제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

스마트카드에서의 메모리 관리 기법

정임영*, 전성익*, 정교일**
*한국전자통신연구원 IC 카드연구팀
**한국전자통신연구원 정보보호기반연구부
e-mail : {imyoung, sijun, kyoil}@etri.re.kr

A Memory Handling in Smart Card

Im-Young Jung*, Sung-Ik Jun*, Kyo-Il Chung**
*IC Card Research Team, Electronics and Telecommunications Research Institute
**Information Security Basic Department, Electronics and Telecommunications Research Institute

요 약

스마트카드의 비휘발성 메모리로서 많이 쓰이는 EEPROM 은 사용에 주의해야 할 특성이 있다. 한번에 읽기, 쓰기에 접근할 수 있는 양의 개념으로 페이지가 쓰이면서, 특히 쓰기에서 여러 페이지에 걸친 부분에 접근할 때는 여지없이 기다려야 하는 블록시간이 존재한다. 이 블록 시간으로 하여 EEPROM 의 메모리 관리는 이음새 없는 하나의 덩어리 공간으로 다룰 때 오버로드를 포함하게 되어 특히, 사용자와 직접적인 통신이 되는 장치에 들어가는 EEPROM 일 때는 그 응답시간에 영향을 주게 되는 부분이다. 또한 쓰기에 있어 EEPROM 의 각 부분은 회수 제한이 있기 때문에, 이를 고려해서 본 논의는 비휘발성 메모리로서 EEPROM 을 대상으로 그 효율적인 관리 기법을 제안한다.

1. 서론

최근 스마트카드가 활발히 개발되고 이의 존재가 우리생활에 점차 큰 부분으로 자리를 잡아가고 있다. 스마트카드는 그 자체 크기로 인한 메모리 크기의 제약과 처리능력의 제한으로 일반 컴퓨터의 연산들과 비교하면 그리 크고, 많은 기능을 수행하지는 못한다. 그러나, 중요한 데이터를 보관하고 이들에 대한 안전한 연산을 제공하는 부분으로 그 역할은 중요하고, 이들의 쓰임은 특히 정보보호 부분에서 그 중요도가 커지고 있다. 그런데, 이런 스마트카드의 연산과 응용프로그램 수행을 뒷받침 할 수 있는 연산 수행공간으로서 스마트카드 운영체제는 되도록 정확하고 빠른 수행을 뒷받침해 주어야 한다. 사용자와의 직접 접촉이 있어 사용자의 요구에 빠른 응답을 해야 하는 장치이고 보면, 빠른 처리시간은 무엇보다 우선적으로 배려되어야 한다. 이에 본 논의는 쓰기에 시간이 많이 걸려, 사용자 요구 전체 처리 시간에 영향을 줄 수 있는 비휘발성 메모리에서 보다 빠른 응답시간을 낼 수 있는 관리 기법을 스마트카드의 비휘발성 메모리로 많이 사용되는 EEPROM 을 대상으로 하여 제안한다.

2. 차세대 IC 카드

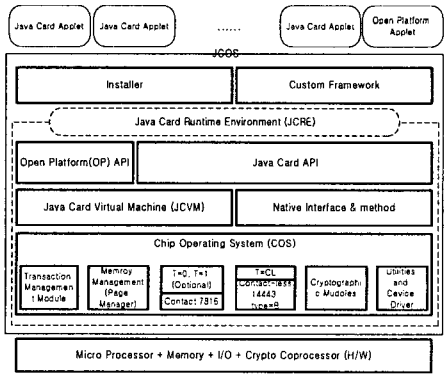

그림 1 차세대 IC 카드 구조

본 논의는 ETRI[1], IC 카드 연구팀이 현재 개발 중인 차세대 IC 카드(Next Generation Integrated Circuit Card-NGIC Card)의 비휘발성 메모리로 쓰이는 EEPROM 의 관리를 제안한다. 차세대 IC 카드는 접촉, 비접촉 표준 ISO 7816, ISO 14443 을 따르고, 그림 1 에서 보듯이 카드운영체제를 이루는 메모리관리 모듈을 비롯한 여러 모듈이외에 암호알고리즘들을 처리할 수 있는 모듈이 탑재되어 있다. 차세대 IC 카드는 자바카드 언어로 만들어진 응용프로그램을 실행할 수 있는 자바카드로서 자바가상기계가 탑재되어 있고 응용 프로그램으로서 애플릿과 카드운영체제를 연결해 주는 Native Interface 도 들어가게 된다. 본 논의는 출발은 차세대 IC 카드의 EEPROM 메모리 관리이지만, EEPROM 을 비휘발성 메모리로 사용하는 모든 장치나 매체에 본 제안은 확대적용이 될 수 있다.

