LISP 머신상의 한글환경 구축*

김 영환 , 김 진형

한국과학기술원 전산학과

Construction of the Hangul Environment on LISP Machine

Young Whan Kim and Jin Hyung Kim

Department of Computer Science, KAIST

요 약

LISP 머신은 기호처리 능력, 고품질의 그래픽 기능 및 편리한 사용자 인터페이스 등을 제공함으로써 복잡한 대규모의 인공지능 프로그램을 개발하는데 많이 사용되고 있다 LISP 머신이 국내에서 효과적으로 활용되기 위해서는 한글처리를 자유로이 할 수 있는 한글환경이 구축되어야 한다

본 논문에서는 LISP 머신 "TI EXPLORER" 에 한글처리를 효과적으로 수행할 수 있는 한글 인터페이스를 개발하여 한글환경을 구축한 내용을 소개한다 한글 인터페이스는 크게 한글모아쓰기 오토마타(Automata), 입 출력 루틴 그리고 입력 편집기로 구성되어 있다

I. 서 론

인공지능 시스템이 기존의 프로그램들과 크게 다른 점은 지식을 컴퓨터에 입력하여 추론 과정을 거침으로써 주어진 문제를 해결할 수 있다는 점이다 지식의 복잡성과 추론 과정의 필요성이 기존의 프로그래밍 기법의 사용을 불가능하게 하여 인공지능 시스템의 구성에 독특한 개발 기법이 사용되고 있다 [1] 인공지능 시스템의 개발은 초기에는 특정 도구나 하드웨어를 사용하지 않고 일반적인 컴퓨터를 이용하여 프로그래밍 언어로 개발하여 왔으나 그 개발에 소요되는 시간 및 노력이 방대해짐에 따라 잘 개발된 소프트웨어 도구, 즉 인공지능 개발 시스템을 요구하게 되었고 이러한 시스템을 효과적으로 수행시킬 수 있도록 설계된 인공지능 워크스테이션(LISP Machine)의 필요성이 대두되었다

LISP 머신은 LISP 언어를 사용하여 구성한 시스템으로서 강력한 인공지능 소프트웨어 개발환경을 지원하고 기호처리를 효율적으로 할 수 있도록 설계된 컴퓨터이다[2] 이러한 LISP 머신은 현재 국내에 소개되고 있는 단계이나, 멀지 않은 장래에 국내의 여러 학교, 연구소 및 산업체에 많이 보급되어 인공지능 연구 및 응용에 사용될 것으로 예상한다 이와같이 LISP 머신을 국내에서 효과적으로 사용하여 현실적인 여러 문제들을 해결하기 위해서는 우선적으로 한글 자료를 처리할 수 있는 환경을 LISP 머신상에서 구현하여야만 한다 국내의 한글처리에 관한 기술수준은 한글 워드프로세서, 한글 터미널, 한글 프린터, 한글 운영체제 등을 개발하면서 많이 발전하였으며 한글 LISP에 관한 연구도 수행되어 LISP에서 한글기호를 사용 할 수 있다 [3] 하지만 LISP 머신상 에서 한글 환경에 대한 연구는 이루어지지 않았고 그러한 시스템도 현재까지는 없는 상태이다

본 논문에서는 KAIST 전산학과 인공지능 연구실에서 보유하고 있는 LISP 머신 TI EXPLORER에 한글 처리를 효과적으로 할 수 있는 한글 인터페이스를 개발하여 한글 환경을 구축한 내용에 대해 설명한다. II장에서 한글 기본환경 설계 및 구현에 대하여 설명하고 III장에서 한글 입출력 프로그램 설계 및 구현에 대하여 설명한 후 IV장에서 결론을 맺는다

II.한글 기본 환경 설계 및 구현

LISP 머신에서 한글 자료를 처리할 수 있게 하기 위해서는 사용자가 키보드를 통하여 한글 자료를 시스템 내로 입력할 수 있어야 하고 입력된 데이타를 분석하여 의도하는 한글

* 본 연구는 전자통신연구소의 수탁연구비로 일부 지원되었음

한 문자씩으로 구성하고 이에 해당하는 코드를 발생하여 시스템 내의 기억장치에 보관할 수 있어야 하며, 입력되는 과정이 화면을 통하여 사용자에게 보여져야 한다. 그리고 자료가 입력된 후, 시스템이 제공하는 편집 기능을 이용하여 편집될 수 있어야 하며, 저장된 자료가 사용자가 원할 때 화면에 출력될 수 있어야 한다. 이와같은 기능을 발휘하는 한글 입출력 인터페이스를 위하여 기본적으로, 한글 폰트, 한글 건반배열 설계, 한글 코드등으로 이루어진 한글 기본환경을 구현하여야 한다.

