

## 임베디드 시스템을 위한 한글 포팅 및 출력 성능 비교

오삼권\*, 박근덕\*\*, 김병국\*\*\*

### 요약

컴퓨터 시스템에서 한글 디스플레이를 위해 자주 사용되는 방법에는 각 한글 자음 및 모음을 5-비트 코드로 정의하고 이들을 조합하여 생성된 각 음절을 2 바이트로 표현하는 표준 조합형 코드와, 일반적으로 한글 표현에 사용되는 한글 음절들 각각에 2 바이트 코드를 부여하여 사용하는 표준 완성형 코드, 그리고 전 세계 대부분의 언어 시스템들의 각 음절에 고유한 코드를 부여하여 컴퓨터가 이들을 통합된 방식으로 표현하고 조작할 수 있도록 한 유니코드가 있다. 임베디드 시스템은 개인용 컴퓨터(PC) 시스템에 비해 낮은 처리 성능과 제한된 저장 공간을 가지는 것이 일반적이다. 그러나 그 용도에 따라 PC에 필적하는 성능을 가지는 경우도 있다. 그러므로 임베디드 시스템은, 한글 디스플레이를 적용할 필요가 있을 경우, 자신의 자원 환경에 적합한 디스플레이 방식을 채택해야 한다. 본 논문은 3.5" TFT LCD 키트인 LP35가 부착된 LN2440SBC 임베디드 보드의 TFT LCD 초기화와 픽셀 디스플레이 함수를 소개한다. 또한 픽셀 디스플레이 함수와 비트맵 폰트를 사용하여 앞서 말한 세 가지 한글 디스플레이 방식에 대한 처리 속도와 필요 메모리 용량을 비교한다. 시험 결과, 표준 조합형은 완성형에 비해 적은 메모리 공간을 필요로 하지만 더 많은 처리 시간을 소모하고 유니코드는 가장 큰 메모리 공간을 필요로 하지만 처리 속도는 가장 빠른 것으로 나타났다

## Hangul Porting and Display Performance Comparison of an Embedded System

Sam Kweon Oh\*, Geun Duk Park\*\*, Byoung Kuk Kim\*\*\*

### Abstract

Three methods frequently used for Hangul display in computer systems are Standard Johab Code in which each of Hangul consonants and vowels is given a 5-bit code and each syllable created by combining them forms a 2-byte code, Standard Wansung Code in which each of all the syllables generally used for Hangul presentation forms a 2-byte code, and Unicode in which each syllable in most of the world's language systems is given a unique code so that it allows computers to consistently represent and manipulate them in a unified manner. An embedded system in general has a lower processing power and a limited amount of storage space, compared to a personal compute(PC) system. According to its usage, however, the former may have a processing power equal to that of the latter. Hence, when Hangul display needs to be adopted, an embedded system must choose a display method suitable for its own resource environment. This paper introduces a TFT LCD initialization method and pixel display functions of an LN2440SBC embedded board on which an LP35, a 3.5" TFT LCD kit, is attached. Using the initialization and pixel display functions, in addition, we compare three aforementioned Hangul display methods, in terms of their processing speeds and amounts of memory space required. According to experiments, Standard Johab Code requires less amount of memory space but more processing time than Standard Wansung Code, and Unicode requires the largest amount of memory space but the least processing time.

Keywords : Standard Johab Code, Standard Wansung Code, Unicode, Embedded System, TFT LCD Display, Performance Comparison

