 일 | $10^{-6}$ |
| 킬로 kilo | K | 천 | $10^{3}$ | 밀리 milli | m | 천분의 일 | $10^{-3}$ |
| – | – | 1 | $10^{0}$ | – | – | 1 | $10^{0}$ |

**필자가 식물의 광합성체계연구를 하면서 펨토과학과 같은 수준의 시설의 확보가 없는 현시점에서 더 이상의 연구가 불가능함을 깨닫고는 접어야 했다.**

**그리고 느낀 것이 앞으로 정밀과학연구는 첨단반도체기술이 등장해야 하는 첨단반도체 국가의 몫이라는 결론을 내렸다.**

**다시 말해 이러한 과학은 모두가 정밀한국과학의 몫이라는 것이다.**

- **자판**
-

| 십진법 | 십육진법 | 약자 | 설명 | 한글설명 |
|---|---|---|---|---|
| 0 | 00 | NUL | Null | 공백 문자 |
| 1 | 01 | SOH | Start of Header | 헤더 시작 |
| 2 | 02 | STX | Start of Text | 본문 시작, 헤더 종료 |
| 3 | 03 | ETX | End of Text | 본문 종료 |
| 4 | 04 | EOT | End of Transmission | 전송 종료, 데이터 링크 초기화 |
| 5 | 05 | ENQ | Enquiry | 응답 요구 |
| 6 | 06 | ACK | Acknowledgment | 긍정응답 |
| 7 | 07 | BEL | Bell | 경고음 |
| 8 | 08 | BS | Backspace | 백스페이스 |
| 9 | 09 | HT | Horizontal Tab | 수평 탭 |
| 10 | 0A | LF | Line feed | 개행 |
| 11 | 0B | VT | Vertical Tab | 수직 탭 |
| 12 | 0C | FF | Form feed | 다음 페이지 |
| 13 | 0D | CR | Carriage return | 복귀 |
| 14 | 0E | SO | Shift Out | 확장문자 시작 |
| 15 | 0F | SI | Shift In | 확장문자 종료 |
| 16 | 10 | DLE | Data Link Escape | 전송 제어 확장 |
| 17 | 11 | DC1 | Device Control 1 | 장치 제어 1 |
| 18 | 12 | DC2 | Device Control 2 | 장치 제어 2 |
| 19 | 13 | DC3 | Device Control 3 | 장치 제어 3 |
| 20 | 14 | DC4 | Device Control 4 | 장치 제어 4 |
| 21 | 15 | NAK | Negative Acknowledgement | 부정응답 |

| | | | | |
|---|---|---|---|---|
| 22 | 16 | SYN | Synchronous idle | 동기 |
| 23 | 17 | ETB | End of Transmission Block | 전송블록 종료 |
| 24 | 18 | CAN | Cancel | 무시 |
| 25 | 19 | EM | End of Medium | 매체 종료 |
| 26 | 1A | SUB | Substitute | 치환 |
| 27 | 1B | ESC | Escape | 제어기능 추가 |
| 28 | 1C | FS | File Separator | 파일경계 할당 |
| 29 | 1D | GS | Group Separator | 레코드 그룹경계 할당 |
| 30 | 1E | RS | Record Separator | 레코드 경계 할당 |
| 31 | 1F | US | Unit Separator | 장치 경계 할당 |
| 32 | 20 | □ | 공백 null | |
| 33 | 21 | , | | |
| 34 | 22 | . | | |
| 35 | 23 | : | | |
| 36 | 24 | ; | | |
| 37 | 25 | " | | |
| 38 | 26 | ' | | |
| 39 | 27 | \| | | |
| 40 | 28 | ₩ | $ | |
| 41 | 29 | / | | |
| 42 | 2A | % | | |
| 43 | 2B | + | | |
| 44 | 2C | − | | |
| 45 | 2D | × | | |
| 46 | 2E | ÷ | | |
| 47 | 2F | = | | |
| 48 | 30 | 〈 | | |
| 49 | 31 | 〉 | | |
| 50 | 32 | 1 | | |
| 51 | 33 | 2 | | |
| 52 | 34 | 3 | | |
| 53 | 35 | 4 | | |
| 54 | 36 | 5 | | |
| 55 | 37 | 6 | | |
| 56 | 38 | 7 | | |
| 57 | 39 | 8 | | |
| 58 | 3A | 9 | | |
| 59 | 3B | 0 | | |
| 60 | 3C | @ | | |
| 61 | 3D | # | | |
| 62 | 3E | ! | | |
| 63 | 3F | ? | | |
| 64 | 40 | _ | | |
| 65 | 41 | ㄱ | | |
| 66 | 42 | ㄴ | | |
| 67 | 43 | ㄷ | | |
| 68 | 44 | ㄹ | | |
| 69 | 45 | ㅁ | | |
| 70 | 46 | ㅂ | | |
| 71 | 47 | ㅅ | | |
| 72 | 48 | ㅇ | | |
| 73 | 49 | ㅈ | | |

| | | |
|---|---|---|
| 74 | 4A | ᄎ |
| 75 | 4B | ᄏ |
| 76 | 4C | ᄐ |
| 77 | 4D | ᄑ |
| 78 | 4E | ᄒ |
| 79 | 4F | ᄀᄀ |
| 80 | 50 | ᄃᄃ |
| 81 | 51 | ᄇᄇ |
| 82 | 52 | ᄉᄉ |
| 83 | 53 | ᄌᄌ |
| 84 | 54 | ᄀᄉ |
| 85 | 55 | ᄂᄌ |
| 86 | 56 | ᄂᄒ |
| 87 | 57 | ᄅᄀ |
| 88 | 58 | ᄅᄆ |
| 89 | 59 | ᄅᄇ |
| 90 | 5A | ᄅᄉ |
| 91 | 5B | ᄅᄐ |
| 92 | 5C | ᄅᄑ |
| 93 | 5D | ᄅᄒ |
| 94 | 5E | ᄆᄉ |
| 95 | 5F | ᅡ |
| 96 | 60 | ᅢ |
| 97 | 61 | ᅣ |
| 98 | 62 | ᅤ |
| 99 | 63 | ᅩ |
| 100 | 64 | ᅪ |
| 101 | 65 | ᅫ |
| 102 | 66 | ᅭ |
| 103 | 67 | ᅳ |
| 104 | 68 | ᅵ |
| 105 | 69 | ᅨ |
| 106 | 6A | ᅦ |
| 107 | 6B | ᅧ |
| 108 | 6C | ᅰ |
| 109 | 6D | ᅴ |

| | | | |
|---|---|---|---|
| 110 | 6E | ㅗ | |
| 111 | 6F | ㅘ | |
| 112 | 70 | ㅙ | |
| 113 | 71 | ㅚ | |
| 114 | 72 | ㅛ | |
| 115 | 73 | ㅜ | |
| 116 | 74 | [ | |
| 117 | 75 | ] | |
| 118 | 76 | { | |
| 119 | 77 | } | |
| 120 | 78 | ( | |
| 121 | 79 | ) | |
| 122 | 7A | & | |
| 123 | 7B | * | |
| 124 | 7C | ^ | |
| 125 | 7D | ` | |
| 126 | 7E | ~ | |
| 127 | 7F | 나가기 | 지우기 Delete |

한글 자소코드표(필자 제작)

## 17 늘어나는 한국어의 수요

세계는 2000년대 들어 한류의 뜨거운 열풍으로 한국을 배우려는 사람이 늘어나고 있다.

한국어의 수요가 증가한다는 것이다.

대부분의 외국인들은 한국어를 학문의 수준으로 배운다고 한다.

이보다는 생활어로 배우는 수준으로 늘어났으면 하는 것이 필자의 바람이다.

한국어는 배우기 힘든 언어라고 하지만 어찌하였든 한국어를 배우게 되면 한글도 배우게 된다.

언어습관은 문화의 차이 이고 언어습관 때문에 한국어라는 외국어를 배우기가 힘들지만 한글을 사용하게 되면 한글의 우수성을 알게 될 것이다.

- 외국인 한국어 학습자 및 자격시험

미국, 중국, 일본, 프랑스, 폴란드, 카자흐스탄, 러시아 등지에서는 한국어를 배우려는 사람들도 생겼다. 이는 한국의 경제 성장에 따른 국제적 위상의 확대와 한류 등의 문화적 영향력의 전파에 힘입은 바가 크다. 그러나 아직 아시아 언어 중에서는 중국어나 일본어에 비해 학습자의 수가 적은 편이며, 체계적인 교수법이나 교재도 부족한 형편이다. 과거에는 영어, 일본어 등 유럽, 미주 및 일본을 중심으로만 한국어 학습용 교재가 발간되었으나, 근래에는 한국어 학습 동기의 다변화와 국내 외국인 수의 증가로

중국어, 타이어, 인도네시아어, 베트남어 등 다양한 언어로 한국어 교재가 발간되고 있다.

미국 정부회계감사원이 발표한 '미국 국무부 외국어 직무수행 평가서'에 따르면, 중국어, 일본어, 아랍어와 더불어 한국어를 미국인이 가장 배우기 힘든 언어로 분류하고 있는데, 이는 인도유럽어족인 영어와 한국어의 여러 상이점에 따른 것이며, 절대적 난이도는 아니라[출처 필요]고 여겨진다.

한편, 한국어를 배우는 대부분의 외국인들은 한국어를 학문으로서 배운다.[11]

### 한국어 검정시험

#### 대한민국

대한민국에는 현재 한국어 능력을 검정하기 위한 시험이 몇 가지 있다. 국어능력인증시험(Test Of Korean Language;TOKL) 한국어능력시험(Test Of Proficiency in Korean ; TOPIK) KBS 한국어능력시험(Korean Language Test) 등이 있다.

#### 일본

일본의 네 가지 한국어 시험 가운데 일본에서 비교적 널리 알려진 시험은 한글능력검정시험과 한국어능력시험이다. 한글능력검정시험을 뺀 나머지 모든 시험은 대한민국 표준어 시험이기 때문에 표준어가 기준이며, 이와 다른 문화어의 맞춤법이나 어법은 오답으로 처리된다.

##### 한글능력검정시험

일본의 특정비영리법인 한글능력검정협회가 주최하는 자격시험으로 6월경(년 2회)에 실시한다. 일본의 한국어 학습자에게 가장 잘 알려진 시험이다. 5급이 가장 낮은 급수이고 4급〈3급〈준2급〈2급〈1급순으로 급수가 올라간다. 일본 국내에서만 통용되며 등급이 영어검정(일본)시험과 거의 같기 때문에 영어검정시험과 비교대조되는 경우가 있다. 2006년부터 "준 1급"이 없어졌다. 1, 2급은 문제의 지문을 포함하여 모든 글이 한국어로 표기되어 있다. 또한 이 시험은 답을 적을 때 대한민국이든 조선민주주의인민공화국이든 어느 한쪽으로 일관성 있게 통일되어 있으면 정답으로 간주한다. 근래 한류 붐의 영향으로 초급 수험자는 상당히 증가했으나 반대로 1,2급 등 고급 레벨에서는 수험자 수가 매우 적다. 2004년 전후에 상급의 시험 문제는 난이도가 계속 상승하는 반면, 3급 이하의 급수는 합격률이 90%를 넘나들게 쉬워지는 현상이 일어났다. 그러나 2006년 다시 출제 기준과 난이도 조정이 이루어진 결과, 낮은 급수도 난이도가 대폭 상승하여 현재에 이른다. 이러한 난이도의 유동에 따라 자격 시험으로서 신뢰도가 떨어진다는 지적도 있다.

##### 한국어능력시험(TOPIK)

한국교육과정평가원이 주최하고 교육과학기술부가 인정하는 자격시험으로 매년 4월과 9월에 실시된다. (한국에서는 2007년부터 일본에서는 2008년부터 연 2회 볼 수 있게 되었다) 한글능력검정시험과 달리 1,2급이 초급이고 3,4급이 중급, 5,6급이 고급단계이다. 한국, 일본 외에 세계 28개국에서 실시되는 국제적 시험이다. 수험자수가 가장 많은 국가는 중국이고 두 번째가 일

본이다. 외국인이 한국의 대학교, 대학원에 입학할 때 이 시험의 성적증명서의 제출이 요구되는 경우가 많다. 또한, 한국의 많은 외국인 대상 어학당에서 이 시험의 결과로 반을 편성한다.