3. 스마트카드에서 비휘발성 메모리로서의 EEPROM

EEPROM 은 읽기나 쓰기에서 한번에 접근할 수 있는 최대량이 있다. 이는 페이지라고 부르는 것의 크기이다. 페이지의 크기는 메모리를 만드는 회사마다 차이가 있는데, 대략 4B, 8B, 16B, 32B, 64B, 128B 의 것이 많이 거론되고 있다. 그런데, 이 페이지는 단순한 양의 단위만은 아니다. EEPROM 은 여러 페이지에 걸친 쓰기에서 한 페이지를 건너 다음 페이지로 이어지는 쓰기 전에 사용자측면에서 기다려야 하는 약 3~10msec 의 블록시간이 존재한다[6][7]. 이 블록 시간은 한 페이지 내의 쓰기 접근에서도 발생하는데, 쓰기 사이의 간격이 약 150usec[6][7]를 넘을 때가 그것이다. 블록시간은 수 초를 다투는 스마트카드 같은 사용자와의 직접 접촉이 빈번한 장치나 매체의 경우 전체 응답시간을 길어지게 만드는 요인이 된다. 그리고, EEPROM 은 100,000 번까지의 쓰기가 허용되는 수명이 있다[6][7][8]. 따라서, 되도록 블록시간을 피할 수 있도록 페이지 단위의 쓰기나 관리와 한 번에 몰아 쓸 수 있는 쓰기 정책이 필요하고, 또한 어느 한 지점의 집중적인 쓰기로 인해 전체 EEPROM 의 수명을 단축시키지 않도록 고른 EEPROM 의 사용이 필요하다.

4. 페이지 단위의 비휘발성 메모리 관리

그림 2 의 경우는 EEPROM 의 고른 사용을 위한 원형 큐나 링크드 리스트 형태의 메모리 관리 기법을 나타낸다. 전체 메모리를 페이지 단위의 관리를 기본으로 해서 메모리에 기록을 할 일이 있으면 되도록 쓰이지 않은 것과 이전에 쓰인 것을 먼저 사용을 하도록 하는 방법이다. 쓰인 것과 쓰이지 않은 페이지를 각각 관리하고 이들의 사용에 선후관계를 정하는 것은 간단한 쓰레기 수집기 역할을 하는 모듈의 구현으로 가능하다. 이런 페이지 단위의 관리를 위해 각 EEPROM 의 페이지가 전원이 나가더라도 간직해야 하는 기본정보를 담을 수 있는 각 EEPROM 페이지 내의 헤더 부분은 그림 3 에서 나타내고 있다. EPROM 의 각 페이지 부분은 어떤 객체가 할당이 되고 또, 이들이 사용하던 공간이 할당 해지되고 하는 일이 일어나고, 꼭 먼저 할당된 EEPROM 의 공간이 먼저 반납되는 것을 보장할 수 없기 때문에 일반적인 경우의 메모리 관리는 링크드 리스트 형식이 될 것이다. 그러나, 먼저 쓰인 부분이 먼저 반납되는 것이 보장이 되는 경우이면 원형 큐 형태의 메모리 관리 형식을 취하면 된다. 원형 큐의 경우는 링크드 리스트 형식의 관리에서 필요했었던 페이지 안의 헤더 부분의 다음 페이지를 가리키는 부분이 없어도 된다. 페이지 헤더 부분은 메모리 제조기술의 발달로 EEPROM 의 크기가 현재 커지고 있고, 또한 페이지의 크기도 되도록이면 처리에 블록시간이 많이 드는 작은 것을 쓰지않기 때문에 한 페이지 당 한 두 바이트 차지하는 페이지 헤더부분이 그리 큰 오버헤드는 되지않는다. 할당이 된 EEPROM 의 페이지들이 이들이 담고있는 데이터의 논리순서에 따라 링크드 리스트로 관리가 된다면, 현재 할당이 되지않은 페이지들은 회수된 페이지들의 순서에 따라 또 하나의 링크드 리스트로 관리가 될 수 있다.

