프린터를 위한 다이렉트 프린팅 S/W의 설계 및 구현에 관한 연구

김성주
삼성전자 디지털 미디어 연구소

A Study of Architecture and Implementation of Direct Printing S/W for Printer

Sung Joo Kim
Digital Media R&D Center, Samsung Electronics

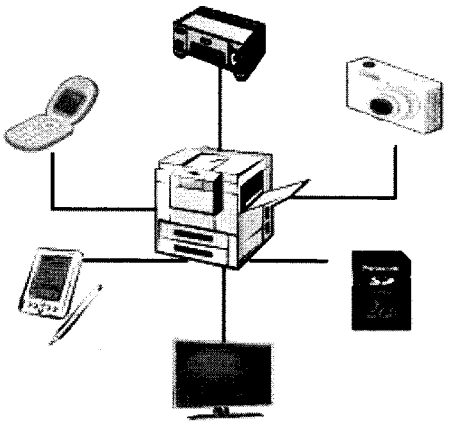
Abstract - ERP(Enterprise Resource Planning) 및 그룹웨어 등의 전자적인 메일, 결재, 문서 관리 수단의 발달에도 불구하고 금융 및 산업계의 프린팅 수요는 매년 급증하고 있다[1]. 특히 근래의 프린터는 스캐닝, 복사, 프린팅, 보안 시스템 등의 다양한 기능들이 융복합화 하면서 사무실에서의 문서 수요와 관리를 책임지는 시스템으로 진화하였다[2]. 다양한 문서 포맷에 대한 프린터의 직접 지원도 ubiquitous의 시나리오에 필수적인 아이템이 되어 가고 있다[2]. 이에 본 논문에서는 인터넷 공간에서 가장 널리 쓰이는 문서 포맷의 하나인 Adobe사 PDF(Portable Document Format) 문서 포맷을 중심으로 프린터에서 직접 문서 해독 및 이미지 처리와 렌더링을 수행하는 S/W 시스템에 대해 그 아키텍처와 구현 예를 보이고자 한다. 아울러 임베디드 시스템에서 direct printing S/W의 고려 사항에 대해 논하고자 한다.

1. 서 론

Ubiquitous의 시대를 맞아 홈 네트워크 및 사무용 네트워크 상에서 사용하는 다양한 디바이스들이 더욱 높아진 CPU 처리 성능과 막강해진 하드웨어 사양을 바탕으로 똑똑해지고 있으며 복잡한 기능을 스스로 처리하는 방향으로 진화하고 있다. 프린터 역시 자체적으로 문서 해독 능력을 갖춤으로써 PC-less 환경에서도 독자적으로 인쇄 가능한 다기능 프린터가 늘어나고 있다. 본 논문에서는 가장 널리 쓰는 장치 독립적 전자 문서 포맷으로서 Adobe사가 개발한 PDF 문서 포맷을 중심으로 임베디드 시스템인 프린터에서 직접 문서의 해독과 인쇄를 수행할 수 있는 direct printing 기술에 대해 논하고자 한다. 본론에서는 direct printing의 개요와 PDF 문서 포맷의 특징, 문서 포맷을 해독하고 대규모 이미지 프로세싱을 수행해야하는 임베디드 시스템의 S/W 아키텍처를 정의한다. 아울러 그 구현 예를 PDF에 대해 보인다. 결론에서는 논의된 내용을 정리하기로 한다.

2. 본 론

2.1 Direct Printing의 개요



<그림 1> Direct Printing 환경의 예

전자 문서의 통상적인 인쇄/출판 과정은 PC를 중심으로 PC에서의 문서 생성 및 편집과 사용자의 인쇄 명령, PC 내에서의 문서 독립적인 프린터 언어로의 변환 후 프린터 전송을 통한 인쇄 과정으로 나뉜다. 프린터 언어의 대표적인 예로는 휴렛팩커드사가 개발한 PCL(Printer Command Language)이 있다. 다양한 PC 어플리케이션을 지원하고 표준화되지 않은 여러 전자 문서 양식을 지원하기 위해 문서 독립적인 프린터 언어가 프린터 제조 회사를 중심으로 정의되어 널리 사용된 것이다. 그러나 PC 환경이 반드시 유용한 것은 아니어서 ubiquitous 환경에서 디바이스간 직접적인 데이터 교환이나 제어를 방해하는 장애물로 작용하는 경우도 있는데 특히 프린터에 있어서 PC를 거쳐야만 한다는 점은 사용자의 사용 환경에 큰 제약을 가하는 요소가 되었다. 이에 전자 문서를 PC 없이 프린터 내에서 스스로 해독하고 인쇄에 필요한 대규모 이미지 프로세싱을 수행하여 인쇄하는 프린터가 속속 등장하였다. 또한 블루투스와 같은 무선환경, 메모리 카드의 채용 등으로 PC-less 환경의 필요성 또한 증대되었다. 그림 1은 이러한 direct printing의 실시 예이다.