1.한글 폰트[4]

한글 입.출력 시스템을 한글 모아쓰기 오토마타를 이용하여 3바이트(byte) 조립식으로 설계하였기 때문에 이에 맞추어 한글 폰트도 초성, 중성, 종성의 자소별로 설계하였다. 한글 입.출력 시스템을 개발하기 위하여 우선적으로 사용할 수 있는 한글 폰트가 필요하였으며 이를 위하여 127개의 자소 폰트로만 구성된 고딕체의 6벌식 폰트를 설계하여 사용하였다. 한글 입.출력 프로그램이 폰트의 수가 많은 아주 세련된 한글 폰트도 수용할 수 있도록 설계되었기 때문에 폰트를 바꾸는 작업이 용이하게 되어 있다. 따라서 이미 개발되어 사용중인 우수한 한글 폰트를 시스템의 큰 변경 없이 쉽게 사용할 수 있다.

한글 폰트의 구현은 LISP 머신이 제공하는 폰트 에디터를 이용하여 수행하였다.

2. 한글 건반배열

본 연구에서는 한글 정보 처리용 건반배열의 표준안[5]을 수용하여 LISP 머신의 건반배열에 맞추어 한글 건반 배열을 설계하였다. 건반 수용문자는 기본 자모 24자, 쌍자음 5자, 복모음 4자로 이루어진 총 33 자의 한글 자모를 수용하였으며 건반 배열은 <그림 1>과 같다.

<그림 1> 건반 배열

'F1' 키는 한글 입력시 한.영 모드 변환키로 사용하였다. 이 한.영 모드 변환키는 편의에 따라 임의의 키를 지정해서 쓸 수 있다.

3. 한글 코드

LISP 머신의 특성과 LISP 머신에서 앞으로 한글 자료의 응용분야등을 고려하여, 8 비트로 이루어진 자소 코드를 사용하여 3 바이트 모아쓰기 방식에 의해 한 글자를 이루도록 한글 코드를 설계하였다.

현재 LISP 머신의 문자집합(character set) 중에서 사용되고 있지 않는 부분(octal 272 부터)을 한글 자소 코드로 지정하여 LISP 머신에서 제공하는 기존의 소프트웨어에서 한글 자료를 사용하더라도 아무런 문제점이 없도록 하였다[6]. 3 바이트 모아쓰기 방식을 택하여 N 바이트 모아쓰기 방식 채택시 생길 수 있는 한글 글자의 가변적인 길이로 인한 단점을 방지하였으며 문서 편집기 내에서 한글 자료처리에 통일성을 기할 수 있도록 하였다. 이러한 한글 코드는 키보드에서 발생되지 않고 소프트웨어적으로 한글 오토마타에서 한글 한 글자가 완전히 인식되면 코드가 발생되게 되어있다. 코드는 사전 순서식으로 자음, 모음 순으로 순차적으로 배열하여 한글 자료의 순서적 배열도 용이하도록 하였다.

한글 코드의 구성은 <표 1>과 같다.

	자음	8진수	10진수		자음	8진수	10진수		모음	8진수	10진수
0	blk	272	186	21	ㅅ	317	207	31	ㅏ	331	217
1	ㄱ	273	187	22	ㅆ	320	208	32	ㅐ	332	218
2	ㄲ	274	188	23	ㅇ	321	209	33	ㅑ	333	219
3	ㄱㅅ	275	189	24	ㅈ	322	210	34	ㅒ	334	220
4	ㄴ	276	190	25	ㅉ	323	211	35	ㅓ	335	221
5	ㄴㅈ	277	191	26	ㅊ	324	212	36	ㅔ	336	222
6	ㄴㅎ	300	192	27	ㅋ	325	213	37	ㅕ	337	223
7	ㄷ	301	193	28	ㅌ	326	214	38	ㅖ	340	224
8	ㄸ	302	194	29	ㅍ	327	215	39	ㅗ	341	225
9	ㄹ	303	195	30	ㅎ	330	216	40	ㅗㅏ	342	226
10	ㄹㄱ	304	196					41	ㅗㅐ	343	227
11	ㄹㅁ	305	197					42	ㅗㅣ	344	228
12	ㄹㅂ	306	198					43	ㅛ	345	229
13	ㄹㅅ	307	199					44	ㅜ	346	230
14	ㄹㅌ	310	200					45	ㅜㅓ	347	231
15	ㄹㅍ	311	201					46	ㅜㅔ	350	232
16	ㄹㅎ	312	202					47	ㅜㅣ	351	233
17	ㅁ	313	203					48	ㅠ	352	234
18	ㅂ	314	204					49	ㅡ	353	235
19	ㅃ	315	205					50	ㅡㅣ	354	236
20	ㅂㅅ	316	206					51	ㅣ	355	237