세계한국말인증시험[12](KLPT)(일본어: 世界韓国語認証試験（せかいかんこくごにんしょうしけん))

한글학회가 주최하는 자격시험으로 4월 10월에 실시된다. 2006년까지는 1,4,7,10월의 연 4회 실시되었으나, 2007년부터 연 2회로 단축되었다. (수험자 수가 적은 것이 원인으로 추측된다) 평가는 500점 만점의 점수제로 토익과 같은 형태이다.

한국어 레벨테스트(KLT)

점수제이고 1000점 만점이다. 시험시간은 90분이고 비교적 단시간에 시험을 볼 수 있다. 한국,일본외에 중국과 미국에서도 수험이 가능하다. 2004년부터 개시되었지만 2009년 1월 시험 실시가 정지되었다.

- 인터넷에서의 한국어

한국어 문서가 전 세계 웹에서 차지하는 비중은 2004년 기준으로 4.1% 정도이며 이는 35.8%인 영어, 14.1%인 중국어, 9.6%인 일본어, 그리고 스페인어, 독일어에 이어 전 세계 6위이다. 한글 문서와 한국어 문서를 같은 것으로 볼 때, 한국어 사용 인구가 약 6천6백만 명으로 전 세계 약 69억 명 인구의 1%에 조금 못 미치는 것에 비해서는 웹 상의 한국어 정보 비중이 많은 편이다.

18 더 빠르고 더 정밀하고 정확한 정보세계

자연과학에 있어서는 인간이 잘못 알고 있는 사실은 수 세기가 지나도록 정설로 믿어오다가 정설이 아님을 밝히는 처음의 연구자는 새로운 그의 이론이 밝혀질 때까지는 심한 반박을 받게 된다.

머리말에서 언급한 1543년 5월 24일 폴란드의 니콜라스 코페르니쿠스가 이에 해당하는 인물이라고 하여야 할 것이다.

그러나 코페르니쿠스는 박해나 반박은 받지 않았다.

니콜라우스 코페르니쿠스(1473년 2월 19일 – 1543년 5월 24일)는, 지동설을 주장하여 근대 자연과학의 획기적인 전환, 이른바 '코페르니쿠스의 전환'을 가져온 폴란드의 천문학자이다.

그는 천구의 회전에 관하여(De revolutionibus orbium coelestium)라는 책을 쓰고 출간을 미루어 오다 출간된 책이 나옴과 동시 그는 사망했다고 한다.

소심했던 그는 출간이후 돌아올, 반박과 박해가 두려워 미루다가 친구들의 권유에 못 이기어 출간했으나 책의 출간과 동시 세상을 떠난 것이다.

성웅이순신도 7년간에서 2 차례 백의종군과 23전 전승을 하고는 선조와 조정의 죄를 묻기 전에 그는 전사 하였다.

갈릴레오 갈릴레이(1564년 2월 15일 ~ 1642년 1월 8일)는 이탈리아에서 태어난 철학자이자 과학자, 물리학자, 천문학자이고 과학 혁명의 주도자이다. 갈릴레오는 요하네스 케플러와 동시대 인물이다. 그는 아리스토텔레스의 이론을 반박했고 교황청을 비롯한 종교계와 대립했다. 그의 업적으로는 망원경을 개량하여 관찰한 것, 운동 법칙의 확립 등이 있으며, 코페르니쿠스의 이론을 옹호하여 태양계의 중심이 지구가 아니고 태양임을 믿었다.

이러한 일로 갈릴레이는 종교재판을 받았다.

특히 이러한 반박과 박해는 종교계에서 심하다는 것을 역사를 통해 알 수가 있다.

종교의 경전은 과학교과서는 아니다.

자연과학에서 새로운 학설은 기성학설과 배치될 경우 심한 대립과 충돌이 야기된다.

이것이 종교와 관련될 경우 종교계는 더욱 반발하게 된다.

지적설계이론의 심한 대립을 받은 이론은 다윈의 진화론이다.

지금도 진화론의 한 부분인 한국의 시조새이론은 사건은 세계가 귀추하고 있다.

과학은 다수결의 원칙이 적용될 수 없는 것이다.

과학은 재판으로 받아들일 수는 없다.

과학은 과학 그 자체로서 받아들여야 한다.

다윈은 결코 생물의 분화가 진화론으로 발전시키지는 아니하였다.

최근에 와서 필자가 주장하는 환경동조산물로 발전하고 있음이 보인다.

생물이 살아남기 위해 진화한 것이 아니고 환경에 동조되었기 때문에 살아남아 생존한 것이라는 것으로 최근 후학들이 수정하여 해석하고 있다.

그만큼 과학의 이론을 증명하는 길은 수세기가 필요한 경우가 많다.

필자는 태양광발생소멸시각변화량 vector에서 놀라운 사실을 발견 하고 지구 흔들림에 대하여 라는 책을 썼다.

필자가 쓴 지구자전축의 흔들림의 원인은 지구를 이루고 있는 70%의 물의 이동이라는 것이다.

또 필자는 오존층이 남북의 양극상공에서 얇아지는 원인도 물의 이동에 의한 자전축의 흔들림에 의한 와류라는 것을 밝히고 있다.

오존층이 결코 화학적 파괴가 아닌 물리적인 것이라는 것이다.

## 필자가 주장하는 “광역동시다발정밀연구” 는 이래서 필요한 것이다.

인간이면 누구나 밤하늘의 별자리에 향수를 가지고 있으며 별자리로 인성을 분류하기도 하고 별자리 점을 치는 점성술이 발달하기도하였다.

지구 흔들림 연구는 수 천 년 전부터 인간이 바다에 배를 띄우고 어업활동을 하고 운반수단으로 항해를 하면서 망망대해에서 어두운 밤 항해의 길잡이로 별자리를 연구한 것에서 시작되었다고 할 수 있다.

극 상공에서 지구자전과 관계없이 유독 변화지 않은 별을 중심으로 별자리가 이동한 것을 보고 시각을 추정할 수 있었고 뱃길의 나아갈 방향을 짐작할 있는 수단이었기 때문이다.

지금의 지구 자전축은 북극성을 향하고 있지만 기원전 15세기의 이집트 사람들이 관찰할 당시에는 용자리 알파별이었다고 한다.

그리스의 히파르코스는 기원전 120년에 이전 천문학자들의 관측과 자신의 관측을 종합하여 세차운동을 발견하였다.

기원전 120년 고대 이집트 천문연구 자료에 의하면 별자리와 천체이동 중심이 변하고 있다고 주장하였다.

이후 세차운동 장동운동 챈들러진동 등으로 지구 흔들림을 설명하고 있다.

이러한 지구 흔들림의 설명은 흔들리는 그 원인에 대한 충분한 자료의 제시가 이루어지지 않고 있으며 설득력이 부족함을 쉽게 알 수 있다.

히파르코스(그리스어: ■■■■■■■■■, 기원전 190년 ~ 기원전 120년)는 고대 그리스의 천문학자, 지리학자, 수학자이다. 헬레니즘 시대에 활동했다.

삼각표를 가지고 다녔으며, 구면 삼각법에 관련된 몇몇 문제를 푼 것으로 보인다. 그는 일식, 월식, 그리고 삼각법을 이용해 처음으로 일식을 예견하는 방법을 개발한 선구자로 알려져 있다. 그의 유명한 발견은 세차운동, 포괄적인 초기의 성단표, 아스트로라베, 천구의 등이다. 클라우디오스 프톨레마이오스의 천동설이 우주론의 주류를 이루기 전까지 그의 발견은 널리 통용되었다.

그리스 사람들이 ‘스텔라 마리스(바다의 별)’ 라 부르던 별은 바다에서 항해하는데 꼭 필요한 별, 곧 북극성(polaris)의 별칭을 이렇게 불렸다. 그러나 이 북극성이라고 해서 영원불변하는 것이 아니라, 시대에 따라 북극성도 변한다. 북극성은 하늘의 북극에서 약간 떨어져 있기 때문에 실제로는 작은 원을 그리면서 이동하고 있다. 좀 더 긴 시간을 두고 북극성에 주목하면 재미있는 일이 벌어진다. 바로 위에서 얘기한 ‘세차운동’ 때문에 지축이 가리키는 별의 위치가 변하고, 이에 따라 하늘의 북극도 변하고 북극성의 역할을 하는 별도 바뀌게 된다.

필자는 세차운동을 이해하려고 무진 애를 썼으나 결국은 지구 흔들림의 원인을 찾게 되는 결과를 낳았다.

실로 천체우주공간에서 유영하는 항성지구가 흔들리는 것에 대한 원인과 실측을 한다는 것은 측량법이 발달하고 기하학이 발달하였다고 할지라도 이를 설득력 있게 계량정리하기는 매우 어려운 것이다.

자연과학의 연구는 선험이론에 근거하여 연구가 이루어지는 것이 대부분이다.

필자는 언젠가부터 선험연구가 정확한가를 분석하는 버릇이 생겨났다.

그리고는 광역동시다발정밀연구의 필요성을 주장하게 되었다.

지구의 자전축이 흔들린다는 사실은 확실하나 지구1공전의 주기 동안 365,25회자전이

일어나며 광역동시다발정밀연구에 해당하는 지구1공전의 주기 동안 365,25회 북극성을 매일 매일 실측한 자료가 있으며 동시다발이라면 그것도 지구표면의 위도와 경도에 따른 여러 지역에서 동시에 연구된 실측자료가 있는가 하는 것이다.
이렇게 반문하면 지금까지의 지구자전축 흔들림에 대한 연구로서 “세차운동, 장동운동, 챈들러 진동” 이니 하는 이론에 설득력이 있느냐 하는 것에는 의문만이 있을 뿐이다.
필자가 태양광발생소멸시각변화량 vector를 구하면서 앞선 선험연구가 설득력이 없다는 결론을 얻었다.
선험연구에 대한 여과 없이 받아들인 후계연구는 매우 위험한 연구라는 것이다.
첫 단추를 잘 못 꿰면 다음단추부터는 줄줄이 오류라는 것이다. 반도체는 과학에 있어서 많은 변화를 가져다주었다.
정밀성에는 엄청난 차이가 있음은 인정해야 한다.
육안관찰과 돋보기관찰, 돋보기관찰과 현미경관찰, 현미경에 있어서도 배율에 따른 관찰결과는 크나큰 차이가 난다.
이래서 지금 2013년 현시대를 펨토과학이라 한다.
정밀도가 10□□일 때와 10□¹ □일 때를 생각해 보자.
또 10□으로 한 수집 자료와 10¹ □로 수집 하누 자료에서 얻은 분석치를 비교하여 정밀도를 생각해 보자
뜨거운 용광로 속의 온도를 아날로그방식온도계에서 얻은 결과와 디지털방식온도계로 얻은 결과 정밀도를 비교해보자.
“빠른 정보, 정밀한 정보, 정확한 정보” 는 과학에 있어서는 생명이요 뿌리이다.
필자는 2006년 농촌진흥청의 연구개발과제의 하나로 “상 전환으로 동절기 엽경미나리 생산” 을 수행하기 위한 자료수집과정에서 처음으로 “광역동시다발정밀연구” 라는 신조용어를 사용하게 되었다.
필자의 연구결과에 의하면 미나리는 “전형적인 지구 1공전주기 한해살이 식물” 이라는 결과를 얻었다.
필자가 1994년 처음으로 미나리에 관심을 갖게 되는 시기에 서울생명과학대학을 중심으로 국내 여러 대학의 교수진 22명으로 구성된 미나리연구회라는 학술회가 있었다.
이렇게 구성된 미나리연구학술회는 물론 어디에도 미나리를 “다년생 식물” 로 분류되어 있었다.
미나리가 자라는 곳은 습지로서 1년 내내 미나리가 자라고 있는 것을 볼 수 있다.
미나리의 일생은 8월말 9월초를 분기점으로 하여 시작하게 된다.
미나리는 7월 초순 개화시작으로 8월말이면 거의 개화가 완료되고 잎과 뿌리가 죽게 되고 줄기의 마디에서 새로운 개체의 미나리가 영양번식을 하는 단계를 남기고 생을 마감한다.
새로운 미나리 세대가 시작된다는 것이다.
그러나 개화가 미쳐 끝나기 전 영양번식 하는 경우도 있다.
이러한 미나리는 잎의 죽음과 뿌리의 죽음시기가 불분명해지고 계속 개화하다 새로운 개체시기로 이어지는 것으로 자칫 잘못관찰하게 되면 다년생으로 오해하게 되기도 한다.
미나리는 습기만 있으면 무조건 계속 뿌리내림이 이어지고 새로운 개체의 영양번식이

일어난다.