2.1.1 Page Description Language의 개요

프린터에서 Page Description Language라 함은 페이지로 구분되는 문서 내용을 그래픽 커맨드의 집합으로 표현하는 Script Language 내지는 Mark-up Language를 말한다[3]. PCL, PostScript, PDF, XPS(XML Paper Specification) 등 여러 가지 방식이 있으며 각 방식마다 특징이 다르다.

PCL은 휴렛팩커드사가 자사용 프린터에 내장할 목적으로 개발한 언어로 프린터와 연결된 외부 디바이스와의 프로토콜로서 설계된 것이다. PostScript는 그래픽 오브젝트의 표현만이 아니라 프로그래밍 언어로서의 조건문, 프로시저, 각종 연산자를 지원한다. PDF는 PostScript의 너무 복잡하고 프로그래밍 언어에 가까운 특징을 제거하고 object 기술자를 채용하였다. 또한 계층적으로 구조화된 문서(Hierarchically Structured Document)로서의 구조를 지닌다. MicroSoft사가 PDF에 대항하기 위해 만든 XPS의 경우 XML을 기반으로 장치 독립적인 문서 포맷으로서 개발되었고 MicroSoft사의 OA 솔루션 및 Windows Vista와 같은 OS에서도 기본으로 지원하고 있다[4].

<표 1> PDL 비교표

PDL명	연도	특징	장점	단점
PCL	1984	프린터 전용 프로토콜. 그래픽요소를 표시하는 command들의 집합으로 표현.	비교적 단순하며 프린터에서의 사용에 특화됨.	프린터용 프로토콜로서 문서 편집, 저장이 고려되지 않음.
PostScript	1982	조건문, 반복문, 프로시저, 연산자 등 interpreter 프로그래밍 언어의 요소를 가짐.	제어문의 활용이 가능하고 다양한 그래픽 요소를 표현 가능.	Adobe사에서 PDF를 개발한 이후 PS는 지원 중단함.
PDF	1993	서로 참조 가능한 hierarchical object tree로써 페이지를 표현. 기본적인 encryption 및 digital signature 지원.	웹상에서 널리 쓰이며 3D, 동영상, interactive feature등 다양한 기능 확장.	문서를 처리하는데 Resource를 많이 요구하며 random access가 많음.
XPS	2006	페이지 내 layout과 그래픽 요소를 기술하는 XML로서 페이지를 표현. ZIP 압축 포맷을 사용.	가장 최근에 정의되고 있는 Spec으로 명확하고 간결함.	아직까지는 MS Windows 환경에서만 사용됨.

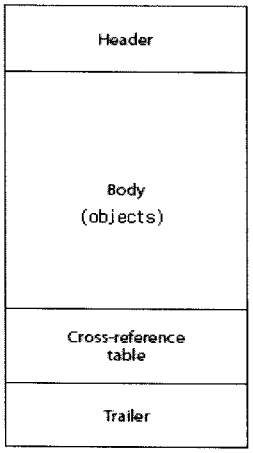
2.1.2 PDF 문서 포맷의 개요

Adobe사에서 개발하여 웹상의 문서 공유 포맷으로 널리 쓰이고 있는 PDF 문서 포맷은 서로 참조 가능한 object를 기본 구성요소로 가진다[5]. 그림 2에는 이와같은 PDF 문서의 기본 구조가 나타나 있다. Header는 PDF 문서의 시작을 알리며 버전 정보가 담겨 있다. Body는 실제 drawing

되는 object들의 모임으로 각 object는 vector graphics, text, image 그리고 이들의 사용방법을 지시하는 command stream 등의 요소를 가지며 이들 object들의 모임으로 하나의 페이지를 구성한다. Cross-reference Table은 index 정보를 포함하여 쉽게 object들을 참조할 수 있도록 한다. 이는 PDF가 단순히 프린터용 PDL이 아니기 때문에 사용자의 자유로운 문서 browsing을 위해 index를 삽입한 것으로 이러한 index를 참조함으로써 쉽게 임의의 페이지에 random access가 가능하다. Trailor는 cross-reference table의 위치를 가리키며 특별한 object에 접근할 수 있도록 하는 참조자의 역할을 한다. 이는 PDF가 한번 작성되었다 할지라도 추후 재수정이 용이하도록 하기 위한 구조이다.