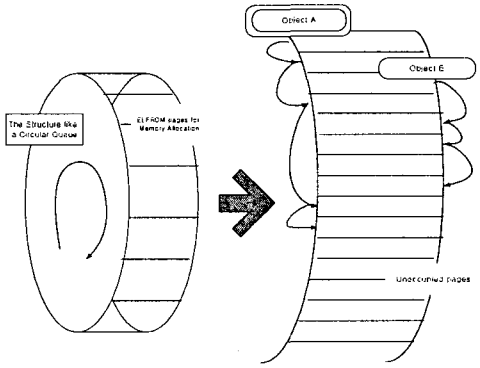

그림 2 EEPROM 의 메모리 운영 구조

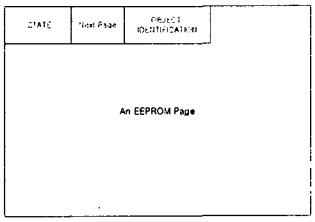

그림 3 EEPROM 페이지 구조

페이지 단위의 쓰기는 쓰기 요구가 몰려있을 때, 쓸 주소들에 속한 페이지별로 분류를 해서 한번의 페이지 접근으로 그 페이지에 속한 것들에 쓰기를 다 행하는 것이고, 한번에 몰아쓰기는 쓰기 요구가 오면 즉

[1] 한국전자통신연구원(Electronics and Telecommunications Research Institute)

시 쓰기를 행하지 않고, RAM 에 저장을 해서 사용하다가 EEPROM 갱신 동기화를 해야 할 필요가 있거나, 이 주소가 들어있는 페이지의 다른 주소에의 쓰기 요구가 들어올 때 한번의 페이지 접근으로 이들 쓰기를 같이 해결하는 방법이다. 이렇듯 페이지 단위로 쓰기를 되도록 한꺼번에 몰아서 하면 블록시간을 줄일 수 있듯이 메모리도 페이지 단위로 할당과 해지를 하는 것이 부담이 적다. 이에 대한 것은 다음 장의 시뮬레이션 결과에서 알 수 있다.

5. 시뮬레이션 결과

표 1 은 시뮬레이션에서 메모리를 할당하고 할당을 해지하는(Free) 회수를 도합 3,000 번으로 했을 시 할당에 성공하고(Allocation Success) 실패하는 회수(Allocation Failure)를 각 메모리를 best-fit 알고리즘에 따라 관리하는 방법과 페이지 단위의 관리 방법을 사용할 시에 같도록 10 번의 실험 회수를 두었을 때의 환경설정 부분이다. 각 실험에서 할당되는 메모리의 크기는 랜덤으로 할당될 바이트 수를 뽑아서 실험을 한 것이다.

표 2 는 표 1 의 10 번의 실험에서 생기는 메모리 조각들의 분포를 나타낸 것이다. 페이지 단위 관리의 경우는 각 페이지 크기에 따라 메모리 조각이 얼마나 생기게 되는지를 페이지의 크기별로 나타내었다. 페이지 단위 관리의 경우는 메모리 조각이 커봐야 페이지 크기보다 작기 때문에 best-fit 의 경우 보다는 전체 가용 메모리 공간이 큰 셈이다. 이는 페이지 단위 관리의 경우는 각 페이지들을 하나의 연속된 공간으로 보기 보다는 그 속에 담기는 내용의 연속성이 있다면 링크로 이음으로서 논리적으로 하나의 공간을 이루도록 할 수가 있기 때문이다. 메모리가 할당이 되고 난 뒤 반납이 될 때도 각 반납되는 페이지의 위치에 상관없이 반납되는 페이지들은 하나의 큰 다시 할당 가능한 메모리 공간이 되는 것이기 때문에 물리적 위치 때문에 재사용될 수 없는 메모리조각 부분이 훨씬 줄게 된다. 이는 좀 큰 메모리 할당 요구가 오더라도 이 요구를 수용할 수 있는 유연한 부분으로 작용을 할 수 있다.