## 1. 서 론

그래픽 LCD, 도트 매트릭스, 전광판과 같은 표시 장치의 핵심은 문자의 표현이다. 단지 몇 자의 문자가 필요한 경우라면 그림으로 처리하면 되지만, 많은 글자들을 표현해야 하는 경우라면 한글 코드를 이용하여 원하는 문자를 표현할 수 있어야 한다. 한글 코드는 컴퓨터의 한글 처리를 위해 정의된 문자 코드 방식으로는 표준 조합형 코드, 표준 완성형 코드, 그리고 유니코드 방식이 있다. 임베디드 시스템은 개인용 컴퓨터(PC) 시스템에 비해 낮은 처리 성능과 제한된 저장 공간을 가지는 것이 일반적이다. 그러나 그 용도에 따라 PC에 필적하는 성능을 가지는 경우도 있다. 그러므로 임베디드 시스템은, 한글 디스플레이를 적용할 필요가 있을 경우, 자신의 자원 환경에 적합한 디스플레이 방식을 채택해야 한다. 한글코드로 표현된 한글을 화면에 출력하기 위해서는 폰트가 필요하다. 폰트[1]란 같은 크기와 모양(체)으로 된 글자 한 벌을 뜻하며, 크게 윤곽선(outline) 폰트와 비트맵 폰트가 있다. 윤곽선 폰트[1][2]는 문자 윤곽을 여러 부분으로 나눈 후 각 영역의 특징점만 추출하여 직선이나 곡선 등을 이용하여 표현 한다. 문자가 선으로 이루어져 있으므로 문자의 유연하고 자연스런 축소 및 확대가 가능하나 글자 크기에 따라 추출한 특징점의 좌표 계산 및, 내부를 채우는 다소 복잡한 과정으로 인해 출력 속도가 느려진다. 이런 연산 과정 때문에 제한된 자원 환경을 가지는 임베디드 시스템에는 적합하지 않다. 비트맵 폰트는 비트맵 방식으로 저장된 문자들을 별도의 연산과정 없이 처리할 수 있으므로 그 처리로 인한 오버헤드가 거의 없어 임베디드 시스템과 같은 소용량의 펌웨어에 적합한 구조를 지닌다.

본 논문은 시랩시스사의 3.5" TFT LCD 키트

인 LP35가 부착된 LN2440SBC 임베디드 보드(S3C2440A CPU, 400MHz)의 TFT LCD 드라이버 제작을 위한 초기화 방법과 픽셀 디스플레이 함수를 소개한다. 또한 픽셀 디스플레이 함수와 비트맵 폰트를 사용하여 표준 조합형, 표준 완성형, 유니코드 방식의 3가지 방식에 대한 한글 출력 처리 속도와 필요 메모리 용량을 비교한다. 시험 결과, 표준 조합형은 완성형에 비해 적은 메모리 공간을 필요로 하지만 더 많은 처리 시간을 소모하고 유니코드는 가장 큰 메모리 공간을 필요로 하지만 처리 속도는 가장 빠른 것으로 나타났다. 특히 국가 간 언어의 호환성을 고려한다면 유니코드 방식이 적합하다.

본 논문의 구성은 다음과 같다. 2장은 표준 조합형 코드, 표준 완성형 코드 및 확장 완성형 코드, 유니코드의 한글 코드와 비트맵 폰트에 대해 설명한다. 3장은 LC2440SBC와 LP35의 초기화 과정과 픽셀 디스플레이 함수 그리고 비트맵 폰트를 이용한 조합형, 완성형, 유니코드의 출력 과정을 설명 한다 4장은 3가지 방식의 출력처리 시간과 필요 메모리 크기를 측정 및 비교하고, 마지막으로 5장에서 결론을 맺는다.

## 2. 한글코드의 종류와 특성

컴퓨터는 기본적으로 영어권 문화에서 만들어진 기계로서 1 바이트 체계인 ASCII 코드만으로 26 개의 알파벳과 특수기호들을 표현할 수 있다. 그러나 11,172 자의 현대 한글은 최대 256 개의 문자를 표현할 수 있는 1 바이트 체계로는 표현할 수 없다. 그래서 여러 종류의 한글 처리 방법이 개발되었고 그 중 표준으로 자리 잡은 것이 표준 조합형 코드와 표준 완성형 코드, 그리고 확장 완성형 코드이다.

표준 조합형 코드(KSC5601-1992)[3][4]는 (그림 1)과 같이 ASCII 코드에서 사용하지 않는 최상위 비트(MSB)를 1로 배정하여 한글임을 나타내고, 한글 1 음절을 초성, 중성, 종성으로 나눠 각각 5비트씩 배정함으로써, 1 음절의 한글을 2 바이트로 표현한다. <표 1>에서 보이는 것과 같이 11,172 자의 한글 음절을 표현할 수 있으며, 훈민정음 창제 원리를 적용하였기 때문에 음절 및 음소 정보를 효과적으로 해석할 수 있다.