그러나 분명한 것은 개화직전 미나리의 줄기단면은 원형에서 화살나무처럼 5각 또는 6각의 돌기가 형성되고 직립줄기로 유지하기 위한 기둥줄기로 변한다.

개화는 직립줄기에서 유지되어야 개화결실이 원활해지기 때문이다.

이러한 결실기가 지나면 줄기를 남기고 잎과 뿌리가 죽게 된다.

미나리라는 식물은 지구 1공전주기사이에 여러 형태의 위상으로 생활상을 연출하는 식물이었다.

해서 필자는 “미나리를 식물연구모델식물” 이라 신조어를 만들었다.

흔히 식물은 4계절에 따라 자라고 가지치고 꽃피고 결실하는 생활상을 연출 한다.

저위도 열대지역은 겨울이 없다고 한다.

현 시대를 지구온난화로 “겨울이 없어지는 등” 하고 있다.

필자가 “태양광발생소멸시각변화량 vector” 를 산출하면서 “0” 의 저위도지역에서 확실한 구분을 갖는 4계절이 발견되었다.

필자가 정확한 정보의 중요성을 주장하는 이유 중의 하나이다.

또 다른 측면에서 더 빠르고 더 정밀하고 정확한 정보세계의 필요성과 오류의 한 측면으로는 이러한 4계절의 발견과 동시 “지구자전축의 흔들림을 실측수치로 도출하고 증명하는” 기회를 가지게 되었다.

즉, 지구자전축의 흔들림 실측에서 “세차운동, 장동운동, 챈들러진동” 의 모순을 찾았다.

세차운동, 장동운동, 챈들러진동은 첫 째로 “지구가 강체일 때” 성립하는 이론이고 다음으로 “지구중심축이 어떤 면에 확실하게 고정되었을 때” 라는 2 가지원칙의 성립조건이 완벽하게 갖추어져야 한다는 결론을 얻었다.

“지구는 강체가 아닌 비강체가 확실한 항성이다.”

강체와 비 강체의 확실한 구분 없이 지구자전축 흔들림의 이론을 정립한다는 것은 크나큰 오류를 불러오는 결과를 가져온다.

역학이나 물리학에서 강체의 세차운동은 운동의 중심점이 평면위의 고정점이 있는 것과 회전력 소멸을 전제조건으로 한다.

쉽게 말하면 팽이(top)를 예를 들어 설명된다.

팽이의 회전은 지면상이라야 하고 회전력이 수직을 유지할 수 없을 정도로 감소할 때 세차운동이 발생한다.

세차운동은 지구중력의 힘과 강체가 보유한 회전력 사이에서 발생하며 강체의 회전력이 약화 소멸하는 전제 조건하에서 발생한다.

이러한 논리로 보아 필자는 지구의 흔들림이 세차운동으로는 설득력이 없음을 알 수 있었고 강체의 세차운동은 자력으로 발생하고 그 자력이 소멸하는 과정에서 세차운동과 자전운동이 2원적으로 발생하는 것이므로 지구는 결코 2원적운동에서 세차운동으로 자력의 운동이 소멸하는 것이 아니기 때문이다.

“태양광발생소멸시각변화량(scalar)을 벡터(vector)화 시키면

벡터란 한 점에서 다른 점을 향하는 방향을 가진 선분으로 표시되는 양《힘 · 속도 · 가속도 등을 나타냄》이라면 여기서 한 점 즉, 시작점은 지구자전축의 중심점이 있어야 하나 지구는 팽이처럼 지면상에서 존재하는 것이 아니고 공간상에서 떠 있는 물체에 해당하기 때문에 지구도 무게 중심이 있고 부피중심이 있다.

따라서 지구라는 물체가 강체라면 고정된 무게중심 또는 부피중심으로 자전이라는 회전운동을 해야 한다.

필자의 실측에서 지구1공전주기에서 한 번도 무게 중심이나 부피중심을 찾을 수가 없었다.

다시 말해 지구의 고정된 무게중심이나 부피중심이 존재하지 않는다는 것이다.

“지구는 자전축 중심이 규칙적으로 변하는 비 강체물체이다” 라는 결론이 나온다.

지구가 비 강체라는 것은 필자가 주장하지 않더라도 지구를 연구한 물리학자들이 직접적이 아닌 간접적으로 인정하고 있다.

달의 운동과 밀물썰물을 설명하는 벌지(bulge)현상에서 찾을 수 있고 해류학자들의 4계절 해류이동에서도 찾을 수 있다.

지구는 면적과 무게의 양적에서 70%가 고체가 아닌 유체물질인 물로서 70%가 구성되어 있고 이 물이 이동과정에서 자전축의 중심이동이 일어난다는 것을 배제할 수가 없다는 것이다.

지금까지 지구자전축의 흔들림 연구에 있어서 지구1공전주기 365,25회에 기준을 두고 매일매일 실측한 자료는 없다는 것이다.

지구는 생성에서부터 지금까지 내부에서 많은 화산활동을 해 오고 있었기 때문에 바다가 육지가 되고 육지는 침강하여 바다가 되기도 하는 많은 변화가 있었다.

이러한 변화는 결국자전축의 무게중심점의 이동이 일어나게 하였고 태양의 남북회기선 어느 위치에 있었느냐에 따라 남북반구의 눈과 얼음이 녹고 쌓여서 무게의 이동으로 자전축중심이동의 변수로 작용하였다고는 인정해야 한다.

지구를 이루고 있는 70%의 물의 이동은 자전축 흔들림의 절대적 변수에 해당한다면 지구는 비 강체이며 자전축의 이동을 세차운동이나 장동운동 그리고 챈들러운동으로는 설명할 수 없고 설득력도 없다고 봐야 하다.

19 한국의 근대과학 암흑사와 새 시대 희망과학미래사

근대사에서 한국과학사는 두 차례 암흑기를 맞게 되었다.

2차 대전으로 일본의 강점기징집과 1950년의 한국전쟁동원이다.

현대과학이 싹틀 시기인 1900년대 전반기인 이시기에 한국과학의 산실이 되어야 할 대학의 연구실학생들은 일본의 전쟁총알받이로 동원되어야 했고 1950년은 학도호국이라는 이름으로 전쟁터에서 사라졌다.

주인 잃은 연구실 공간에 실험 자료가 생겨날리 없어 신참의 연구욕망은 외국어를 배

워야 하는 2중고에서 한국과학은 암흑기를 보내고 있어야 했다.

필자가 2006년 “상 전환으로 동절기 엽경미나리 생산” 이라는 과제를 수행하기 위해 옛 문헌자료를 수집하는 과정에서 더 이상 발전이 없음을 느끼고 세계 석학들이 쓴 식물생리학을 구입하게 되고 심지어 원서까지 찾아야 했다.

그러나 그 어디에도 12월의 비닐하우스에 4월말의 인공기상을 만들어 주는 기술은 없었다.

확실하게 알 수 있는 것은 석학들이 썼다는 3권의 식물생리학 총 2,000페이지의 글의 참고문헌에서 한국인의 이름은 어디에서도 찾아 볼 수 없었다는 것을 기억한다.

근대 식물생리의 연구역사는 1729년 프랑스 천문학자 드 마렌에서 시작됨을 알 수 있었고 그 당시 마렌은 미모사 잎의 잠 운동 연구가 수행 되었다는 것이다.

이어 독일의 뷔닝에 의한 식물의 밤과 낮 개화의 인자는 무엇인가 하는 연구였다.

오늘날의 적색광과 근 적색색광의 경계에 대한 연구이었고 이들 경계가 개화에 미치는 영향에 관한 연구이다.

마렌에 이어 300년이 지난 현재 식물의 개화생리 연구 결말은 한결 같은 전통적인 방법으로는 알 수 없다는 두 줄에 불과한 포기에 가까운 선언이었다.

이렇게 엄청남 과거사가 있은 줄로 모르고 대한민국의 한 농부인 필자는 “상 전환으로 동절기 엽경 미나리 생산” 에 도전한 것이었다.

한 광고사 광고방법이 “순간의 선택이 10년을 좌우한다.” 와 같이 필자의 이런 도전이 “생체시계진자 vector"라는 탄생을 낳았다.

“전통적인 방법으로는 알 수 없다는 두 줄에 불과한 포기에 가까운 선언” 이 생체시계진자 vector로 해결을 본 셈이라고 할 수 있다.

한국의 근대사는 그런 가운데에서도 한국의 빨리 빨리 혼은 30년의 짧은 기간 동안 세계사에서 드문 경제 문화 등 모든 부분에서 발전이 있었고 그 중 특히 반도체산업은 세계선두의 첨단반도체국가의 입지를 확보하게 되었다.

다행이 반도체는 세계 모두가 초기에 해당하는 신흥 산업이라는 출발라인이 동일했기 때문이다.

필자가 “차세대과학은 광역동시다발정밀연구” 라고 할 때 더 빠르고 더 정밀하고 정확한 정보체계는 한글 자소에 의한 각종 미들웨어, 시스템프로그램, 응용프로그램은 물론 소프트웨어가 구축되어 광역동시다발정밀연구에 필요한 정밀한 기기나 기구를 개발할 한국과학미래는 희망적이라 할 수 있다.” 라는 주장이요 예언이다.

앞으로 생물과학은 인간의 미래가 달린 생명과학으로 화석에너지의 고갈에 따라 생활에너지인 탄소수소화합물에너지까지 식량에너지 생산 방법으로 해결해야하기 때문에 광합성의 해법과 신물질 생산 과학은 중요한 위치에 놓이게 된다.

인간이 개발하는 과학 중에서 제일 난해하고 개발속도가 느린 부분의 과학이 생물과학이라 할 수 있다.

그리고 가장 고급 엘리트들이 참여해야 하는 부분의 과학이 생물과학분야이나 투자에 비해 환원 속도가 느리다는 이유로 도외시 된 부분이다.

| 벼 출수생리(벼의 생리와 생태, 조 동삼 외 14 향문사 1995.04.10 p56 표1-10) | | | | |
|---|---|---|---|---|

| 품종 | 파종기 (월/일) | 출수일수 | | | 주간총엽수 | | | 출엽간격일 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | A | B | C | A | B | C | A | B | C |
| 오대벼 | 12/28 | | (3.12)75 | (3.01) 64 | 12.2 | 10.5 | 10.5 | 12.7 | 6.9 | 6.0 |
| | 2/1 | | (4.17)76 | (3.30) 58 | 13.3 | 11.0 | 11.0 | | | |
| | 3/5 | (8.12)160 | (5.13)70 | (4.28) 55 | 14.0 | 11.0 | 12.0 | | | |
| | 4/2 | (8.27)148 | (6.27)87 | (6.01) 61 | 14.0 | 12.7 | 12.3 | | | |
| | 4/30 | (10.01)155 | (7.14)76 | (7.01) 63 | 12.8 | 13.0 | 13.0 | | | |
| 동진벼 | 12/28 | | (2.24)59 | (2.14) 49 | 14.0 | 10.2 | 10.2 | 12.8 | 8.9 | 8.8 |
| | 2/1 | | (3.31)59 | (3.20) 48 | 14.0 | 9.3 | 9.3 | | | |
| | 3/5 | (8.24)173 | (5.10)67 | (4.19) 46 | 15.0 | 17.3 | 11.0 | | | |
| | 4/2 | (9.16)168 | (8.1)122 | (7.27)117 | 15.0 | 16.7 | 19.3 | | | |
| | 4/30 | (10.04)158 | (8.11)104 | (7.21) 83 | 15.3 | 15.3 | 15.3 | | | |
| 주/이 온도A = 20/15℃, B = 25/20℃, C=30/25℃ ※( )속의 수는 출수월일을 나타내며 M S brothers가 일력에서 환산한 것임) | | | | | | | | | | |

위 표와 사진은 벼의 이삭이 나오는 시기인 출수시험과 들깨의의 꼬투리가 나타나는 화뢰 출현 시기를 인공기상실에서 조사한 연구이다.