PDF 문서의 자료구조는 위에서 언급한 drawable object들의 database로서 다루어지는데 object들은 문서 파일 내 임의의 위치에 무작위적인 순서로 등장할 수 있기 때문에 반드시 trailor와 cross-reference table을 사용하여 전체 object들의 참조 관계를 파악하며 처리해야 한다.

한 문서 내에는 수십 개에서 수백만 개 이상의 object가 등장하기 때문에 sequencial한 flow에 의한 script 방식의 처리와는 차이가 있다. 먼저 trailor와 cross-reference table을 통해 catalog object 정보를 읽는다. catalog object에는 문서 전체에 적용되는 layout 정보나 그 외 기본 정보들이 포함되어 있고 아울러 page tree object에 관한 정보가 들어있다. 해당되는 page tree object를 찾으면 page tree object를 통해 전체 page object들을 접근할 수 있다. page tree object를 통해 문서 내 모든 page의 시작 object들을 알 수 있다. 임의의 페이지를 읽어 들이거나 인쇄할 때는 해당 page object로부터 참조되는 object들을 따라 실행해나가는 방법으로 이루어진다. 각 page object에는 해당 페이지가 사용하는 resource object 정보가 들어 있는데 이러한 resource에는 image object, font object, graphics object 등이 있다. 이러한 resource와 함께 content stream이라는 정보가 담겨 있는데 이 content stream이 해당 페이지에서 실행할 command list이다. 각 command들은 필요에 따라 resource object들을 입력 데이터로 사용하여 command를 실행함으로써 그래픽 요소들이 페이지 상에 그려진다.



<그림 2> PDF 문서 기본 구조

모든 PDL은 형식의 차이는 있으나 각종 그래픽 요소들을 페이지 상에 그리기 위해 resource와 지시자 또는 command로 이루어져 있다. resource는 image 데이터, font 데이터와 같이 특정 지시자 또는 command가 소모하는 파라미터 및 자료 구조이며, 지시자 또는 command는 선그리기, 텍스트 출력과 같은 특정 동작을 실제 발생시키는 명령어의 역할을 한다. 다만 PDL에 따라 지원 가능한 그래픽 object의 종류 및 기능은 상당히 상이하다. PDF는 object들의 참조관계를 tree구조로 정의하여 resource 및 command를 관계 짓고 실행하는 구조이며 FlateDecode, LZW, JPX, JPEG 등 10여개의 필터(이미지 코덱 및 압축 알고리즘)를 지원하고 투명도(transparency)를 지원하는 등 다양한 기능을 지원한다. 또한 PDF의 장점인 장치 독립적 문서 인쇄를 위해 좌표변환을 위한 좌표 공간 및 칼라를 위한 color space가 user space coordinate, device space coordinate와 user color space, device color space와 같이 각각 나뉘어 있다.