Trial	1	2	3	4	5
Free	1443	1447	1440	1482	1478
Allocation Success	804	1029	881	1356	1464
Allocation Failure	753	494	676	162	58

Trial	6	7	8	9	10
Free	1479	1456	1459	1478	1444
Allocation Success	1334	1028	1124	1366	963
Allocation Failure	187	516	417	156	593

표 1 시뮬레이션 환경 설정

Trial	Approach	Fragments (B)
1	Best-Fit	20, 42, 61, 65, 69, 85, 111, 118, 126, 131, 159, 672, 1095, 1095, 1209
	Page SIZE 4	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,4992
	Page SIZE 8	1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,3,3,3,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,7,7,7,7,7,4896
	Page SIZE 16	1,1,1,1,1,2,2,2,2,2,3,3,3,4,4,4,4,4,5,5,5,5,5,5,6,6,6,6,6,7,7,7,8,8,9,10,10,10,10,12,12,13,14,14,15,15,4784
	Page SIZE 32	1,2,2,2,2,4,4,5,5,5,7,7,9,10,10,10,12,13,14,14,15,16,16,17,17,17,19,19,19,20,20,21,21,21,22,22,22,22,23,24,24,26,28,31,4416
2	Best-Fit	32, 80, 116, 126, 351, 523, 999, 1131, 1276
	Page SIZE 4	1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,4564
	Page SIZE 8	1,1,2,2,2,2,3,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5,5,5,6,6,6,7,7,4496
	Page SIZE 16	2,2,3,4,4,4,4,5,5,5,6,7,8,8,9,9,10,10,11,11,11,11,11,11,12,13,13,13,13,14,14,15,4336
	Page SIZE 32	2,4,4,5,5,6,8,8,9,10,11,11,11,11,12,13,13,14,14,16,16,18,19,20,20,21,23,25,26,27,27,29,29,31,4096
3	Best-Fit	9, 18, 32, 35, 54, 82, 120, 184, 220, 356, 358, 432, 553, 738, 838, 918, 1146, 2655
	Page SIZE 4	1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,8744
	Page SIZE 8	1,1,1,1,2,2,2,2,2,2,3,3,3,3,4,4,5,5,5,5,5,5,6,6,6,7,7,7,7,8660
	Page SIZE 16	1,1,1,1,2,2,2,2,2,3,3,3,4,6,7,7,7,7,8,8,8,10,11,12,13,13,13,13,13,14,14,15,8560
	Page SIZE 32	1,1,1,2,2,2,3,3,4,7,7,8,8,10,11,12,13,14,14,16,16,16,17,18,18,19,22,23,23,24,29,29,29,29,29,31,8288
4	Best-Fit	25, 132, 135, 190, 560, 659, 1651
	Page SIZE 4	1,1,1,1,1,1,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3340
	Page SIZE 8	1,1,1,2,2,2,3,3,3,4,4,4,4,4,5,5,5,6,6,6,6,6,6,7,7,7,3272
	Page SIZE 16	2,3,4,4,4,5,5,6,6,6,6,6,7,7,9,9,9,10,10,11,11,12,12,13,14,14,15,3168
	Page SIZE 32	2,4,4,5,6,6,7,7,9,10,11,12,14,14,16,16,19,20,21,22,22,25,25,26,27,28,29,31,2944
5	Best-Fit	28, 43, 148, 211, 255, 362
	Page SIZE 4	1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,1036
	Page SIZE 8	1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,7,976
	Page SIZE 16	1,2,3,3,3,4,4,5,5,6,6,8,9,9,10,10,10,11,12,12,13,13,13,14,14,14,15,848
	Page SIZE 32	1,3,5,9,10,11,13,13,14,14,16,16,18,19,19,20,20,21,22,22,24,25,26,26,28,28,29,30,31,544
6	Best-Fit	2, 77, 138, 547, 846, 940, 1250, 1396, 1688, 1765
	Page SIZE 4	1,1,1,1,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,3,3,3,3,8704
	Page SIZE 8	1,1,1,1,1,2,2,3,3,3,4,4,5,5,5,5,6,6,6,6,7,7,7,7,7,7,6648
	Page SIZE 16	1,1,1,3,3,4,5,5,5,6,6,7,7,8,8,9,9,10,10,11,13,14,14,15,15,15,8544
	Page SIZE 32	1,1,3,5,6,8,8,9,9,10,10,11,13,14,14,15,15,16,17,19,20,21,21,22,23,23,31,8384
7	Best-Fit	19, 27, 51, 90, 136, 242, 292, 331, 342, 361, 361, 623, 643
	Page SIZE 4	1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3480
	Page SIZE 8	2,2,2,2,2,2,2,3,3,4,4,4,4,5,5,5,5,5,5,5,5,5,6,6,6,6,6,7,7,7,7,7,7,7,7,3360
	Page SIZE 16	2,2,2,2,3,4,6,6,6,7,7,8,8,8,10,10,10,11,12,12,13,13,13,13,13,13,13,13,14,14,15,15,15,15,15,3152
	Page SIZE 32	2,3,6,6,7,7,10,10,12,12,13,13,13,13,14,15,15,15,18,18,19,20,22,24,24,26,27,28,29,29,29,29,30,31,31,2848
8	Best-Fit	3, 3, 10, 12, 14, 24, 28, 68, 173, 279, 808, 857, 1098, 1176
	Page SIZE 4	1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,4496
	Page SIZE 8	1,1,1,1,1,1,2,2,2,2,2,3,3,3,4,4,4,5,5,5,5,5,6,6,6,6,6,7,7,7,7,7,7,4424
	Page SIZE 16	1,1,2,2,2,2,4,5,5,5,6,6,6,7,7,7,8,8,8,9,9,9,9,10,10,11,11,12,13,14,14,14,15,15,4272
	Page SIZE 32	2,2,2,6,6,7,7,8,8,9,9,11,12,14,15,16,16,17,17,18,20,21,21,21,22,23,23,24,25,25,25,26,26,27,29,30,30,31,31,3904
9	Best-Fit	12, 65, 78, 85, 92, 168, 232, 249
	Page SIZE 4	1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,936
	Page SIZE 8	1,1,1,1,2,2,2,3,3,3,3,4,4,4,4,4,4,4,5,5,6,6,6,7,7,7,7,7,864
	Page SIZE 16	1,1,1,2,3,3,4,4,4,4,4,5,6,7,8,8,8,8,9,10,10,11,11,12,12,13,14,14,15,15,15,15,720
	Page SIZE 32	1,2,3,4,4,5,7,8,10,11,12,12,13,14,14,15,15,16,16,17,17,20,20,20,20,22,24,24,24,25,26,27,31,31,448
10	Best-Fit	21, 22, 26, 42, 62, 102, 141, 255, 489, 739, 914, 1190, 1368
	Page SIZE 4	1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,5312
	Page SIZE 8	1,1,1,2,2,2,2,2,2,2,2,2,2,3,3,4,4,4,4,4,4,4,5,5,5,5,5,5,5,6,6,6,6,7,7,5224
	Page SIZE 16	1,1,1,2,2,2,2,2,2,3,4,4,4,4,5,5,5,5,5,6,8,8,10,10,10,10,11,12,12,12,13,13,13,13,14,14,14,15,15,5072
	Page SIZE 32	1,1,2,2,2,3,4,4,4,5,5,10,10,10,12,13,14,15,15,16,17,18,18,18,20,21,21,21,21,24,24,26,27,28,28,29,29,29,29,30,30,4736