<표 1> 한글 코드표

III. 한.글 입출력 프로그램 설계 및 구현[6]

II장에서 설명한 한글 기본환경을 바탕으로 하여 한글 입.출력 프로그램은 크게 한글 모아쓰기 오토마타, 시스템 입.출력 루틴, 입력 편집기인 Rubout Handler로 구성되어 있다.

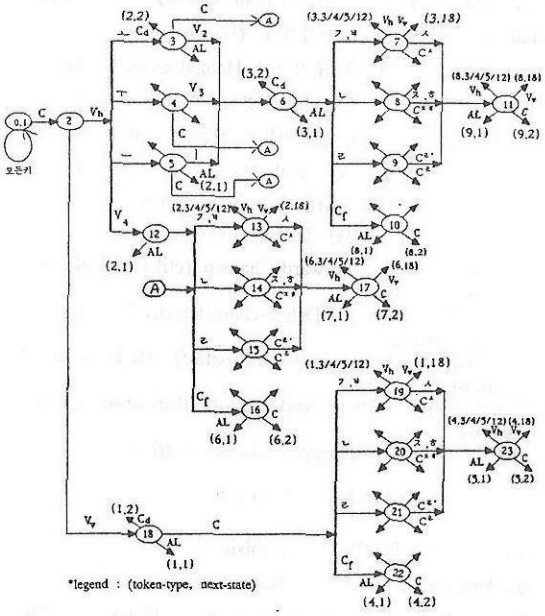
1. 한글 모아쓰기 오토마타

가. 설계 원칙

- o 기본자소 33자(단자음:14, 단모음:10, 중자음:5, 복모음:4)를 입력으로 한다.
- o 국문법상 허용되지 않는 글자는 형성할 수 없다. 형성할 수 없는 자소가 입력되었을 때는 에러로 처리하고 이 자소를 무시한다.
- o 키보드에서 입력 시 한.영 입력 모드 구분은 한.영 구분 키를 사용한다. 이 외에도 'Hyper' 키를 이용하여 한.영 구분 키를 사용하지 않고서도 한글입력이 가능하게 하였다.
- o 한글 한 글자는 초성, 중성, 종성 각 1 바이트씩 3 바이트로 구성된다.
- o 모아쓰기 진행 중 'rubout' 키가 입력되면 현재 모아쓰기 중인 자소가 모두 삭제되고 현재 모아쓰기 중인 자소가 없으면 그 전에 인식된 글자를 삭제한다.
- o 현재 커서 위치에서 인식될 때까지 모아쓰기를 한 후 인식된 글자 외의 남은 자소를 다음 커서 위치에서 모아쓰기하여 화면에 출력한다. 이 때 자소 하나 하나가 입력될 때마다 글자가 변화하는 과정을 계속 화면에 출력한다.

나. 전체 구성

한글 오토마타는 전부 24개의 상태로 이루어진 상태 변환 도표(state-transition diagram)로 구성된다. 상태 변환 도표는 <그림 2>와 같다. 각 상태에서 상태 변화는 입력된 문자에 따라 결정된다. 입력된 문자는 모두 22개의 문자집합(char-class)으로 구분하였다. 문자집합의 구분은 <표 2>와 같다. 각 상태에서 그 때까지 형성된 글자의 token-type을 결정하여 글자표시 및 코드형성에 사용한다. Token-type은 <표 3>과 같다.