광역동시다발정밀연구에 해당하는 한 예라고 볼 수 있다.

필자가 쓴 “지구가 흔들리기 때문에 생체시계가 돌아간다.” 에서 인용한 것으로 계절에 민감한 벼와 들깨의 위상이 달라지는 것을 관찰한 것이다.

생물 산업은 과거의 농경과는 다른 새로운 가치산업으로 발전하지 않으면 고갈된 자원에다 1인당에너지사용량이 늘어난 시대에 필요한 소비량을 충당하지 못하는 경우가 발생한다.

특히 아프리카와 같은 지역은 질병과 영양부족으로 시달리고 있다는 사실이다.

차세대과학은 첨단반도체산업과 인력문화1위 강국 한국에 의해 새로운 출발을 할 수밖에 없는 것으로 그만큼 한국입지가 중요하다.

나노(nano-)와 기가(giga-)수준은 차세대과학으로서는 걸음마 단계로 보아야 한다.

펨토(femto-)와 페타(peta-)를 넘어 아토(atto-) 엑사(exa-)수준의 더 빠르고 더 정밀하며 정확한 정보체계로 된 세계가 첨단반도체와 인력문화1위 강국의 대한민국을 기다리고 있다는 것을 알아야 한다.

세계 총인구 70억의 0.7%에 해당하는 겨우 5,000만 한국인이라 할지라도 사회적인 기능과 역할은 매우 크다는 것이다.

첨단반도체대국 한국 !

경제대국 한국 !

문화대국 한국 !

인력문화 1위 강국 한국 !

이는 우연이 아닌 필연이다.

자소 51개로

**11,172개 글자 값과 조립글자 11,172개, 11,172개 발성기호를 동시에 입력되는 반도체 기기최적화를 사용하는 정보화시대 한글을 사용하는 한국은**

“바로입력자소 조립자소”

이점 때문에 더 빠르고 더 정밀하고 더 정확한 정보의 주역이 되어 세계를 이끌고 지배할 것이다.

**2 × 10□⁴ =115,281,504,606,846,976의 엑사(exa-)과학이 기다린다.**

20 맺는 말

1996년 6월 유니코드 2,0작업에서 11,172자가 배정되고 2012년 9월 27일 유니코드 6.2의 작업까지 끝나는 동안 웹상에서 한글을 사용하는데 이상은 없는 것으로 되어 있다.

**유니코드 2.0이 제정되기 전까지 무수한 논쟁이 계속 되었지만 한글에서 글자가 정확하게 몇 자인지 모르고 있었다는 실로 어처구니없는 일이 있어왔던 것과** 반도체기기에서 중요한 정보 처리를 위해서는 **자소단위**로 처리된다는 개념을 모르고 있었다는 웃지 못 할 지난날도 있었다.

**반도체 기기에서는 충분한 글자 값이 확보되고 사용할 글자의 자소 값이 배정되어지는 것이 우선적으로 해결 되어야 호환성의 운영체제가 해결된다.**

**뒤 늦게 발견된 것으로 옛한글까지 자모 240자(초성 90자, 중성 66자, 종성 82자, 채움 2자)를 사용하게 된다면 이론적으로 한글은 약 50만 자가 된다는 것이다.**

현대어로 사용되어야 하는 글자의 자소단위로 필요한 값은 글자 11,172개와 자소 51개를 합하여 11,223개이다.

유니코드에서 현재의 $2^1$ □ = 65,536에서 2바이트 8계단수의 값에 ① 기본 다국어 평면 BMP, ② 보조 다국어 평면 SMP, ③ 보조 상형 문자 평면 SIP, ④ 3차 상형 문자 평면 TIP, ⑤ 보조 특수 목적 평면 SSP 로 처리하고 있다.

**그 이전 A라는 회사에서 사용하던 한글 문서(코드)는 B라는 회사 컴퓨터에서는 사용할 수 없고, 역으로 B에서 사용하던 한글 문서(코드)는 A라는 회사에서 사용할 수 없게 되어 버린 것이다. 이와 같은 한글 코드간의 호환성이라는 문제가 대두되면서 나름대로 노력을 시도하였지만 별다른 효과가 없었다. 이때까지만 해도 소프트웨어는 하드웨어를 팔기 위한 하나의 수단이었으므로 일단 컴퓨터를 많이 팔아서 세력을 얻는 방법이 효과적이라고 여겼기 때문에 해결이라는 것은 애당초 생각조차 하기 힘들었던 시기였다. 시간이 지날수록 불편함이 가중되었고 그만큼 사용자들의 표준화 요구도 더욱 절실해졌다. 이러한 문제를 해결하기 위해서 업체 중심으로 한글 코드 표준화를 시도하게 되었지만 앞서와 같이 한글을 자소단위가 아닌 조합형이니 완성형이니 하여 다루면 결국 호환성이 없게 된다.**

국제 표준화 작업에 대한 충분한 대비를 하지 않고 있다가 "소 잃고 외양간 고치는 격"으로 뒤늦은 연구 작업을 하였기 때문에 우리의 잘못이 라고 자책하고 있으나 이는 자책의 문제가 아니고 반도체기기의 특성을 전혀 모르고 있었다는 것이다.

반도체기기에서 정보단위로서 글자는 자소단위라는 것을 인식하고 자소단위에는 낱개 자소(낱 자소)와 조립자소가 있고, 중국의 글자는 비록 조립자소라고 하드라도 조립의 구성자소가 277개가 되어 정보입력방식이 바로입력 되는 바로자소가 아닌 전환자소라는 개념만 도입되었더라도 제작자간의 다툼은 없었을 것이다.

필자가 2012년 분류한 반도체기기의 적성으로 한글이 초성, 중성, 종성 자소로 된 수직수평의 조립자소이자 바로자소라는 것을 발견한 토대를 축으로 한글은 현재와 미래를 위해 어떠한 모습을 갖추어야 하는지에 대한 결론은

①. 글자 값으로 한글 원리를 반영하고 31비트 수준이면 표현 가능한 문자는 모두 처리할 수 있어서 옛한글도 현대어처럼 다룰 수 있어야 하고, 비 한글 문자(한자, 영문자, 일 문자, 기타 문자)도 사용하는 데 문제가 없어야 한다.

②. 한국어 정보 처리를 위해서는 음소 단위로 처리할 수 있어야 한다.

③. 다른 프로그램과 충돌이 없어야 한다.

④. 처리된 자료 및 기술은 산업 현장, 학계, 정부 기관에서 공유하도록 공개적이어야 한다.

⑤. 국제적인 표준 규격을 따라야 한다.

⑥. 정부의 적극적 정책지원이 있어야 한다.

이를 토대로 반도체기기운영체제와 기계언어를 개발해야 하는 시대가 왔다는 것을 인식하고 대한민국의 정부와 국민의 적극적이고 아낌없는 투자지원이 이루어져야 할 것이다.

반도체시대의 정보속도는 곧 경쟁력이자 자산이다.

더 빠르고 더 정밀하고 더 정확한 정보능력이야말로 가장 값비싼 가치의 경쟁력을 갖춘 국가의 자산이다.

# : 한글 인코딩의 역사와 유니코드

개발자Tip 2012.02.29 16:41

| | |
|---|---|
| 醫떵븴□□/button〉<br>醫떵븴□□/div〉 | 1918 |

NHN Business Platform 쇼핑서비스개발팀 오영은

분명 제대로 보이는 한글 이름의 파일을 내려 받았는데 읽을 수 없는 이상한 이름으로 저장된 파일을 받아본 경험이 있을 것입니다. 보통 '인코딩이 깨졌다.'라고 말하는 이런 상황은 왜 발생하는 것일까요? 그 이유는 컴퓨터에서 한글을 표현하는 다양한 방식이 있는데 해당 방식이 서로 맞지 않기 때문입니다.

최초로 컴퓨터가 발명되고 오랜 기간 동안 발전되어 온 지역이 미국이기에 해당 지역에서 사용하는 언어의 문자 집합인 영어 알파벳과 이와 비슷한 문자 체계를 지닌 유럽어 알파벳 처리에 대한 연구가 가장 먼저 시작되었습니다. 이 외의 다른 문자 집합(character set)은 기존에 수립된 인코딩(영어 및 유럽어 문자 집합용)으로 표현하기에는 한계가 있었기 때문에 이들을 처리하기 위한 연구가 추가로 진행되었습니다.

**요즘은 어느 전자 기기에서나 한글을 제대로 입력할 수 있고 일부 소형 기기에서는 한글을 더 빠르게 입력할 수도 있어 컴퓨터에서 한글을 처리하는 작업이 너무나 쉽고 당연하게 받아들여지고 있습니다. 하지만 한글을 제대로 표현하기 위한 한글 인코딩 체계가 수립되기까지는 수십 년의 세월이 걸렸습니다.**

현재 우리나라에서 주로 사용하고 있는 CP949 또는 EUC-KR(둘은 엄밀히 다릅니다) 인코딩과 유니코드를 제대로 이해하기 위해서는 한글을 표현하기 위한 그간의 역사를 알 필요가 있습니다. 2편 연작으로 기획된 이 기사의 1편에서는 한글 인코딩의 역사를 다루고, 2편에서는 'Java 언어를 기준으로 한글을 처리하는 방법'을 다루도록 하겠습니다.

## 문자 집합과 인코딩

컴퓨터는 수치 연산을 위해 설계되었다. 컴퓨터 발명 초기에는 문자를 표현해야 하는 요구가 없었다. 영어 단어 'compute'는 단순히 '계산하다'라는 뜻이고, 초창기의 컴퓨터와 '전자계산기'는 동의어이기도 했다. 그러나 (너무나 당연하지만) 문자를 표현해야 하는 요구가 발생했다. 컴퓨터 간에 문자 데이터를 교환해야 할 일이 생기기도 했다. 이기종 컴퓨터끼리 문자 데이터를 교환하기 위해서는 표준이 필요하다. 이런 이유로 ASCII(American Standard Code for Information Interchange)와 같은 표준 문자 인코딩이 만들어졌다.

문자를 표현하기 위해서는 가장 먼저 '문자 집합'을 정의해야 한다. 문자 집합은 표현해야 할 문자를 정하고 순서를 지정한 것이다. 영어라면 'A', 'B', 'C'에서 'Z'까지(소문자 a에서 z), 한글이라면 '가', '각', '간'에서 '힣'까지다. 물론 숫자나 특수 문자뿐만 아니라 인쇄와 통신을 제어하기 위한 제어 문자도 문자 집합에 포함되어야 한다. 이러한 문자 집합을 코드 형태(일반적으로 행렬)로 표기한 것을 코드화된 문자 집합(CCS, coded character set)이라고 한다. 예를 들어 '가'에는 10001, '각'에는 10002와 같이 코드를 할당하는 방식 말이다. 그리고 문자 집합을 컴퓨터에 저장하기 위해서 옥텟(octet, 8비트 단위) 형태로 표현한 것을 인코딩 방식(CES, character encoding scheme)이라고 한다.