※ 제일저자(First Author) : 오삼권  
접수일:2009년 09월 31일, 완료일:2009년 12월 11일

\* 호서대학교 컴퓨터공학부

ohsk@hoseo.edu

\*\* 호서대학교 컴퓨터공학부

\*\*\* 호서대학교 컴퓨터공학부

▣ 이 논문은 2006년 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임(2006-0087)

K/E	초성	중성	종성
1비트	5비트	5비트	5비트

(그림 1) KSC5601 조합형 코드의 구성 원리

<표 1> 표준 조합형 코드 영역

내 용	상위 바이트	하위 바이트
한글(11,172자)	0x84~0xD3	0x41~0x7E 0x81~0xFE
한자(4,888자)	0xE0~0xF9	
특수문자(986자)	0xD9~0XDE	
사용자 정의문자(188자)	0xFE	
TG 특수문자(126자)	0XD4	0x80~0xFE

표준 완성형 코드(KSC5601-1987)[3][4]는 초성, 중성, 종성의 조합으로 이루어진 한글의 약 70 %가 자주 사용되지 않는 점을 이용하여 만들어진 것으로서 한글 음절 2,350 자, 한자 4,888 자, 특수문자 1,128 자, 사용자 영역 188 자와 정의되지 않은 영역 282 자가 배정되어 있다. 이 중에서 한글 영역은 0xB0A1~0xC8FE의 영역을 차지한다. 한글 음절 2,350 자는 조합형으로 만들어 낼 수 있는 음절 수인 11,172 자의 20 %에 불과하지만 평소 사용하는 한글의 99.999 %를 포함하고 있다.

확장 완성형 코드[4]는 마이크로소프트사에서 한글 Windows 95를 발매하면서 제정하여 사용한 코드로서 <표 2>와 같이 표준 완성형 코드의 2,350 자와 표준 완성형 코드에서 표현할 수 없었던 한글 8,822 자를 추가로 배정 하여 현대어 11,172 자를 모두 처리할 수 있는 코드이다. 그러나 기존의 표준 완성형 코드를 '가나다'순으로 배열하고, 사용하지 않는 영역에 추가된 8,822 자를 배치함으로써 한글의 순서가 혼란스럽게 섞이게 된다. 따라서 사전식 순서를 이용한 정렬 및 탐색에 어려움이 있다.

<표 2> 확장 완성형 코드 영역

구 분	범 위
ASCII 영역	0x20~0x7F
표준완성형 한글문자(2,350자)	0xB0A1~0xC8FE
확장 완성형 한글(8,822자)	0x8141~0xB0A0
한자	0xCA1~0xFDFE

다국어 환경의 경우, 각 국가가 국가 간의 호환성을 배려하지 않고 자국 언어의 코드화를 진행함에 따라 상호 호환성의 문제점이 있었다. 이의 해결을 위해 등장한 유니코드는 1989년 컨소시엄 형태로 설립된 기관이자 코드명이다. 유니코드는 1995년 9월 ISO/IEC JTC1 국제표준으로 제정되었다.

ISO/IEC 10646-1의 문자판에는 전 세계의 26개 언어에 대한 문자와 특수기호마다 2 바이트 코드 값을 부여하고 있다. 최대 65,536 자의 문자를 수용할 수 있으며, 주요 국가의 언어를 위해 38,885 자가 할당되어 있고, 사용자 정의 영역으로 6,400 자. 그리고 새로 추가될 언어를 위해 나머지 20,000 여자를 비워 두고 있다.