표 2 메모리 조각 분포

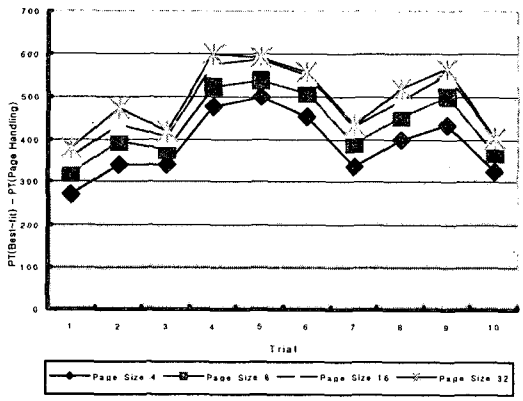

그림 4 페이지 트랜스퍼 회수 차이

그림 4 는 표 1 의 설정에서 EEPROM 을 하나의 큰 이음새 없는 메모리 공간으로 봤을 때의 EEPROM 의 쓰기 성능에 영향을 주는 메모리 쓰기 할당 방법으로 best-fit 의 경우와 본 논의의 페이지 단위의 핸들링 방법에서 페이지를 건너뛰어 쓰게 되는 경우의 회수를 시뮬레이션으로 계산해서 두 방법 사이의 페이지 트랜스퍼의 차이를 나타낸 것이다[2]. 그리고, 이 실험은

[2] 각 할당된 공간은 다 쓰기에 사용이 된다고 했을 때를 가정한 것으로 페이지 트랜스퍼가 있을 시 블록타임을 염두에 두어 페이지 트랜스퍼의 회수를 센 것이다.