## 영어의 문자 집합과 인코딩

최초의 컴퓨터인 ENIAC(Electronic Numerical Integrator And Computer)이 만들어진 이후 약 15년이 지나서야 문자 집합이라는 개념이 생겼다. 그 시초는 ASCII이다. ASCII는 0x00부터 0x7F까지의 총 127개 문자(제어 문자, 특수 문자, 숫자, 알파벳 등)로 이루어져 있다. 이는 미국에서 제정된 표준이니 영어 알파벳을 표현하기에는 큰 문제가 없었겠지만, '□'와 같은 문자를 표현할 수 없어 유럽어에는 사용할 수 없었다. 이를 해결하기 위해 확장 ASCII(Extended ASCII)를 제정하여 기존의 ASCII로 정의하지 못했던 128번부터 255번까지의 새로운 문자를 정의할 수 있게 되었다. 즉, 새로 추가된 128개의 코드(0x80 ~ 0xFF)로 프랑스어, 독일어 등의 유럽어를 표현할 수 있게 된 것이다. 이와 같이 다양한 유럽어를 표현할 수 있는 확장 ASCII는 ISO-8859 유럽 통일 표준안으로 제정되었다. (표준안의 ISO 버전은 언어에 따라 약간씩 다르다. 예를 들어 ISO 8859-1은 서유럽 언어를, ISO 8859-2는 동유럽 언어를 표현하기 위한 표준안이다.)

한글의 문자 집합과 인코딩

영어나 유럽어는 알파벳을 기초로 사용하므로 256개의 코드를 이용하여 문자 집합을 표현하는 것이 가능하다. 하지만 **중국, 일본, 한국(CJK, Chinese-Japanese-Korean)에서 사용하는 문자 집합인 한글, 한자, 가나, 병음, 주음부호 등은 그 개수가 많아 확장 ASCII 코드로도 이를 모두 처리할 수 없다. 따라서 CJK 문자를 처리하기 위한 별도의 방안이 필요하다.**

그렇다면 컴퓨터에서 한글을 표현하는 방법에는 무엇이 있을까? 한글 표현 방법은 크게 조합형과 완성형으로 나눌 수 있으며, 이를 좀 더 세분화하면 N바이트 조합형, 3바이트 조합형, 7비트 완성형, 2바이트 조합형, 2바이트 완성형으로 나눌 수 있다. 조합형이란 한글의 제자 원리에 기반 하여 초성, 중성, 종성에 각각 코드를 할당하는 방식이고, 완성형이란 '가', '각', '간'과 같은 완성된 문자에 코드를 할당하는 방식이다. 이 중 완성형이 한글 표준안으로 채택되었고, 따라서 유니코드의 한글 표현 방식에도 완성형이 먼저 채택되었다.

Microsoft Windows 95 이전, 본격적으로 한글 문자 집합과 한글 글꼴이 운영체제에 포함되기 전인 DOS(Disk Operating System) 시절에는, 결과를 출력하기 위해 BIOS(Basic Input/Output System)를 직접 제어하거나 '한글 카드'라 불리는 ISA(Industry Standard Architecture) 인터페이스와 같은 별도의 하드웨어를 사용해야 했다.

N바이트 조합형

유럽어만 표시할 수 있었던 컴퓨터에서 한글을 표시하기 위한 고육지책으로 고안된 최초의 한글 표현 방식이다. 이 방식은 대형 컴퓨터에서 단말기를 이용하여 한글을 표현하는 데 사용되었다. N바이트 조합형은 각각의 개별적인 한글 자모를 영문자 하나하나에 대응시키고, 시작과 끝에 SI(Shift In)와 SO(Shift Out)을 추가하여 한글과 영어를 구분하는 방식이다. 예를 들어, '한글'은 ^N^bDAzl^O로 표현할 수 있다(^N은 SI이고 ^O는 SO임).

다음 표는 N바이트 조합형의 한글/알파벳 매핑 규칙을 보여준다.

| 자음 | | 모음 | |
|---|---|---|---|
| ㄱ | A | ㅏ | b |
| ㄲ | B | ㅐ | c |
| ㄴ | D | ㅑ | d |
| ㄷ | G | ㅒ | e |
| ㄸ | H | ㅓ | f |
| ㄹ | I | ㅔ | g |
| ㅁ | Q | ㅕ | j |
| ㅂ | R | ㅖ | k |
| ㅃ | S | ㅗ | l |
| ㅅ | U | ㅛ | r |
| ㅆ | V | ㅜ | s |
| ㅇ | W | ㅠ | w |
| ㅈ | X | ㅡ | z |
| ㅉ | Y | ㅣ | \| |
| ㅊ | Z | | |
| ㅋ | [ | | |
| ㅌ | ₩ | | |
| ㅍ | ] | | |
| ㅎ | ^ | | |

3바이트 조합형

초성, 중성, 종성에 1바이트씩 할당하여 사용하는 방식이다. 3바이트 조합형은 80년대 개인용 컴퓨터(PC)에서 사용되었다. 단지 중성과 종성

이 없는 글자를 위해 채움 문자(fill code)를 정의하고 있다. N바이트 조합형이 한글을 2바이트에서부터 5바이트로 표현하는 데 반해, 3바이트 조합형은 3바이트로 표현하며 이는 KS X 1001 표준안의 한글 채움 문자 방식과 유사하다.

7비트 완성형

세운상가에서 만들었다고 하여 청계천 한글이라고도 한다. 소문자 뒤에 대문자가 오는 경우가 거의 없고, 특수 문자 뒤에 영문자가 오는 경우가 거의 없다는 점에 착안하여 고안한 방식이다. 즉, 처음 1바이트가 소문자이거나 특수 문자이고, 그 다음 1바이트가 대문자이면 한글로 표현한다. 글자 표현이 1,300여자 정도로 제한된다는 점과 일부 영어 단어가 한글로 표시(1990년대 초반 자주 사용되던 'dBASE'라는 프로그램 이름이 '늦ASE'로 표시)되는 문제가 있다.

표현 글자 수가 적었지만, 7비트 완성형은 당시 많이 사용했던 허큘리스 그래픽 카드를 확장하여 만들어진 기능이기 때문에 과거 80~90년대의 컴퓨터에서 작업 처리 속도를 지연하지 않는 유일한 방식이었다. 이런 이유로 당시에는 어느 정도의 수요가 있었다.

2바이트 조합형

초성, 중성, 종성에 각각 5비트씩 할당하고, 처음 1비트(MSB, most significant bit)는 1로 설정하여 한글임을 표시하는 방식이다. 대우, 삼보 등의 여러 회사에서 각각의 조합형 방식을 제안하여 사용하였으며, 나중에는 삼보 컴퓨터가 주도한 상용 조합형(KSSM)이 표준처럼 사용되었다. 예를 들어 '한글'은 1 10100(ㅎ) 00011(ㅏ) 00101(ㄴ) 1 00010(ㄱ) 11011(ㅡ) 01001(ㄹ) (0xD0, 0x65, 0x8B, 0x69)로 표현된다.

다음 표는 2바이트 조합형의 한글 매핑 규칙을 보여준다.

| 비트 조합 | 초성 | 중성 | 종성 |
|---|---|---|---|
| 00001 | 채움 | | 채움 |
| 00010 | ㄱ | 채움 | ㄱ |
| 00011 | ㄲ | ㅏ | ㄲ |
| 00100 | ㄴ | ㅐ | ㄳ |
| 00101 | ㄷ | ㅑ | ㄴ |
| 00110 | ㄸ | ㅒ | ㄵ |
| 00111 | ㄹ | ㅓ | ㄶ |
| 01000 | ㅁ | | ㄷ |
| 01001 | ㅂ | | ㄹ |
| 01010 | ㅃ | ㅔ | ㄺ |
| 01011 | ㅅ | ㅕ | ㄻ |
| 01100 | ㅆ | ㅖ | ㄼ |
| 01101 | ㅇ | ㅗ | ㄽ |
| 01110 | ㅈ | ㅘ | ㄾ |
| 01111 | ㅉ | ㅙ | ㄿ |
| 10000 | ㅊ | | ㅀ |
| 10001 | ㅋ | | ㅁ |
| 10010 | ㅌ | ㅚ | |
| 10011 | ㅍ | ㅛ | ㅂ |
| 10100 | ㅎ | ㅜ | ㅄ |
| 10101 | | ㅝ | ㅅ |
| 10110 | | ㅞ | ㅆ |
| 10111 | | ㅟ | ㅇ |
| 11000 | | | ㅈ |
| 11001 | | | ㅊ |
| 11010 | | ㅠ | ㅋ |

| | | |
|---|---|---|
| 11011 | ㅡ | ㅌ |
| 11100 | ㅢ | ㅍ |
| 11101 | ㅣ | ㅎ |

2바이트 완성형

완성된 음절을 코드와 일대일 대응시키는 방식이다. 예를 들어, '가'는 0xB0A1, '각'은 0xB0A2로 코드화한다. ISO 2022 표준을 기준으로 하였으며, KS C 5601:1987 표준안으로 채택되었다. ISO 2022 표준은 ASCII 영역과 겹치지 않도록 첫 번째 비트 값을 1로 규정하였으므로, KS C 5601 표준안은 0xA1A1부터 0xFEFE까지의 영역(94x94, 8,836글자)만을 사용하였다. 더군다나 8,836개의 글자 중에서 부호 및 일본, 러시아 글자에 1,598자를, 한자에 4,888자를 할당하여 한글에는 2,350자밖에 사용할 수 없었다. 2바이트 완성형의 문제는 1990년에 방영된 MBC 드라마 제목 '똠방각하'를 표현하지 못하면서 불거졌는데, 완성형으로는 '똠'을 표현할 수 없어 한글의 제자 원리를 무시한 방식이라는 비난을 피할 수 없게 되었다.

이러한 문제를 해결하기 위해 1992년에 KS C 5601 표준안에 완성형/조합형을 모두 표준으로 지정하였으나, 표준 조합형은 기존에 사용되던 상용 조합형(KSSM)과 코드가 맞지 않아 거의 사용되지 않았다. 1990년대 초반까지 조합형과 완성형이 모두 사용되었지만, 국가 주도의 프로그램, 운영체제에서 완성형을 기본으로 지원하였고, Microsoft Windows 95에서 확장 완성형이 사용됨으로써 조합형은 사장되었다. 현재는 KS C 5601은 KS X 1001로, KS C 5636은 KS X 1003으로 변경되었다. KS X 1003 표준안은 ASCII와 동일하나, 역슬래시(₩)가 원화(₩)로 표기되는 것만 다르다.

확장 완성형

Microsoft가 독자적으로 제정한 문자 집합으로 완성형 코드에서 표현할 수 없던 8,822자가 추가되었다. 통합형 한글 코드(UHC, Unified Hangul Code)라고도 하며, 현대 한글을 모두 표현할 수 있다. 그러나 완성형 영역의 2,350자는 자음, 모음 순서대로 배열되어 정렬에 문제가 없었으나, 확장 완성형의 문자 정렬에는 문제가 있었다.

한글의 인코딩 방식

EUC-KR은 KS X 1001과 KS X 1003 표준안의 인코딩 방식이며, CP949(MS949, x-windows-949)는 확장 완성형의 인코딩 방식이다. 그러므로 EUC-KR은 2,350자의 한글, CP949는 11,172자의 한글을 표현할 수 있다. 그러나 Java에서는 CP949와 MS949를 다르게 취급한다. CP949는 IBM에서 처음 지정한 코드 페이지(sun.nio.cs.ext.IBM949)가 기준이고 Microsoft가 제정한 확장 완성형은 MS949(sun.nio.cs.ext.MS949)를 기준이다. 그러므로 Java에서는 CP949와 EUC-KR이 사실상 같으며, 확장 완성형을 사용하기 위해서는 MS949로 지정해야 한다.

유니코드

한글뿐 아니라 일본어와 중국어에도 컴퓨터에서 해당 언어를 표현할 수 있는 독자적인 문자 집합이 있다(KPS-9566, EUC-CN, EUC-TW, EUC-JP, Shift JIS, Big5, GB, HZ 등). 문제는 '어떻게 동시에 한국어, 중국어, 일본어를 표현하느냐'이다. 하나의 문자 집합을 사용하는 문서에서는 이를 동시에 표현할 수 없다(escape sequence를 이용하여 여러 문자 집합을 표현할 수 있으나 이는 널리 쓰이지 않았다).