한글코드는 <표 3>과 같이, 옛한글을 표현하기 위한 조합형 자모 코드 240 자(HANGUL JAMO)와 표준 조합형(KSC 5601) 한글자모 94 자(HANGUL COMPATIBILITY), 확장 완성형의 한글 음절 11,172 자를 가나다순으로 배열해 놓은 완성형(HANGUL)의 3종으로 되어 있다. 또한 한국, 중국, 일본의 한자를 통합한 한중일 통합 한자 영역이 있다.[5]

<표 3> 주요 유니코드 범위 목록

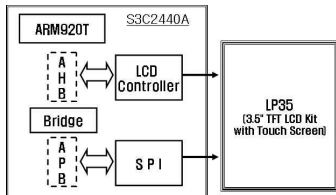
코드 범위	내 용
0x1100 ~ 0x11FF	한글 자모 (240 자)
0x3130 ~ 0x318F	한글 호환성 자모 (94 자)
0x3400 ~ 0x4DBF	한중일 통합 한자 확장-A
0xAC00 ~ 0xD7AF	확장형 한글 음절 (11,172 자)

한글코드로 표현된 한글의 화면출력을 위해서는 폰트가 필요하다. 이 중 비트맵 폰트[1]는 문자를 차지하는 영역을 1, 그 외의 부분을 0으로 표현된 자료를 이진수로 변환하여 저장하는 방식자료를 비트맵 폰트는 별도의 연산과정 없이 1과 0을 판별하여 해당 픽셀을 출력하면 되므로 그 처리의 오버헤드가 거의 없고, 화면 출력 속도가 빠르므로 따라서 임베디드 시스템과 같은 소용량 펌웨어에 적합한 구조자료를 하지만 문자 비트맵 정보의 저장을 위해 많은 메모리 공간이 필요하다. 또한 문자의 확대나 축소 시, 글자 윤곽이 계단 형식으로 거칠어지는 등의 문자 변형에 따른 어려움이 있다.

### 3. TFT LCD 초기화 및 한글포팅

본 논문의 구현은 시랩시스사의 터치 스크린 기능을 가진 3.5" TFT LCD 키트인 LP35와 ARM920T기반의 삼성 S3C2440A CPU가 내장된 LN2440SBC 임베디드 보드를 사용하고, 컴파일러는 ARM Developer Suite v1.2를 사용하며, 임베디드 보드로의 실행 이미지 다운로드를 위해 시랩시스사의 Arm Down v3.8을 사용한다.

LN2440SBC 임베디드 보드[6][7]는 (그림 2)와 같은 구조를 가진다. LN2440SBC가 사용하는 S3C2440A 마이크로프로세서에는 ARM920T 코어와 TFT/STN LCD 컨트롤러, SPI 등이 내장되어 있고 3.5" TFT LCD 키트인 LP35가 부착되어 있다. ARM920T와 LCD 컨트롤러는 고속으로 동작하는 장치들이 연결되는 AHB(advanced high performance bus) 버스를 통해 통신을 하고 SPI(serial peripheral interface)와는 저속의 주변장치와 연결되는 APB(advanced peripheral bus) 버스를 통해 통신을 한다. ARM920T는 SPI[8]를 통해 LCD 장치로 제어 명령을 전송한다.



(그림 2) LN2440SBC 블록 다이어그램

TFT LCD의 구동을 위한 초기화 과정[9]은 (그림 3)과 같이 클럭 속도 설정, 범용 입출력 핀(GPIO)을 LCD와 SPI 용으로의 할당, SPI의 마스터/슬레이브 및 보오 레이트 설정, LCD 컨트롤러의 레지스터 설정을 통한 LCD 기능 선택, SPI를 통한 LCD 장치로의 파워 온(power on) 명령[10] 전달 등이 수행된다.

이로써 기본적인 하드웨어 초기화 과정이 끝나며, 문자의 LCD 출력을 위해 디지털 화면의 최소단위인 픽셀을 출력하는 TFT\_PutPixel 함수를 구현한다. TFT\_PutPixel 함수는 인자 값으로 출력할 픽셀좌표(x, y)와 색상 값을 가진다.

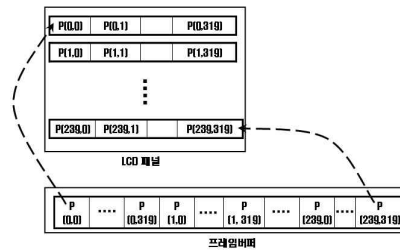
S3C2440A의 LCD 컨트롤러[11]는 시스템 메

모리의 일부를 프레임버퍼로 사용한다. (그림 4)와 같이 프레임 버퍼와 LCD 패널의 좌표체계는 서로 다르다. 따라서 LCD 패널의 XY 직교좌표를 LCD 패널에 대응되는 프레임버퍼의 선형적인 주소의 좌표로 변환해 주고, 프레임버퍼 배열에 그 좌표들의 색상값을 저장하는 루틴이 필요하다. 이는 TFT\_PutPixel 함수에 의해 수행된다.