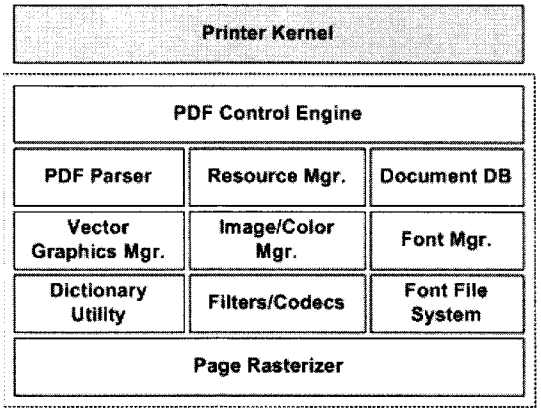
2.2 Direct Printing S/W 아키텍쳐

PDL을 PC에서 구현하는 것과 달리 CE에서 구현하는 데는 많은 고려사항이 있다. 특히 시스템 메모리의 제한과 CPU performance의 제약이 가장 크다. PDF의 경우 페이지 당 그래픽 object의 수, 그래픽 object 자체의 복잡도에 따라 사용하는 메모리와 performance가 크게 차이난다. 이러한 문제를 해결하기 위해 그래픽 object를 rasterizing할 때 한 페이지를 수개~수십개의 sub-band로 나누어 처리한다. 즉 한번에 페이지 전체에 메모리를 할당하는 대신 band에만 할당하고 band boundary 안에 포함되는 object만을 선별적으로 처리한다. 이와 같은 처리를 전체 band 수만큼 반복하여 한 페이지의 이미지를 얻는다. 또한 처리속도를 높이기 위해 전체 문서 내 object를 hash table화하고 hash 검색을 통해 object를 참조한다. 이러한 방법은 대부분의 PDL에서 공통적으로 적용된다.

그림 2는 본 논문에서 제안하는 PDF S/W 아키텍쳐이며 실제 흑백 레이저 프린터에 적용되었다.

본 논문에서 설명하는 CE환경에서의 구현 방법과 최적화 방법은 실제 흑백 레이저 프린터에 실장하여 시험하였다. 흑백 레이저 프린터로는 Samsung ML-4050N을 채택하였으며 시스템 메모리는 64MB, 문서의 저장 및 처리 데이터 캐쉬를 위해 40GB HDD를 내장하였다. 또한 시스템 CPU는 ARM 400MHz를 장착하였다. 표 2는 각종 다양한 PDF 문서를 대상으로 PDF의 engine speed를 측정한 결과이며 평균 24.5PPM(Page Per Minute)의 높은 PDF 문서 처리 속도를 얻었다. 평균은 각 문서별 페이지수를 고려한 가중 평균으로 구하였다. engine speed는 첫 페이지 인쇄 시간을 제외한 나머지 페이지들의 분당 처리 속도를 말한다.

일부 scaling, color blending, bitmap 연산 등 page rasterizing시 하드웨어 가속을 사용할 수 있다면 훨씬 더 높은 인쇄 속도를 기대할 수 있으며 시스템에 FPU가 있어 폴로팅 연산 속도를 가속할 경우도 마찬가지이다. 이는 대부분의 PDL은 높은 그래픽 연산 속도를 요구하며 벡터 그래픽 엔진과 폰트 엔진을 포함하기 때문이다. 또한 PDL에 따라 다양한 이미지 코덱 및 압축 해제 알고리즘을 사용하므로 이들 코덱과 압축 알고리즘의 S/W 최적화나 어셈블리 최적화 또한 처리 속도를 높이는데 큰 도움이 된다.



<그림 3> PDF Emulation S/W 아키텍쳐

<표 2> 제안한 PDF Emulation S/W 성능 측정 결과

문서명	페이지수	인쇄속도(PPM)
Test1.pdf	7	16.74
Test2.pdf	34	40.49
Test3.pdf	56	7.53
Test4.pdf	42	34.31
Test5.pdf	30	27.32
Test6.pdf	4	17.31
Total	173	24.5

3. 결 론

본 논문에서는 direct printing 및 PDL의 개요와 특징을 비교하여 서술하였다. 가장 널리 쓰이는 PDL 중 하나인 PDF를 예로 들어 PDL의 고려사항과 구현 방법, PDF 포맷을 간단하게 설명하였다. 또한 프린터에서의 구현 방법에 대해 PDF emulation S/W를 예로 들어 기술하였다. 이의 메모리 및 처리 속도 면에서의 고려사항과 최적화 방법을 제시하고 실제 구현된 PDF S/W 아키텍쳐와 성능을 나타내었다. 이는 프린터에만 국한되지 않고 일반적인 CE 기기에서 PDL의 처리를 하고자할 때도 그대로 적용된다. 예를 들어 PDF Viewer를 CE환경이나 모바일에서 구현하고자 할 경우도 해당된다. 이를 통해 PDL의 CE 환경에서의 구현과 최적화에 대한 연구에 도움이 되고자 한다.

[참 고 문 헌]

[1] Nikkei Electronics, pp.49-66, 2007년 9월호.
[2] Lisa Purvis, et.al, "Creating personalized documents: an optimization approach", Proceedings of the 2003 ACM symposium on Document engineering, pp.68-77, 2003
[3] http://en.wikipedia.org/wiki/Page_description_language
[4] http://en.wikipedia.org/wiki/XML_Paper_Specification
[5] Adobe Systems Inc., "PDF Reference Version 1.7", pp.47-48, 2006