이런 문제는 유럽어의 문자 집합에도 있었다. 유로화를 나타내는 '€' 기호에는 ISO 8859-15(Latin 9)의 코드 값 중 0xA4이 할당되었으나 ISO 8859-1(Latin 1)의 0xA4 코드에 할당된 문자는 '¤'다. 이 문제를 해결하기 위해 전 세계적으로 사용되는 모든 문자 집합을 하나로 모아 탄생시킨 것이 유니코드이다. ISO 10646 표준에서 UCS(Universal Character Set)을 정의하고 있다. 유니코드 1.0.0은 1991년 8월 제정되었으며, 그 후 약 5년이 지나서야 유니코드 2.0.0에 한글 11,172자가 모두 포함되었다. 현재 버전은 2010년 10월 11일 제정된 6.0이다.

유니코드 값을 나타내기 위해서는 코드 포인트(code point)를 사용하는데, 보통 U+를 붙여 표시한다. 예를 들어, 'A'의 유니코드 값은 U+0041로 표현한다(₩u0041로 표기하기도 함). 유니코드는 공식적으로 31비트 문자 집합이지만 현재까지는 21비트 이내로 모두 표현이 가능하다. 유니코드는 논리적으로 평면(plane)이라는 개념을 이용하여 구획을 나누며, 평면 개수는 0번 평면인 기본 다국어 평면(BMP; Basic Multilingual Plane)에서 16번 평면까지 모두 17개이다. 대부분의 문자는 U+0000~U+FFFF 범위에 있는 기본 다국어 평면에 속하며, 일부 한자는 보조 다국어 평면(SMP, Supplementary Multilingual Plane)인 U+10000~U+1FFFF 범위에 속한다. 이 중 한글은 U+1100~U+11FF 사이에 한글 자모 영역, U+AC00~U+D7AF 사이의 한글 소리 마디 영역에 포함된다(자세한 내용은 위키백과의 '유니코드 목록' 내용 참조).

유니코드 인코딩 방식

유니코드의 인코딩 방식으로는 코드 포인트를 코드화한 UCS-2와 UCS-4, 변환 인코딩 형식(UTF, UCS Transformation Format)인 UTF-7, UTF-8, UTF-16, UTF-32 인코딩 등이 있다. 이 중 ASCII와 호환이 가능하면서 유니코드를 표현할 수 있는 UTF-8 인코딩이 가장 많이 사용된다. UTF-8은 코드 포인트 범위에 따라 다음 표에서 보는 바와 같이 인코딩 방식이 다르다.

다음 표는 코드 포인트 범위에 따른 UTF-8 인코딩 방식을 보여준다.

| 코드 포인트 범위 | 비트 수 | 인코딩 |
|---|---|---|
| U+0000~U+007F | 7 | 그대로 인코딩 |
| U+0080~U+07FF | 11 | 110xxxxx 10xxxxxx |
| U+0800~U+FFFF | 16 | 1110xxxx 10xxxxxx 10xxxxxx |
| U+10000~U+1FFFFF | 21 | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx |

위의 표에서 xxxx로 표시된 부분에는 원래의 비트 값을 순서대로 적는다. 즉, U+0080을 비트 값으로 표현하면 000 1000 0000인데, 인코딩 방식에 의해 11000010 10000000으로 변환되어, 0xC2 0x80으로 저장된다. 예를 들어, '한글'을 코드 포인트로 표현하면 U+D55C U+AE00인데, 이를 UTF-8 인코딩하면, 0xED 0x95 0x9C 0xEA 0xB8 0x80이 된다.

U+D55C U+AE00

1101 0101 0101 1100 1010 1110 0000 0000 2진수 표현

1110 1101 1001 0101 1001 1100 1110 1010 1011 1000 1000 0000 인코딩 방식에 따라 인코딩

한글 완성형의 코드 포인트 범위는 U+AC00~U+D7AF이므로, UTF-8 인코딩에서 한글은 무조건 3바이트 인코딩이다. 그래서 URL에 파라미터 값이 %ED%95%9C%EA%B8%80과 같이 표시된다면 UTF-8 인코딩일 확률이 높다(ISO8859, EUC-KR, UTF-8 인코딩 중 하나라면 말이다).

이번 기사에서는 프로그램 언어에서 한글을 제대로 처리하기 위해 알아야 할 기본 지식을 알아보았다. 다음 편 기사에서는 유니코드로 한글을 표현하는 방법과 Java에서 한글을 처리하는 방법에 대해서 다루고자 한다.

## 부록 3, 유니코드와 JAVA를 이용한 한글 처리

NHN Business Platform 쇼핑서비스개발팀 오영은

참고

이 글은 "한글 인코딩의 이해 1편: 한글 인코딩의 역사와 유니코드"에 이어지는 글로, 월간 "마이크로소프트웨어" 2012년 5월호에 "유니코드와 JAVA를 이용한 한글 처리"라는 제목으로 실렸습니다.

encoding 부호화

scheme [ski : m] n.① 계획, 기획, 설계. [SYN.] □ PLAN.② 획책, 책략, 음모; 비현실적인 계획.③ 짬, 조직, 기구.④ 일람표, 도표(schema), 도식(圖式), 도해.⑤ 개요, 대략; 요강.⑥ 약도; 지도.

프로그래밍에서 문자열을 다루다 보면, 여러 인코딩 방식(CES, Character Encoding Scheme)을 마주하게 된다. 특히 한글은 문자집합(CCS, Coded Character Set)의 구성에 따라 조합형, 완성형, 유니코드 등으로 나누어지는데, 본 문서에서는 유니코드에서 한글을 어떻게 표현하는지, 그리고 Java와 같은 언어에서 애플리케이션을 개발할 때 어떻게 한글을 처리하면 좋은지 알아보려 한다.

유니코드란

유니코드란 전 세계적으로 사용하는 모든 문자 집합을 하나로 모은 것이다. 유니코드 1.0.0은 1991년 8월 제정되었으며, 그 후 약 5년이 지나서야 유니코드 2.0.0에 한글 11,172자가 모두 포함되었다. 현재 버전은 2010년 10월 11일 제정된 6.0이다.

유니코드 값을 나타내기 위해서는 코드 포인트(code point)를 사용하는데, 보통 U+를 붙여 표시한다. 예를 들어, 'A'의 유니코드 값은 U+0041로 표현한다(₩u0041로 표기하기도 함). 유니코드는 공식적으로 31비트 문자 집합이지만 현재까지는 21비트 이내로 모두 표현이 가능하다. 유니코드는 논리적으로 평면(plane)이라는 개념을 이용하여 구획을 나누며, 평면 개수는 0번 평면인 기본 다국어 평면(BMP; Basic Multilingual Plane)에서 16번 평면까지 모두 17개이다. 대부분의 문자는 U+0000~U+FFFF 범위에 있는 기본 다국어 평면에 속하며, 일부 한자는 보조 다국어 평면(SMP, Supplementary Multilingual Plane)인 U+10000~U+1FFFF 범위에 속한다. 이 중 한글은 U+1100~U+11FF 사이에 한글 자모 영역, U+AC00~U+D7AF 사이의 한글 소리마디 영역에 포함된다.

유니코드의 인코딩 방식

유니코드의 인코딩 방식으로는 코드 포인트를 코드화한 UCS-2와 UCS-4, 변환 인코딩 형식(UTF, UCS Transformation Format)인 UTF-7, UTF-8, UTF-16, UTF-32 인코딩 등이 있다. 이 중 ASCII와 호환이 가능하면서 유니코드를 표현할 수 있는 UTF-8 인코딩이 가장 많이 사용된다. UTF-8은 코드 포인트 범위에 따라 다음 표에서 보는 바와 같이 인코딩 방식이 다르다. 다음 표는 코드 포인트 범위에 따른 UTF-8 인코딩 방식을 보여준다.

표1. 유니코드 범위 목록에서의 한글 관련 범위

| 코드 포인트 범위 | 비트 수 | 인코딩 |
|---|---|---|
| U+0000 ~ U+007F | 7 | 그대로 인코딩 |
| U+0080 ~ U+07FF | 11 | 110xxxxx10xxxxxx |
| U+0800 ~ U+FFFF | 16 | 1110xxxx10xxxxxx10xxxxxx |
| U+10000 ~ U+1FFFFF | 21 | 11110xxx10xxxxxx10xxxxxx10xxxxxx |

위의 표에서 xxxx로 표시된 부분에는 원래의 비트 값을 순서대로 적는다. 즉, U+0080을 비트 값으로 표현하면 000 1000 0000인데, 인코딩 방식에 의해 11000010 10000000으로 변환되어, 0xC2 0x80으로 저장된다. 예를 들어, '한글'을 코드 포인트로 표현하면 U+D55C U+AE00인데, 이를 UTF-8 인코딩하면, 0xED 0x95 0x9C 0xEA 0xB8 0x80이 된다.

U+D55C U+AE00

11010101 01011100 10101110 00000000 2진수 표현

11101101 10010101 10011100 11101010 10111000 10000000 인코딩 방식에 따라 인코딩

유니코드에서 한글을 표현하는 방법

유니코드 범위 목록(Mapping of Unit characters)을 살펴보면, 한글 표현을 위한 코드 영역 개수는 다른 언어 글자를 위한 코드 영역 개수보다 대체로 많다는 것을 알 수 있다. 유니코드에서 한글을 표현하기 위한 코드 영역은 다음과 같다.

표2 유니코드 범위 목록에서의 한글 관련 범위

| 이름 | 처음 | 끝 | 개수 |
|---|---|---|---|
| 한글 자모 (Hangul Jamo) | 1100 | 11FF | 256 |
| 호환용 한글 자모 (Hangul Compatibility Jamo) | 3130 | 318F | 96 |
| 한글 자모 확장 A (Hangul Jamo Extended A) | A960 | A97F | 32 |
| 한글 소리마디 (Hangul Syllables) | AC00 | D7AF | 11184 |
| 한글 자모 확장 B (Hangul Jamo Extended B) | D7B0 | D7FF | 80 |

한글 소리마디

초성, 중성, 종성으로 이루어진 한글을 표현하기 위한 영역이다. 현대 한글에서 표현 가능한 11,172자를 모두 포함하고 있다. 완성형 한글처럼 각 음절(소리마디)마다 코드를 매핑 하는 방식이다. 범위는 '가'(U+AC00)부터 '힣'(U+D7A3)까지이다. 그림1에서 볼 수 있듯이, 각 음절은 초성, 중성, 종성 순으로 정렬되어 있으며, 확장 완성형과 다른 점은, 초성, 중성, 종성의 분리가 가능하다는 점이다.

그림 1 유니코드에서의 한글 소리마디

그림 1에 나와 있듯이 초성은 19개, 중성은 21개, 종성은 28개로 이루어져 있다. 초성, 중성, 종성에 대한 값은 다음 표3에서 알 수 있다.