프레임버퍼의 해당 픽셀 주소에 비디오 데이터를 기록하면, LCD 전용 DMA는 프레임버퍼 메모리에 있는 비디오 데이터를 CPU의 중재 없이 비디오 데이터 포트로 전송하고, 그 결과 LCD 패널에 픽셀이 출력된다.



(그림 3) TFT LCD 구동 순서도



(그림 4) 프레임버퍼와 LCD 패널의 주소체계

다음으로 표준 조합형 코드, 표준 완성형 코드, 유니코드를 이용하여 한글을 출력하는 과정을 설명한다. 확장 완성형 코드는 공식적인 표준 코드가 아니며, 유니코드와 필요 자원 환경은 비슷하나 호환성 및 성능면에서 명백히 떨어지므로 비교 대상에서 제외 하도록 한다.

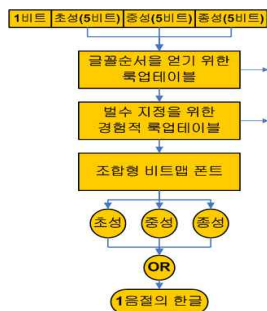
폰트로는 소형 임베디드 시스템 환경에 적합한 비트맵 폰트를 사용한다. 비트맵 폰트는 조합형이나, 완성이나, 아니면 유니코드냐에 따라 구조와 음절의 배열순서가 다르기 때문에 각자에 맞는 전용 폰트를 사용해야 한다. 즉, 표준

조합형 폰트는 초성, 중성, 종성의 비트맵 정보를 가지며, 표준 완성형 폰트는 음절 2,350 자의 비트맵 정보를 가진다. 유니코드 폰트는 음절 11,172 자의 비트맵 정보를 가지고 있다. 본 논문에서 사용할 폰트는 16X16 크기를 가지므로, 1 음절의 표현을 위해 32 바이트가 필요하다.

문자를 디지털 화면에 출력하는 과정은 다음과 같다.

1. 문자 코드를 입력 받는다.(조합형, 완성형, 유니코드)
2. 문자 코드를 가지고 해당 문자의 비트맵 정보가 저장된 폰트 파일의 인덱스를 구한다.
3. 구한 인덱스를 가지고 비트맵 폰트 파일을 참조하여 해당 문자의 비트맵 정보를 구한다.
4. 문자의 비트맵 정보를 TFT\_PutPixel 함수를 이용하여 출력한다.

2번의 경우, 한글 코드의 구조적 특징에 따라 비트맵 폰트에 저장된 글자의 인덱스를 구하는 방법이 달라진다.



(그림 5) 조합형 코드 출력 과정

표준 조합형 코드를 사용할 경우, (그림 5)와 같은 과정을 가진다. 1 번의 룩업 테이블을 통해 조합형 코드를 분할하여 얻은 초성, 중성, 종성의 코드 값에 대응하는 글꼴순서를 얻은 후, 글꼴순서를 인덱스로 사용하여 2 번 룩업(lookup) 테이블을 참조해 글꼴의 별수를 지정한다. 2 번의 별수 획득용 룩업 테이블은 초성에 따라 달라지는 중성의 별수를 지정하는 테이블과, 중성에 따라 달라지는 종성 및 초성의 별수를 지정

하는 테이블로 구성되어 있다. 이렇게 구한 별수를 인덱스를 가지고 조합형 비트맵 폰트를 참조하여 해당 글꼴의 비트맵 정보를 구하고, 초성, 중성, 종성의 글꼴을 OR 연산하여 한 음절의 한글 비트맵 정보를 완성한다.[3]

표준 완성형 코드를 사용할 경우의 한글영역 구조는 <표 4>와 같다. 따라서 0xB0A0를 베이스 주소로 사용하여, 구하고자 하는 표준 완성형 코드에서 베이스 주소와 사용하지 않는 영역의 수만큼을 빼면 글자의 순서를 구할 수 있고, 이 순서를 인덱스로 하여 표준 완성형 비트맵 폰트에서 글자의 비트맵 정보를 구할 수 있다.

