LN2440SBC 시스템을 위한 한글 포팅 및 출력 방식 비교

김병국*, 박근덕*, 오삼권*
*호서대학교 컴퓨터공학과
e-mail:greatkuky@naver.com

Hangul Porting and Display Method Comparison for an LN2440SBC System

Byoung Kuk Kim*, Geun Duk Park*, Sam Kweon Oh*
*Dept. of Computer Engineering, Hoseo University

요 약

컴퓨터 디스플레이를 위한 한글 표현 방법에는 한글 음절의 초성, 중성, 종성 각각에 코드를 부여하고, 이를 조합하여 1 음절을 2 바이트로 처리하는 표준 조합형 코드와 각 음절마다 2 바이트 코드를 부여하는 표준 완성형 코드, 그리고 모든 국가의 언어를 표현하기 위한 만국 공통의 문자부호 체계인 유니코드 방식이 있다. 일반적으로 임베디드 시스템은 PC에 비해 상대적으로 속도가 느리고 저장 공간 또한 제한되어 있으나 그 용도에 따라 PC에 필적하는 성능을 가지는 경우도 있다. 따라서 화면에 한글을 출력할 경우, 임베디드 시스템의 자원 환경에 맞는 적합한 방식을 채택해야 한다. 본 논문은 시랩시스(CLabSys)사의 3.5" TFT LCD 키트인 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 결과에 따르면, 표준 조합형은 적은 메모리 공간을 차지하지만 초성, 중성, 종성을 조합하는데 시간이 소요되고, 완성형은 조합형에 비해 출력 처리 속도는 빠르나 모든 음절의 저장을 위해 비트맵 폰트의 용량이 크며, 유니코드는 비트맵 폰트의 용량은 가장 크지만, 국가 간 언어의 호환성을 보장하고 출력 속도 또한 세 가지 방식 중 가장 빠른 것으로 나타났다.

1. 서론

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. 일반적으로 임베디드 시스템은 PC에 비해 상대적으로 속도가 느리고 저장 공간 또한 제한되어 있으나 그 용도에 따라 PC에 필적하는 성능을 가지는 경우도 있다. 따라서 임베디드 시스템의 자원 환경에 맞는 적합한 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 키트인 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 결과에 따르면 빠른 출력 속도를 위해서는 표준 완성형 코드와 유니코드 방식이 적합하고, 적은 메모리 용량을 위해서는 표준 조합형 코드 방식이 적합한 것으로 나타났다. 특히 국가 간 언어의 호환성을 고려한다면 유니코드 방식이 적합하다.

본 논문의 구성은 다음과 같다. 2장은 표준 조합형 코

드, 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 출력처리시간과 필요 메모리 크기를 측정 및 비교하고 마지막으로 5장에서 결론을 맺는다

2. 관련 연구

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. 한글 음절 11,172 자, 한자 4,889자, 특수문자 986자 등을 표현할 수 있으며, 훈민정음 창제 원리를 적용하였기 때문에 음절 및 음소 정보를 효과적으로 해석할 수 있다.

K/E	초성	중성	종성
1비트	5비트	5비트	5비트

(그림 1) KSC5601 조합형 코드의 구성 원리

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, 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.

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 26 개 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 1>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 1> 주요 유니코드 범위 목록

코드 범위	내 용
0x0000 ~ 0x007F	기본라틴어
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)

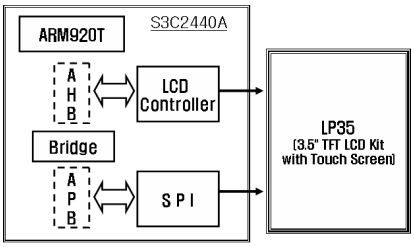
한글코드로 표현된 한글의 화면출력을 위해서는 폰트가 필요하다. 이 중 비트맵 폰트[1]는 문자를 차지하는 영역을 1, 그 외의 부분을 0으로 표현된 자료를 이진수로 변환하여 저장하는 방식이다. 비트맵 폰트는 별도의 연산과정 없이 1과 0을 판별하여 해당 픽셀을 출력하면 되므로 그 처리의 오버헤드가 거의 없고, 화면 출력 속도가 빠르다. 따라서 임베디드 시스템과 같은 소용량 펌웨어에 적합한 구조이다. 하지만 문자 비트맵 정보의 저장을 위해 많은 메모리 공간이 필요하다. 또한 문자의 확대나 축소 시, 글자 윤곽이 계단 형식으로 거칠어지는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

3. 구현

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 사용 한다.

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 장치을 위한 AHB(advanced high performance bus) 버스를 통해 통신을 하고 SPI(serial peripheral interface)와는 저속의 주변장치를 위한 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 클럭 속도 설정, 범용 입출력 핀(GPIO)의 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 수행 된다. 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. LN2440A LCD 컨트롤러[11]는 시스템 메모리의 일부를 프레임버퍼로 사용한다. 프레임버퍼와 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 DMA 컨트롤러는 프레임버퍼 메모리에 있는 비디오 데이터를 CPU의 중재 없이 비디오 데이터 포트로 전송하고, 그 결과 LCD 패널에 픽셀이 출력 된다

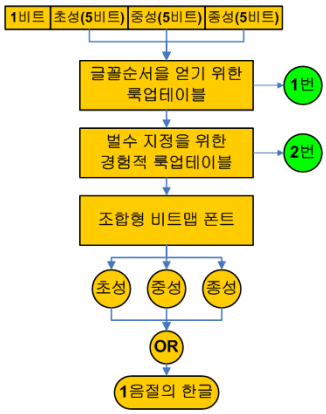
다음으로 표준 조합형 코드, 표준 완성형 코드, 유니코드를 이용하여 한글을 출력하는 과정을 설명한다. 확장 완성형 코드는 공식적인 표준 코드가 아니며, 유니코드와 필요 자원 환경은 비슷하나 호환성 및 성능면에서 명백히 떨어지므로 비교 대상에서 제외 하도록 한다

폰트로는 소형 임베디드 시스템 환경에 적합한 비트맵 폰트를 사용한다. 비트맵 폰트는 조합형이냐, 완성이냐, 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. 코드 유형을 입력 받는다(조합형, 완성형, 유니코드)
2. 코드를 가지고 해당 문자가 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.