페이지 단위의 메모리 관리에서 각 페이지의 크기가 4B, 8B, 16B, 32B 인 경우를 대상으로 한 것이다. 이 페이지 트랜스퍼 회수의 차이는 best-fit 알고리즘을 사용하는 경우의 페이지 트랜스퍼 회수에서 페이지 단위의 관리 기법의 페이지 트랜스퍼의 회수를 뺀 것으로서 각 시도에 대해 모두 200 번 이상의 페이지 트랜스퍼가 best-fit 알고리즘을 그냥 적용할 경우에 더 일어남을 그림 4 에서 볼 수 있다.

페이지 단위의 메모리 관리 기법을 best-fit 알고리즘과 비교를 한 것은 best-fit 알고리즘이 메모리 할당에 가장 많이 쓰이고 메모리 조각의 낭비가 심하지 않은 방법으로 알려져 있기 때문이다. Best-fit 이외에 worst-fit, first-fit 알고리즘 모두 페이지를 고려하지 않은 메모리 할당 방법이므로 best-fit 알고리즘 적용의 경우와 페이지 트랜스퍼 회수는 별 다른 결과를 보이지는 않을 것이다. 페이지 크기가 달라지는 경우에도 10 번의 실험에서 모두 페이지 단위의 메모리 관리가 비슷한 정도로 페이지 트랜스퍼가 best-fit 알고리즘에서 보다는 작게 나타나, 블록시간으로 인한 응답시간의 지연은 덜할 것으로 예측된다.

6. 결론

본 논의는 EEPROM 을 사용하는 매체나 장치를 대상으로 효율적인 메모리 관리 방법에 대한 제안을 한 것이다. 페이지 단위의 관리는 EEPROM 이 가진 특성을 잘 이용한 방법으로 당장 할당요구가 들어왔을 때 사용할 수 없는 메모리 조각을 줄일 수 있고, EEPROM 을 사용 시 쓰기에서 블록시간을 줄임으로서 빠른 처리를 요하는 장치나 매체의 경우 민감한 응답시간을 줄일 수 있다. 또한, 고른 EEPROM 공간의 사용으로 전체 EEPROM 수명의 소진도 둔화 시킬 수 있는 방법이다. 논의의 전개는 자바 스마트카드에서 쓰이는 EEPROM 의 관리로부터 시작되었지만, EEPROM 이 쓰이는 모든 장치에서 이의 관리 방법으로 확대 적용이 될 수 있다.

향후 과제는 페이지 단위의 관리에서 EEPROM 의 각 페이지에 헤더로 잡히는 공간을 줄이는 부분, 메모리에 저장되는 정보의 압축 부분에 대한 고려와 몰아쓰기의 경우, EEPROM 에 아직 기록되지 않은 데이터 부분과 이미 기록된 부분사이의 동기화를 해치지 않는 선에서 좀더 효율적인 쓰기 방법의 개발에 집중될 것이다. 또한, 할당해지 된 공간의 수집을 담당하는 핵심적 기능만을 수행하는 스마트카드 용 압축된 형태의 쓰레기 수집기에 대한 연구도 남아 있다.

참고문헌

[1] Marcus Oestreicher, *Transactions in Java Card*, In 15th Annual Computer Security Applications Conference (ACSAC'99), pp. 291-298. IEEE, 1999.

[2] Clemens H. Cap, Nico Maibaum, Lars Heyden, *Extending the Data Storage Capabilities of a Java-based Smartcard*, Chair for Information and Communication Services, University of Rostock, Germany, unpublished.

[3] Sun MicroSystems, Inc., *Java Card™ 2.1 Application Programming Interface*, Final Revision, June 7, 1999.

[4] Sun Microsystems, Inc., *Java Card™ 2.1 Runtime Environment Specification*, Final Revision, June 7, 1999.

[5] Sun Microsystems, Inc., *Java Card™ 2.1 Virtual Machine Specification*, Final Revision, June 7, 1999.

[6] Atmel, *AT28BV256 Specification*, http://www.atmel.com, 1999.

[7] Atmel, *AT28LV010 Specification*, http://www.atmel.com, 1998.

[8] Zhiqun Chen, *Java Card Technology for Smart Cards* Addison-Wesley, June 2000.

[9] Marek Rusinkiewicz and Amit Sheth, *Specification and execution of transactional workflows*, In W. Kim, editor, Modern Database Systems : The Object Model, Interoperability, and Beyond, ACM Press : Cambridge, Messachussetts, 1994.