표 3 한글 소리마디에서 초성/중성/종성에 대한 순서 값

| 값 | 초성 | 중성 | 종성 | 값 | 초성 | 중성 | 종성 |
|---|---|---|---|---|---|---|---|
| 0 | ㄱ | ㅏ | 채움 | 14 | ㅊ | ㅜ | ㄼ |
| 1 | ㄲ | ㅐ | ㄱ | 15 | ㅋ | ㅝ | ㄽ |
| 2 | ㄴ | ㅑ | ㄲ | 16 | ㅌ | ㅞ | ㄾ |
| 3 | ㄷ | ㅒ | ㄳ | 17 | ㅍ | ㅟ | ㄿ |
| 4 | ㄸ | ㅓ | ㄴ | 18 | ㅎ | ㅡ | ㅀ |
| 5 | ㄹ | ㅔ | ㄵ | 19 | | ㅢ | ㅅ |
| 6 | ㅁ | ㅕ | ㄶ | 20 | | ㅣ | ㅆ |
| 7 | ㅂ | ㅖ | ㄷ | 21 | | | ㅇ |
| 8 | ㅃ | ㅗ | ㄹ | 22 | | | ㅈ |
| 9 | ㅅ | ㅘ | ㄺ | 23 | | | ㅊ |
| 10 | ㅆ | ㅙ | ㄻ | 24 | | | ㅋ |
| 11 | ㅇ | ㅚ | ㄼ | 25 | | | ㅌ |
| 12 | ㅈ | ㅛ | ㄽ | 26 | | | ㅍ |
| 13 | ㅉ | ㅜ | ㄾ | 27 | | | ㅎ |

한글 음절의 코드 포인트 값은 시작 값인 U+AC00에 ((초성 값 x 21) + 중성 값) x 28 + 종성 값을 더하면 된다. 예를 들어, '한'이라는 글자는 'ㅎ', 'ㅏ', 'ㄴ'으로 구성되어 있으며, 각각 18, 0, 4 값을 가지고 있으므로, '한'의 코드 포인트 값은 U+AC00 + ((18 x 21) + 0) x 28 + 4 = U+AC00 + U+295C = U+D55C가 된다. 이를 역으로 생각해 보면, 한글 음절에 대해 초성, 중성, 종성의 분리가 가능하다. 즉 한글 음절의 코드 포인트 값에서 U+AC00을 뺀 값을 ①이라 한다면, 다음과 같이 정리할 수 있다.

①의 값을 (21 x 28)로 나눈 몫은 초성
①의 값을 (21 x 28)로 나눈 나머지를, 28로 나눈 몫은 중성

①의 값을 28로 나눈 나머지는 종성

호환용 한글 자모

닿소리와 홀소리 같은 한글 자모를 표현하기 위한 영역이다. 이 영역은 현대 한글에서 사용하는 자음(U+3130 ~ U+314E), 모음(U+314F ~ U+3163), 채움 코드(U+3164), 옛한글 자모(U+3165 ~ U+318E)로 구성되어 있다.

한글 자모, 한글 자모 확장

첫 가 끝 코드라고도 부른다. 한글 자모 확장은 유니코드 버전 조합형 한글로도 이해할 수 있다.

한글 소리마디에서는 완성형 한글을, 호환형 한글 자모에서는 초성/중성/종성 구별 없는 자음과 모음을 표현하는 것과 달리, 한글 자모, 한글 자모 확장에서는 초성/종성을 구별하는 자음과 모음으로 구성되어 있다. 이를 이용하여 조합형 한글을 표현할 수 있다. 또한, 한글 자모 영역에는 옛한글에서만 사용된 초성, 중성, 종성이 있으므로, 옛한글을 표현하는 데 전혀 문제가 없다.

그림 3 한글 자모 코드 영역

U+1100부터 U+115E까지는 초성, U+1161부터 U+11A7까지는 중성, U+11A8부터 U+11FF까지는 종성이다.

유니코드 정규화(Unicode equivalence)

한글 소리마디와 한글자모, 한글 자모 확장 이렇게 두 개의 코드 영역이 있다는 것은 같은 글자를 표현하는 서로 다른 두 개의 방법이 있다는 것을 말한다. 이것은 한글뿐만 아니라 다른 언어에서도 나타나는 현상이다. 가령 "□"을 표현할 때 U+00F1을 사용할 수도 있고, U+006E (라틴 소문자 "n") 과 U+0303( 결합 틸데 "□")을 연이어 사용하여 표현할 수도 있다. 유니코드 정규화(Unicode equivalence)란 이렇게 연속적인 코드를 사용하여 표현한 어떤 글자를 처리하는 방법을 다루는 명세이다. 유니코드 정규화에는 다음과 같은 네 가지 방법이 있다.

표 4 유니코드 정규화 방법과 예

| 정규화 방법 | 예 |
|---|---|
| NFD (정준 분해)<br>Normalization Form Canonical Decomposition | □ (U+00C0) → A (U+0041) + □ (U+0300)<br>위 (U+C704) → □ (U+110B) + □ (U+1171) |
| NFC (정준 분해한 뒤 다시 정준 결합)<br>Normalization Form Canonical Composition | A (U+0041) + □ (U+0300) → □ (U+00C0)<br>□ (U+110B) + □ (U+1171) → 위 (U+C704) |
| NFKD (호환 분해)<br>Normalization Form Compatibility Decomposition | □ (U+FB01) → f (U+0066) + i (U+0069) |
| NFKC (호환 분해한 뒤 다시 정준 결합)<br>Normalization Form Compatibility Composition | 樂 (U+F914), 樂 (U+F95C), 樂 (U+F9BF) → 樂 (U+6A02) |

이중 한글 처리와 관련된 것은 NFD(소리마디를 첫 가 끝 코드로 분해)와 NFC(첫 가 끝 코드를 소리마디로 결합)이다.

```
1   import java.text.Normalizer;
2
3   public class NormalizerTest {
4       private void printIt(String string) {
5           System.out.println(string);
6           for (int i = 0; i <string.length(); i++) {
7               System.out.print(String.format("U+%04X ", string.codePointAt(i)));
8           }
9           System.out.println();
```

```
10              }
11
12              @Test
13              public void test() {
14                  String han = "한";
15                  printIt(han);
16
17                  String nfd = Normalizer.normalize(han, Normalizer.Form.NFD);
18                  printIt(nfd);
19
20                  String nfc = Normalizer.normalize(nfd, Normalizer.Form.NFC);
21                  printIt(nfc);
22              }
23
24          }
```

Java는 유니코드 정규화 기능을 지원하고 있다. 아래 코드는 그 예제이다.

?

아래는 위의 코드를 실행한 결과이다.

한

U+D55C

ㅎㅏㄴ

U+1112 U+1161 U+11AB

한

U+D55C

만약, 아래아 한()을 NFC로 만들고자 한다면, 'ㅎ'(U+1112), 'ㆍ'(U+119E), 'ㄴ'(U+11AB)를 결합하면 된다. 옛한글의 경우 글꼴에 따라 출력이 되지 않을 수 있으므로, 옛한글 글꼴을 지원하는 은글꼴 또는 함초롬체를 사용하여야 한다.

한양 PUA(Private Use Area) 코드

윈도 XP까지 시스템 글꼴을 제작해 온 한양정보통신에서 옛한글을 표현하기 위해 사용한 유니코드의 사용자 정의 영역 코드(U+E0BC ~ U+F77C)를 말한다. 옛한글에서 많이 쓰이는 5천여 개의 완성형 음절을 정의하였으며, 비표준이나 현재까지도 많이 사용하고 있다.

Java와 한글

Java는 String에서 사용하는 인코딩은 UTF-16 BE(Big Endian)이다. 문자열 전송/수신을 위해서 직렬화가 필요할 때에는 변형된 UTF-8(Modified UTF-8)을 사용한다. Java의 DataInput, DataOutput 인터페이스 구현체에서는 문자열을 기록하거나 읽어 들일 때 이 변형된 UTF-8을 사용한다. 변형된 UTF-8의 인코딩 규칙은 표5에서 볼 수 있다.

표 5 변형된 UTF-8 인코딩 규칙

| 코드 범위 | 인코딩 규칙 |
| --- | --- |
| U+0000 | 11000000 10000000 (0xC080) |

| | |
|---|---|
| U+0001 ~ U+FFFF | UTF-8 인코딩과 동일 |
| U+010000 ~ U+1FFFFF | UTF-16 인코딩한 값을, UTF-8 인코딩함 (CESU-8) |

변형된 UTF-8에서 U+0000을 2바이트로 표시하는 이유는 인코딩된 결과에 널 문자(00)가 나타나지 않도록 하기 위해서이다. C언어와 같이 NULL 문자를 문자열의 끝으로 처리하는 언어에서 U+0000을 읽을 때, 문자열의 끝으로 잘못 처리하는 일이 없도록 하기 위해서이다. 그리고 U+010000 이상의 코드를 표현하기 위한 CESU-8(Compatibility Encoding Scheme for UTF-16:8-bit)은 UTF-8의 변형인데 코드 포인트 U+010000 이상의 글자를 표현하기 위한 방법이다. Java에서 글자를 표현하기 위해서 2바이트 크기를 가지는 char를 사용하는데, 전체 유니코드 글자를 2바이트로 표현할 수 없기 때문에 이러한 방식을 사용한다. Java의 변형된 UTF-8은 CESU-8에 NULL 문자 처리(U+0000)을 추가한 것이다.

한글의 표현과 인코딩

Java에서는 유니코드의 코드 포인트 값을 String.codePointAt(int); 메서드를 이용하여 확인할 수 있다. 다음은 '한글'(U+D55C U+AE00)에 대한 코드 포인트 값을 출력한 예이다.

?

```
1       String string = "한글";
2       for (int i = 0; i <string.length(); i++) {
3           System.out.print(String.format("U+%04X ", string.codePointAt(i)));
4       }
5       System.out.println();
```

코드 포인트에 대한 개념을 이해하고 있다면, 한글/영어 개수를 세거나, 바이트 수에 맞추어 한글/영어 문자열 자르기 등은 어렵지 않을 것이다.

Java에서 인코딩된 값을 알아보려면, getBytes() 메서드를 이용하여 확인할 수 있다. 다음은 '한글'에 대한 인코딩 값을 출력한 예이다.

?

```
1       String string = "한글";
2       byte[] bytes = string.getBytes();
3       for (byte b : bytes) {
4           System.out.print(String.format("0x%02X ", b));
5       }
6       System.out.println();
```

여기에서 염두에 둘 점은 Java에서 문자열은 항상 UTF-16 BE 인코딩으로 저장되며, file.encoding시스템 프로퍼티에 의해 인코딩 값이 결정된다는 점이다. 특히 C언어를 많이 다루어 본 개발자라면 문자열을 C의 1바이트 char 배열로 여기는 경향이 강하기 때문에 이 차이점을 잘 이해해야 한다. 언어 차원에서 유니코드와 같은 캐릭터 인코딩을 지원하지 않는 C와 달리 Java에서는 언어 차원에서 유니코드와 여러 코드 페이지를 지원한다.

Java는 String 객체 내부(메모리상에서) UTF-16 BE 인코딩으로 문자열을 저장하고, 문자열을 입/출력할 때에만 사용자가 지정

한 인코딩 값 또는 운영체제의 기본 인코딩 값으로 문자열을 인코딩한다. JVM 기본 인코딩은 JVM 로딩 시에만 초기화되므로, 코드 중간에서 file.encoding 프로퍼티를 바꾸는 것은 아무 의미가 없다. 만약 file.encoding이 지정되어 있지 않다면, OS 환경변수(예: LANG) 값을 따른다. Java에서 글자를 깨뜨리지 않으려면, 문자 집합의 이름을 지정해야 한다. 예를 들어, 문자열 객체의 getBytes() 메서드를 이용하여 바이트 배열을 얻고자 할 때, getBytes() 대신 getBytes(String charsetName) 메서드를 사용하고, 반대로 바이트 배열에서 문자열 객체를 얻고자 할 때, new String(byte[] b) 대신 new String(byte[] bs, String charsetName) 메서드를 사용한다.

웹과 한글

한글 처리, 특히 웹에서의 한글 처리는 무척 까다롭다. 그 이유는 사용자의 환경이 매우 다르다는 데 있다. 웹 프로그래밍을 하려면, 운영체제의 기본 인코딩, Java 소스 코드의 인코딩, JSP 파일의 인코딩, HTTP 요청의 인코딩, HTTP 응답의 인코딩, 데이터베이스의 인코딩, 파일의 인코딩 – 이렇게 많은 인코딩과 마주하게 된다.

org.apache.tomcat.util.http.Parameters.processParameters 메서드

?위 코드를 보면 알 수 있듯이, 인코딩이 올바르지 않게 설정되면 파라미터에 잘못된 값이 들어감을 알 수 있다. 다음 코드는 URL 디코딩이 잘못되면 어떤 결과가 초래되는지 쉽게 살펴볼 수 있는 예이다.