표준 조합형 코드를 사용할 경우, (그림 3)과 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 벌수를 지정한다. 벌수 획득용 룩업 테이블은 초성에 따라 달라지는 중성의 벌수를 지정하는 테이블과, 중성에 따라 달라지는 종성 및 초성의 벌수를 지정하는 테이블로 구성되어 있다. 이렇게 구한 벌수를 인덱스를 가지고 조합형 비트맵 폰트를 참조하여 해당 글꼴의 비트맵 정보를 구하고, 초성, 중성, 종성의 글꼴을 OR 연산하여 한 음절의 한글 비트맵 정보를 완성한다.[3]



(그림 3) 조합형 코드 출력 과정

표준 완성형 코드를 사용할 경우의 한글영역 구조는 <표 2>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 2> 표준완성형 코드의 한글 영역 구조

코드 범위	내 용
0xB0A0 - 0xB0FF	가~괋 (96 자)
0xB100 - 0xB19F	160자 (사용 안함)
0xB1A0 - 0xB1FF	괌~깸 (96 자)
----------	----------
0xC89 0 - 0xC8FF	혤~힝 (96 자)

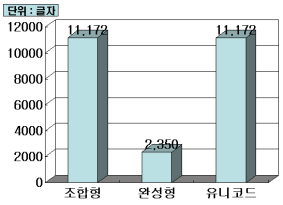
유니코드를 사용할 경우, 완성된 한글 음절이 0xAC00 (가)에서 0xD7A3(힣) 까지 11,172 자가 사전식 순서로 연속적으로 할당되어 있다. 따라서 특정 음절의 코드 값에서 첫 번째 음절인 '가'의 코드 값을 빼주면 배열 순서를 알 수 있으며, 이 배열 순서를 인덱스로 사용하여 유니코드 비트맵 폰트에서 해당 문자의 비트맵 정보를 구할 수 있다.

4. 성능평가

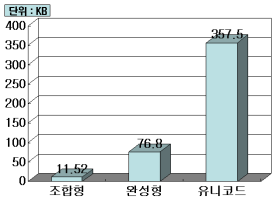
(그림 4)는 표준 조합형 코드, 표준 완성형 코드, 유니코드를 이용하여 한글을 출력할 때, 표현 가능한 한글글자의 수, 필요한 메모리 크기, 문자 출력 처리 시간을 보여준다.

음절 출력 처리시간은 '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 측정한 후, 1음절의 평균 처리 시간을 구한다.

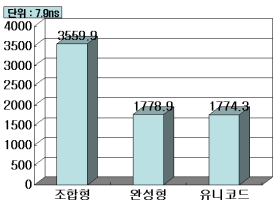
측정한 결과를 살펴보면, 표준 조합형 코드 방식은 메모리를 가장 적게 사용하지만 초성, 중성, 종성의 조합을 위한 처리 시간으로 그 처리시간이 다른 방식에 비해 약 2 배 정도 소요됨을 알 수 있다. 표준 완성형 코드의 경우에는, 그 처리 시간이 빠르고 많은 메모리를 필요로 하지는 않으나 2,350자의 한정된 글자만 표현이 가능하다. 마지막으로, 유니코드 방식은 메모리는 가장 많이 사용하지만, 현대 한글 11,172자를 전부 표현할 수 있으며 그 처리시간 또한 가장 빠른 것으로 나타났다. 그리고 또 한가지의 장점은 국제 언어 간의 호환성이 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 4) 성능평가결과

5. 결론 및 향후계획

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 유니 코드 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. 향후에는 LP35에서 제공하는 터치스크린 기능을 추가하여 보다 편리하고 쉬운 사용자 인터페이스를 제공할 예정이다.

참고문헌

[01] 채영훈, 문승진, "임베디드 시스템을 위한 한글, 일본어, 한자 폰트 제작 및 출력에 관한 연구", 한국 인터넷정보학회 춘계학술발표대회자료, 제 7권, 제1호, pp.155-159, 2006

[02] 정근호, 허길, 이영표, 최재영, "임베디드 시스템에서의 폰트처리 기술", 한국정보과학회지, 제20권, 제7호, pp.55-61, 2002

[03] 이희문, "PIC를 이용한 LCD 모듈 활용", 성안당, 2006

[04] 전상훈, "C로 구현한 한글코드 프로그래밍 마스터", 골드, 2005

[05] http://www.unicode.org/

[06] Samsung Electronics, "S3C2440A 32-BIT CMOS MICROCONTROLLER USER'S MANUAL Revision 1", Samsung Electronics, 2004

[07] 오삼권, "uC/OS-II를 탑재한 S3C2440A 싱글 보드 컴퓨터를 위한 모니터링 시스템의 구현", 호서대학교 논문집, 제 26권, 2007

[08] John Catsoulis, "Designing Embedded Hardware, 2nd ED의 역서", 한빛미디어, 2007

[09] 김병국, 박근덕, 오삼권, "LN2440SBC 임베디드 시스템을 위한 TFT LCD 초기화 및 그래픽스 라이브러리 함수 구현", 한국정보처리학회 춘계학술발표대회 제출, 2009

[10] Samsung Electronics, "Product Infomation LTV350QV -F04", Samsung Electronics, 2005

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