<표 4> 표준완성형 코드의 한글 영역 구조

코드 범위	내용
0xB0A0 - 0xB0FF	가~꽁 (96 자)
0xB100 - 0xB19F	160자 (사용 안함)
0xB1A0 - 0xB1FF	꽁~깡 (96 자)
-----	-----
0xC89 0 - 0xC8FF	헬~힉 (96 자)

유니코드를 사용할 경우, 완성된 한글 음절이 0xAC00(가)에서 0xD7A3(힉) 까지 11,172 자가 사전식 순서로 연속적으로 할당되어 있다. 따라서 특정 음절의 코드 값에서 첫 번째 음절인 '가'의 코드 값을 빼주면 배열 순서를 알 수 있으며, 이 배열 순서를 인덱스로 사용하여 유니코드 비트맵 폰트에서 해당 문자의 비트맵 정보를 구할 수 있다.

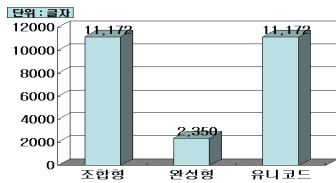
### 4. 성능평가

성능 평가를 위해서 LN2440SBC 임베디드 보드의 코어 속도를 최고속도인 405Mh로 설정한다. 측정은 PWM(pulse width modulation) 타이머를 이용하며, 세밀한 측정을 위해 타이머 클럭을 126.5Mh로 설정하여, 1틱 당 7.9ns의 시간이 소요 되도록 하였다.

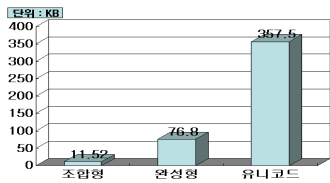
(그림 6)은 표준 조합형 코드, 표준 완성형 코드, 유니코드를 이용하여 한글을 출력할 때, 표현 가능한 한글글자의 수, 필요한 메모리 크기, 문자 출력 처리 시간을 보여준다.

비트맵 폰트는 문자 코드로 비트맵 정보가 저장된 배열의 주소를 계산하며, 이 주소를 이용하여 해당문자의 비트맵 정보를 읽어온다. 이 과정은 룩업테이블들을 이용하여 연산되기 때문에, 실행시간이 일정하다. 따라서, 음절 출력 처리시간은 임의의 음절 '각'이라는 음절의 조합형 코드, 완성형 코드, 유니코드를 인자 값으로 받아 출력 함수를 호출하는 시점부터 출력 함수가 종료되는 시점까지의 시간을 측정한다. 평균값을 얻기 위해, 같은 음절 10 개의 출력 시간을 측정 한 후, 1 음절의 평균 처리 시간을 구한다.

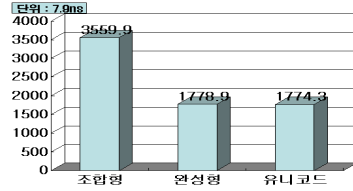
측정한 결과를 살펴보면, 표준 조합형 코드 방식은 메모리를 가장 적게 사용하지만 조성, 중성, 중성의 조합을 위한 처리 시간으로 그 처리 시간이 다른 방식에 비해 약 2 배 정도 소요됨을 알 수 있다. 표준 완성형 코드의 경우에는, 그 처리 시간이 빠르고 많은 메모리를 필요로 하지는 않으나 2,350 자의 한정된 글자만 표현이 가능하다. 마지막으로, 유니코드 방식은 메모리는 가장 많이 사용하지만, 현대 한글 11,172 자를 전부 표현할 수 있으며 그 처리 시간 또한 가장 빠른 것으로 나타났다. 그리고 또 한가지의 장점은 국제 언어 간의 호환성이 매우 우수하다는 것이다. 요약하자면, 메모리 크기가 작고 빠른 문자 출력 처리시간을 필요로 하지 않는다면 조합형 방식이 적합하며, 빠른 처리 속도를 원하지만 메모리 크기가 작다면 완성형 코드 방식이 적합하고, 메모리 크기가 여유가 있을 경우라면, 처리 속도와 호환성이 가장 좋은 유니코드 방식이 가장 적합하다.