?

```
1               String hangul = "한글";
2               String[] encodings = new String[] {"EUC-KR", "UTF-8", "ISO8859-1"};
3
4               for (String encoding1 : encodings) {
5                   String encoded = URLEncoder.encode(hangul, encoding1);
6                   System.out.println(encoded);
7                   System.out.print("₩t");
8
9                   for (String encoding2 : encodings) {
10                      String decoded = URLDecoder.decode(encoded, encoding2);
11                      System.out.print(decoded + "₩t₩t");
12                  }
13                  System.out.println("₩n");
14              }
```

```
1       public void processParameters(byte bytes[], int start, int len, String enc) {
2       ...
3       tmpName.setBytes(bytes, nameStart, nameEnd □ nameStart);
4       tmpValue.setBytes(bytes, valStart, valEnd □ valStart);
5       ...
6       addParam(urlDecode(tmpName, enc), urlDecode(tmpValue, enc));
7       ...
8       }
```

웹에서 한글이 왜 깨지는가? 브라우저 인코딩 값과 서버 인코딩 값이 다르기 때문이다. Tomcat에서는 파라미터 인코딩 및 키와 값을 설정하기 위해 org.apache.catalina.connector.Request.parseParameters 메서드와 org.apache.tomcat.util.http.Parameters.processParameters 메서드를 이용하여 처리하고 있다.

org.apache.catalina.connector.Request.parseParameters 메서드

?

```
1       protected void parseParameters() {
2       ...
3       String enc = getCharacterEncoding();
4       ...
5       if (enc != null) {
6           parameters.setEncoding(enc);
7       } else {
8           parameters.setEncoding("ISO-8859-1");
9       }
10      ...
11      }
```

웹에서 여러 인코딩을 지원하려면, 인코딩된 URL 문자열과 사용한 인코딩 정보를 파라미터로 전달해야 한다. 예를 들어, "/search.nhn?query=%C7%D1%B1%DB&ie=EUC-KR" 과 같이 URL이 설정되어 있다면, ie 파라미터 값을 이용하여 query의 파라미터 값을 URL 디코딩하면 된다. 그리고 가능하다면 Javascript의 encodeURI 메서드 (또는 encodeURIComponent 메서드)를 사용하는 것이 좋다.

Javascript에서의 URL 인코딩

Javascript는 escape, encodeURI, encodeURIComponent 메서드를 이용하여 URL을 인코딩할 수 있다. 이 중 escape 메서드는 A~Z, a~z, 0~9, @*-_+./ 문자가 아니면 유니코드 형식으로 인코딩하는데, ASCII 문자는 %XX, 그 외는 %uXXXX 형태로 인코딩된다. 예를 들어, '한글'을 escape 메서드로 인코딩하면, %uD55C%uAE00으로 인코딩되므로, Tomcat에서 URL 디코딩 시에 문제가 발생하게 된다. 일반적으로 문자열을 URL 인코딩하기 위해서 encodeURI 메서드를 많이 사용하며, :;=?& 문자는 인코딩하지 않는다.

Java의 URLEncoder.encode 메서드와 Javascript의 encodeURI 메서드는 공백(whitespace)을 '%20'으로 인코딩하느냐, '+'로 인코딩하느냐만 다르다. 마지막으로 encodeURIComponent 메서드는 encodeURI 메서드와 유사하지만, :;/=?&도 인코딩한다.

브라우저에서의 EUC-KR 인코딩

EUC-KR 인코딩은 2,350자의 한글만 사용할 수 있다. 그러면 EUC-KR 인코딩으로 이루어진 웹 페이지에서 '똠방각하'와 같은 문자열은 어떻게 처리될까? 브라우저 별로 다국어가 포함된 URL 인코딩 처리하는 방법이 다르다. EUC-KR 인코딩으로 표현 가능한 '한글'을 웹 URL에 넣어 브라우저 인코딩 테스트를 해보면 Internet Explorer, Firefox, Chrome 모두 한글을 '%C7%D1%B1%DB' 로 인코딩한다. 그러나 EUC-KR로 인코딩할 수 없는 '똠방각하' 를 처리할 때는 브라우저마다 결과가 다르다. 브라우저 별 EUC-KR 인코딩 방법을 테스트하기 위해 EUC-KR 인코딩을 사용하는 검색 시스템인 알타비스트를 이용해보기로 한다.

브라우저 검색 URL의 p 파리미터 값을 살펴보자.

표 5 브라우저 별 EUC-KR 인코딩 결과

| 브라우저 | 인코딩된 값 | 화면에 표시되는 문자열 |
|---|---|---|
| Internet Explorer | %26%2346624%3B%B9%E6%B0%A2%C7%CF | 똠방각하 |
| Firefox | %A4%D4%A4%A8%A4%C7%A4%B1%B9%E6%B0%A2%C7%CF | ㅍㅗㅁ방각하 |
| Chome | %8Cc%B9%E6%B0%A2%C7%CF | c방각하 |

Chrome은 EUC-KR에 있는 확장 완성형의 문자를 지원하지 않기 때문에 '똠'을 인코딩할 수 없다. Internet Explorer는 EUC-KR에 없는 문자의 경우 유니코드 포인트 값으로 표현한다. 즉 '똠'의 유니코드 코드 포인트 값인 46624(U+B620)으로 URL 을 인코딩 한다. Firefox는 한글 채움 문자를 이용하여 음절을 표시하고 있다. 한글 채움 문자는 KS X 1001 표준안에 정의되어 있으며, (채움) 초성 중성 종성의 형태로 표시하고, 초성, 중성, 종성의 값이 없는 경우 (채움)으로 표시한다. EUC-KR 인코딩에서는 (채움) 값이 0xA4 0xD4이므로, 다음과 같이 인코딩된다.

표6 EUC-KR로 똠방각하를 인코딩할 때

| 인코딩 | 값 |
|---|---|
| %A4%D4 | (채움) |
| %A4%A8 | ㅍ |
| %A4%C7 | ㅗ |
| %A4%B1 | ㅁ |
| %B9%E6 | 방 |
| %B0%A2 | 각 |
| %C7%CF | 하 |

알아두면 좋은 것들

영문 MS Windows는 CP1252, 한글 MS Windows는 MS949가 기본 인코딩이다. 리눅스에서는 LANG 환경 변수에 따라 다르지만, ko, ko_KR, ko_KR.eucKR은 모두 EUC-KR 인코딩이며, ko_KR.UTF-8만 UTF-8 인코딩이다. CentOS의 경우 /etc/sysconfig/i18n에서 시스템 기본 인코딩을 설정할 수 있다. 참고로 i18n은 국제화(internationalization)를 의미하며, l10n은 지역화(localization)을 의미한다. 18과 10이라는 숫자는 i와 n 사이, 또는 l과 n 사이의 글자 수를 의미한다. 요즘 편집기는 여러 인코딩을 처리할 수 있으므로, 보통 문서의 처음에 BOM(Byte Order Mark)이라는 값을 지정하여 인코딩 정보를 저장한다. UTF-8은 0xEF 0xBB 0xBF이며, 나머지 인코딩에 대한 BOM 값은 위키백과(http://en.wikipedia.org/wiki/Byte_order_mark)를 참고하면 좋다.

[1] 유니코드, http://ko.wikipedia.org/wiki/%EC%9C%A0%EB%8B%88%EC%BD%94%EB%93%9C

[2] UTF-8, http://ko.wikipedia.org/wiki/UTF-8

[3] 유니코드 정규화, http://ko.wikipedia.org/wiki/%EC%9C%A0%EB%8B%88%EC%BD%94%EB%93%9C_%EC%A0%95%EA%B7%9C%ED%99%

[4] 옛한글, http://ko.wikipedia.org/wiki/%EC%98%9B%ED%95%9C%EA%B8%80

[5] 바이트 순서 표식, http://ko.wikipedia.org/wiki/%EB%B0%94%EC%9D%B4%ED%8A%B8_%EC%88%9C%EC%84%9C_%ED%91%9C%EC%8B%

NBP 쇼핑서비스개발팀 오영은

"아는 것이 힘이다(scientia est potentia)" - 프란시스 베이컨. 익숙한 것에 머무르지 않고, 더 나은 것에 대한 동경으로, 무지(無知)의 두려움으로부터 헤어나기 위해, 이제 겨우 한 발자국씩 떼고 있습니다.

Tag : 유니코드 , 한글 , 인코딩 , JAVA

## 필자의 저작내용

s/w & 저작물

| 순위 | 등록일자 | 등록번호 | 제 호 |
|---|---|---|---|
| 1 | 2005-10-31 | 2005-01-189-005496 | GPS식상전환progarm |
| 2 | 2006-02-06 | 2006-01-189-000538 | GPS식 상전환 온도프로그램 |
| 3 | 2006-07-24 | 2006-01-189-003724 | GPS식 상전환 프로그램(2 way system) |
| 4 | 2007-04-17 | 2007-01-189-001994 | GPS-type 상 전환 program (일출몰시간변화량) |
| 5 | 2008-10-06 | 2008-01-129-005071 | GPS type dephase and rephase technique program (Tw... |
| 6 | 2009-03-31 | 2009-003190 | quantum thoroughepoch's twilight-type dia-cabin plant system by MSbrothers 식물공장 |
| 7 | 2009-04-17 | c-2009-003616 | 생체시계진자 vector |
| 8 | 2009-04-20 | c-2009-003635 | 미나리는 한해살이식물이다. |
| 9 | 2009-07-30 | 2009-01-189-003940 | GPS-type secondly metabolite generate technique p... |
| 10 | 2009-09-14 | c-2009-007741 | 태양광발생소멸시각변화량vector에 대한 연구 |
| 11 | 2009-09-15 | 2009-01-189-004688 | GPS-type multi-technical precise timer s/w |
| 12 | 2009-12-30 | 2009-01-154-010186 | 따로 하나로 직구매프로그램(make a direct deal with... |
| 13 | 2010-01-11 | c-2010-000-444 | 종의 분화는 환경동조산물(The diffrentiation of species that is circumstances alignment |
| 14 | 2010-04-12 | c-2010-004004 | 광합성 흡열작용을 이용한 기온양극화 대책방향 연구고찰 |
| 15 | 2010-07-02 | 2010-01-219-003794 | GPStype Ubiquitous S/W (식물공장 환경제어프로그램... |
| 16 | 2010-04-13 | c-2010-004043 | 식물공장 광 디자인에 대한 연구 |
| 17 | 2011-07-06 | c-2011-006182 | 지구가 흔들리기 때문에 생체시계가 돌아간다. |
| 18 | 2011-07-27 | c-2011-007039 | 지구 흔들림에 대하여 |
| 19 | 2011-11-02 | 2011-01-189-007298 | 냄새없는 12시간현미발아program |
| 20 | 2011-12-12 | 2011-01-182-009100 | 무취발아 가수분해program |
| 21 | 2012-01-05 | c-2012-000237 | RFID를 이용한 따로 하나로직구매 운용 PROGRAM s/w |
| 22 | 2012-01-12 | c-2012-000597 | 벼 멸칭육묘 program s/w |
| 23 | 2012-02-20 | c-2012-003291 | GPStype RFID 물류유통정보관리 program s/w |
| 24 | 2012-04-02 | c-2012-006802 | 현미의 12시간이내 발아기술과 전처리에 대한 고찰 |
| 25 | 2012-06-20 | c-2012-012135 | 물레현미발아프로그램 s/w |
| 26 | 2012-07-07 | c-2012-013515 | 수직수평조립반도체문자입력프로그램등록 s/w |
| 27 | 2013-03-05 | c-2013-004691 | 컴퓨터용의 빠르고 정밀하며 정확한 자소 한글의 이해 |

# 더 빠르고 더 정밀하고 더 정확한 한글자소

**블로그** 요람에서부터 123세까지의 활백 생활지침서 http://blog.daum.net/daiperng

**저자** 최농부

**발행일** 2016.03.07 04:04:10

Daum 블로그