<그림 2> 상태변환 도표

Char-Class	ELEMENT
Vh	ㅗ,ㅛ,ㅜ,ㅠ,ㅡ
Vv	ㅏ,ㅑ,ㅓ,ㅕ,ㅣ,ㅐ,ㅒ,ㅔ,ㅖ
C	ㄱ, ... , ㅎ, ㄲ,ㄸ,ㅃ,ㅆ,ㅉ
Cz	C - {ㄱ,ㅁ,ㅅ,ㅇ,ㄹ,ㅍ,ㅎ}
Cxs	C - { ㅈ,ㅎ }
Cd	ㅋ,ㅌ,ㅊ
Cf	C - Cd
CA	C - { ㅅ }
CJ	C - { ㄱ }
AL	한글이 아닌 모든 문자
V2	ㅏ,ㅑ,ㅣ
V3	ㅓ,ㅕ,ㅣ
V4	Vh - { ㅗ,ㅜ,ㅡ }
Cz'	C - Cz

<표 2> 문자집합 구분표

TOKEN TYPE	TOKEN
1	C-V
2	C-V
3	C-V-V
4	C-V-C
5	C-V-C+C
6	C-V-C
7	C-V-C-C
8	C-V-V-C
9	C-V-V+C-C
:0	한글과 기타
-1	ERROR

<표 3> Token-type 분류

다. 세부 설계

(1) 한.영 구분

Key-board에서 입력시 한.영 구분은 한.영 구분키를 이용한다. 이 구분키는 시스템에서 정의되어 사용되지 않는 키를 지정하여 사용하면 된다. 본 시스템에서는 'F1' 키를 사용하였다.

이와같은 모드 변환 방법외에 영문 모드에서 변환 키 사용 없이 바로 한글을 입력시킬 수도 있도록 시스템을 설계하였다. 즉 'Hyper' 키와 원하는 한글 키를 동시에 치면 영문 모드에서도 한글이 시스템 내로 입력되게 하였다. 한글 모드이면 키보드에서 입력되는 데이타가 코드 생성 프로그램에 의해 해당 한글 자소 코드로 변환된다. 'Hyper' 키를 이용하였을 때도 모드 변환 시와 마찬가지로 해당 한글 자소 코드로 변환된다.

(2) 화면 출력

한글 모아쓰기 과정에서는 그 변화를 계속하여 화면에 출력해야 한다. 모아쓰기 과정 중 어떤 한 순간에 한글 한 글자는 항상 초성, 중성, 종성 3 바이트로 변환된다. 만약 중성이나 종성이 아직 입력되지 않는 상태이면 이 부분은 공백코드(fill-code)로 대치된다. 이렇게 형성된 3 바이트로부터 각 자소에 해당되는 폰트의 인덱스를 계산하여 항상 초성, 중성, 종성의 순으로 세번 같은 커서 위치에 화면 출력한다. 지금까지 형성된 글자의 token-type을 이용하여 해당 폰트의 인덱스를 계산한다.

(3) 3 바이트 코드 구성

한글 모아쓰기 과정은 항상 3 바이트 코드를 만들어 가는데, 이는 지금까지 입력된 자소의 token-type과 자소 코드를 이용하여 생성된다.

글자가 완전히 인식되면 이의 3 바이트 데이타를 1 바이트씩 세번에 걸쳐 rubout 버퍼에 출력하는데 이는 시스템이 한꺼번에 3 바이트를 출력할 수 없고 한번에 1 바이트씩 출력하도록 설계되어 있기 때문이다.

2. 시스템 입.출력 루틴

LISP 머신은 플레버(Flavor)를 기초로 한 윈도우 시스템을 이용하여 입출력 인터페이스를 구성하고 있다. 각 윈도우는 사용하고자 하는 *지정폰트(default font)*를 윈도우 *객체*의 슬롯(slot) 값으로 갖고 있어서 따로 폰트를 지정하지 않으면 그 폰트를 사용하여 윈도우에 글자를 출력한다. 새로운 성질의 윈도우를 만들기 위해서는 그 윈도우를 구성하는 플레버를 새로 만들면 된다. 많은 플레버들이 각각 단편적인 특성을 갖는 컴포넌트 플레버로 만들어져 있으며 필요에 의하여 각각의 특성을 혼합(mixing)함으로써 다양한 기능의 윈도우를 만들 수 있다. 따라서 본 연구에서는 한글 폰트를 사용하는 오토마타를 수행시켜서 글자를 조립하여 주고 한글의 적절한 음절을 구성할 수 있는 특징만을 제공해 주는 컴포넌트 플레버로서 Hangul-mixin을 정의하였다. Hangul-mixin을 기존의 LISP 리스너 윈도우에 혼합하여 한글을 사용하는 *LISP* 리스너로서 Hangul-Listener를 구성하였다. Hangul-mixin을 편집기 윈도우에 혼합하면 편집기에서도 한글 사용이 가능하다. Hangul-mixin은 아래와 같은 슬롯을 가지고 있다.