a. 표현 가능한 글자 수



b. 비트맵 폰트 메모리 크기



c. 1음절 처리시간

(그림 6) 성능평가결과

## 5. 결론

본 논문은 시랩시스사의 3.5" TFT LCD 키트인 LP35가 부착된 LN2440SBC 임베디드 보드의 TFT LCD 드라이버 제작을 위한 초기화 방법과 픽셀 디스플레이 함수를 소개하였다. 또한 픽셀 디스플레이 함수와 비트맵 폰트를 사용하여 표준 조합형, 표준 완성형, 유니코드의 세 가지 방식에 대한 한글 출력 처리 속도와 필요 메모리 용량을 비교했다. 그 결과 빠른 속도를 위해서는 표준 완성형 코드와 유니코드 방식이 적합하고, 적은 메모리 용량을 위해서는 표준 조합형 코드 방식이 적합한 것으로 나타났다. 특히 국가 간 언어 호환성이 매우 중요한 요소라면, 유니코드 방식의 사용이 적합하다.

## 참고 문헌

- [1] 채영훈, 문승진, "임베디드 시스템을 위한 한글, 일본어, 한자 폰트 제작 및 출력에 관한 연구," 한국 인터넷 정보학회 춘계학술발표대회자료, 제 7권, 제 1호, pp. 155-159, 2006.
- [2] 정근호, 허길, 이영표, 최재영, "임베디드 시스템에서의 폰트처리 기술," 한국정보과학회지, 제20권, 제7호, p. 55-61, 2002.
- [3] 이희문, PIC를 이용한 LCD 모듈 활용, 성안당, 경기도 파주시, 2006.
- [4] 전상훈, C로 구현한 한글코드 프로그래밍 마스터, 골드, 서울, 2005.
- [5] <http://www.unicode.org/>
- [6] Samsung Electronics, "S3C2440A 32-BIT CMOS MI CROCONTROLLER USER'S MANUAL Revision 1," Samsung Electronics, 2004.
- [7] 오삼권, "uC/OS-II를 탑재한 S3C2440A 싱글 보드 컴

퓨터를 위한 모니터링 시스템의 구현," 호서대학교 논문집, 제 26권, pp. 63-69, 2007.

[8] John Catsoulis, Designing Embedded Hardware, 2nd ED의 역서, 한빛미디어, 서울, 2007.

[9] 김병국, 박근덕, 오삼권, "LN2440SBC 임베디드 시스템을 위한 TFT LCD 초기화 및 그래픽스 라이브러리 함수 구현," 한국정보처리학회 춘계학술발표대회 논문집, 제 16권, 제 1호, pp. 639-642, 2009 .

[10] Samsung Electronics, "Product Infomation LTV350 QV -F04," Samsung Electronics, 2005.

[11] 홍건표, 하동호, ARM920T S3C24101, S3C2440A를 이용한 개발자들을 위한 ARM 프로세서, OHM, 서울, 2006.

**오 삼 권**



1980년 : 한국항공대학교 항공전자공학 (학사)  
 1986년 : University of South Florida(USA) (컴퓨터과학 석사)  
 1994년 : Queen's University (Canada)(컴퓨터과학박사)  
 1994년~1995년 : 한국전자통신연구소(선임연구원)  
 1995년~현재 : 호서대학교 컴퓨터공학부(교수)  
 관심분야 : Real-Time and Distributed Systems, OS Communication Protocol, Fault Tolerance

**박 근 덕**



2005년 : 서울대학교 전기컴퓨터공학부 (공학박사)

2006년~현재 : 호서대학교 컴퓨터공학부 조교수  
 관심분야 : 임베디드소프트웨어공학, 서비스 지향 컴퓨팅, XML 응용

**김 병 국**



2009년 : 호서대학교 컴퓨터공학과 (학사)

2009년~현재 : 호서대학교 일반대학원 컴퓨터공학과 (컴퓨터공학 석사과정)  
 관심분야 : Embedded System, Real Time Operating System