- o Hangul-font : 지정폰트(default font)명
- o Hangul-flag : 입력상태 표시
- o *Hg-mode-change-key* : 한.영 구분키 정의
- o Hg-stroke-buffer : 입력된 키 스트로우크(Stroke)들을 저장하는 버퍼
- o Hg-stroke-state : 현 상태번호 표시
- o *Hg-fill-code* : 공백 코드값

- o Hg-code-buffer . 모아쓴 한 음절(3 바이트) 저장
- o Hangul-state : 0이면 영문 모드
- o Hg-stroke-num : 현재 모아쓰기 중인 글자의 자소 수
- o Hg-jaso-left . 모아쓰기된 후 시스템으로 넘겨주지 않고 남은 자소의 갯수
- o Hg-old-code-buffer : 바로 전에 모아쓴 한 음절을 저장
- o Hg-jaso-left-buffer : 입력 명령어 :tyi에 의해 넘겨줄 자소를 저장

Hangul-mixin에서 필요로 하는 특성으로 Hangul-mixin에서 입력을 위하여 정의된 method는 다음과 같다

·after ·init
한글 폰트가 적재되지 않았으면 자동으로 디스크에서 Hangul-font에 지정한 한글 폰트를 적재한다
·set-hangul-font font
Hangul-font 슬롯에 한글 폰트를 지정하는 method로서 이 폰트가 지정폰트가 된다
·any-tyi
한 바이트의 한글자소를 입력시킨다 Rubout-handler-buffer에 한글자소가 남아 있으면 이 자소를 넘겨주고 아니면 오토마타를 불러서 모아쓰기된 한 음절을 넘겨준다. 영문 모드의 경우에도 오토마타를 통해서 영문 알파벳이 넘겨진다.
:tyo ch
한글 자소 코드인 ch를 출력한다
:string-out string &optional (start 숫자) end
한글, 영문 혹은 혼합된 스트링을 start부터 end까지 출력한다 신속한 처리를 위해서 한글이 포함된 스트링이면 Hg-sheet-string-out 함수를 호출하고 영문 스트링이면 sheet-string-out을 호출한다.
·hg-sheet-clear-char sheet &optional font
커서가 위치한 곳에서 한 글자의 크기 만큼을 지워서 공백을 만들어 준다. 이때 폰트를 지정해 줌으로써 그 폰트가 차지하는 크기를 계산하여 지우고자 하는 높이 및 넓이를 지정할 수 있다
hg-syllable-out cho-sung joong-sung jong-sung sheet & optional (advance-p nil) (erase-p t)
세 개의 자소 (cho-sung joong-sung jong-sung)을 조합하여 한 음절을 만들어서 Sheet에 출력한다 이때 커서의 이동 플래그인 advance-p가 T이면, 즉 새로운 글자가 인식되어 현재 커서 위치 다음에 화면 출력 해야할 경우, 한 글자 넓이 만큼 커서를 전진시킨 후 글자를 출력한다 또한 erase-p가 T이면 즉, rubout 키가 입력 되어 현재 커서위치를 지우고자할 경우, 커서 위치를 지워서 공백을 만든 후 출력한다
·hg-make-room-for-hg-char sheet &optional hg-font
지정된 위치에 한글 글자를 삽입하기 위하여 뒷부분에 있는 글자들을 뒤로 밀어내고 공백을 제공한다
hg-sheet-insert-string string
커서의 위치에 한글 스트링을 삽입한다 이를 위해 hg-make-room-for-hg-char가 필요하다
hg-sheet-advance
커서를 한글 폰트의 넓이 만큼 전진시킨다 이때 한글 리스너 윈도우의 오른쪽 경계에 도달하면 다음 줄로 커서를 이동시킨다. .hg-syllable-out에 의해 불려진다
hg-sheet-string-length sheet string
주어진 스트링(한글이 포함될 수 있음)의 길이를 바이트로 계산한다
hg-sheet-delete-char
커서의 위치에서 한 글자(한글인 경우는 한 음절)를 제거하고 뒷부분을 앞으로 당긴다
hg-sheet-delete-string string
커서의 위치에서 스트링의 길이 만큼을 제거한 후 뒷부분을 앞으로 당긴다 .hg-sheet-delete-char를 이용한다
untyi
한글 자소를 다음 tyi에 의하여 읽을 수 있도록 적절한 버퍼에 넣어준다. 기존 시스템에서는 untyi 할 수 있는 것은 이전에 tyi 해서 읽은 것만을 허용하는데 반해서 한글을 위해서는 임의의 글자를 untyi 할 수 있도록 하였다
compute-motion string
한글 스트링을 한글 리스너 윈도우에 출력한 후 위치해야 할 커서의 위치를 계산한다

나. 한글 Readtable의 구성

한글 자소를 시스템에서 사용하는 글자로 정의하기 위해서는 Readtable에 등록시켜야 한다. 이를 위해서 새로운 Readtable을 만들고 이를 Global 변수인 *hg-readtable*에 저장하였다. 이 한글 Readtable은 Hangul-mixin이 호출되면 default Readtable로서 지정된다

3. 입력 편집기 (Rubout Handler)

LISP 머신에는 사용자가 입력한 내용을 편집할 수 있는 기능이 있다. 이를 입력 편집기(Input Editor) 혹은 Rubout-Handler라고 한다 입력 편집기가 한글 데이타를 처리할 수 있도록 기존의 명령어들을 수정하여 Hangul-mixin에 연결하였다. 따라서 한글에 관한 이들 편집 명령어들은 Hangul-Listener에서만 수행된다. 모든 편집 명령어들은 한글 글자가 3 바이트로 구성되었으므로 우선 한글인지 영문인지를 구분한 후 한글인 경우에는 3 바이트 단위로 처리할 수 있도록 수정하였다 한글 사용을 위해서 수정된 명령어는 아래와 같다

- o 글자 단위 이동 Forward-Character(ctrl-f), Backward-character(ctrl-b)
- o 글자 단위 삭제 . Delete-character(ctrl-d), Rubout-character(rubout)
- o 단어 단위 이동 Forward-word(meta-f), Backward-word (meta-b)
- o 단어 단위 삭제 : Delete-word(meta-d), Rubout-word(meta-rubout)
- o 전후 글자 교환 · Transpose-character(ctrl-t)

위와 같은 기능을 지원하기 위해 다수의 함수들이 정의되었다.

Rubout Handler를 Hangul-mixin에 연결하기 위하여 . Rubout-handler라는 method가 Hangul-mixin 플레버에 정의되어 있다 이 method는 키로 입력된 글자들을 Rubout-handler-buffer에 넣어 주고 입력 편집기의 편집 명령어를 적용하여 그 내용을 수정할 수 있다.

IV. 결론

LISP 머신상에 한글기호 처리를 할 수 있는 한글 개발환경을 구축하였다. 하지만 보다 효율적인 한글 환경을 구축하기 위해서는 개발된 한글 환경을 바탕으로 하여 화면 편집기인 ZMACS의 한글화를 이루어야 한다 한글 인터페이스 개발시 수행한 입력 편집기의 한글화 경험이 ZMACS 한글화에 큰 도움이 되리라 생각된다

참 고 문 헌

[1] Chuck Williams, "Expert Systems, Knowledge Engineering, and AI Tools - An Overview," IEEE EXPERT, Winter 1986
[2] 이 성환, 김 진형, "LISP 머신의 현황 및 개발동향," 정보과학회지, 1986 9.
[3] 김 영환, 김 진형, "인공지능 연구를 위한 한글 개발 환경 구축 및 시험용 전문가 시스템 구현," 정보과학회 추계학술 발표논문집, 1986 10
[4] 한글 폰트 에디터 및 아이지, 한국전자통신연구소, 1986 3
[5] "정보 처리용 건반 배열," KSL 5715, 한국공업규격, 1982 6 17
[6] EXPLORER System Manual, Sperry Corporation, 1985