임베디드 프로세서를 이용한 스마트 배터리 관리 시스템 구현에 대한 연구

A Study on Implement of Smart Battery Management System using Embedded Processor

오 창 록*, 이 성 원*★
Chang-Rok Oh*, Seong-Won Lee*★

Abstract

Recently portable mobile devices such as smart-phones and notebooks have rapidly increasing demands. Those devices consume more power because they are expected to offer more complex functionality including multimedia features. For these reasons engineering efforts are changing to focus on maximizing energy efficiency within a limited battery capacity instead of increasing computational performance. In this paper, we propose a battery management system using event driven programming technique on a embedded processor. We also show that the proposed system satisfies SBS (Smart Battery Specification) v1.1. The proposed system maintains minimum code size and memory size comparing to those of RTOSs. The proposed system can be also easily incorporated in the conventional RTOSs as a form of firmware.

요 약

최근 휴대 가능한 스마트폰, 노트북과 같은 모바일 기기의 수요가 급격하게 늘어나고 있다. 또한 이러한 기기는 한 제품에 여러 기능을 추가하는 복합화와 대용량의 멀티미디어 기능과 같은 스마트 기능이 주류를 이루면서 더 많은 전력을 소모하게 된다. 이에 따라 기존의 속도와 성능을 향상시키기 위한 노력에서 한정된 배터리 용량을 효율적으로 사용하여 효율을 극대화 시키려는 노력으로 변화하고 있다. 본 논문에서는 임베디드 프로세서를 이용해 이벤트 드리븐 프로그래밍 방식을 사용한 배터리 관리 시스템을 제안하고 제안된 시스템이 SBS(Smart Battery Specification) v1.1을 만족 할 수 있음을 보였다. 제안하는 배터리 관리 시스템의 특징은 기존의 임베디드 시스템에서 실시간 운영체제를 이용한 배터리 관리 시스템에 비해 전체 코드 크기와, 필요한 메모리 크기를 줄 일 수 있다. 또한 Firmware 형태로 구성하여 쉽게 기존의 운영체제에 포함 할 수 있다.

Key words : Battery, Battery Management system, BMS, Event-Driven, Embedded Processor

* 광운대학교 컴퓨터 공학과
(Dept. of Computer Engineering, KwangWoon University)
★ 교신저자 (Corresponding author)
※ 본 논문은 교육과학기술부의 재원으로 한국연구재단(2011-0003501)및 지식경제부가 지원하는 산업융합원천기술개발사업(10039145)과 2010년 광운대학교 교내 학술연구비 지원에 의해 연구되었음
接受日:2011年 12月 01日, 修正完了日: 2011年 12月 19日
揭載確定日: 2011年 12月 26日

Ⅰ. 서론

다양한 휴대용 모바일 기기들의 등장으로 내장 배터리의 용량과 사용효율이 중요한 이슈가 되고 있으며 제한된 배터리 용량을 효율적으로 관리 할 수 있는 배터리 관리 시스템에 대한 연구가 많이 진행 중이다.[1][2][5][7] 현재 휴대용 기기의 대부분은 리튬이온, 리튬 폴리머 이차전지를 사용 하고 있다. 리튬이온 전지는 과 충·방전 시 전지의 수명이 짧아지며

전극에 금속이 석출되고, 간혹 발열에서 발화에 이르는 위험성이 있지만, 이를 방지하면 다른 이차전지에 비해 높은 전압과(4V) 에너지 밀도가 높아 휴대용 기기의 소형 경량화와 장시간 동작을 가능하게 한다. 그럼에도 불구하고 최근의 멀티미디어 등의 고성능 요구와 더불어 배터리의 용량이 정해져 있기 때문에 한번 충전으로 최대한 오래 사용 할 수 있도록 관리하는 배터리 관리 시스템이 필요하다.

배터리 관리 시스템을 구현하는 방법에는 FSM (Finite State Machine)을 이용한 H/W로 구현하는 방법과 실시간 운영체제(Real-Time Operation System)를 이용한 구현이 있다. FSM을 이용한 H/W로 구현하는 방법은 적은 소비전력으로 구현이 가능하고, 대량생산에 용이하여 상업적 제품의 가격적인 장점이 있지만 특화된 기능만을 지원하기 때문에 기능을 추가하기 위해 새롭게 설계해야 하고, 복잡한 기능을 수행하는데 어려움이 있다. 실시간 운영체제를 이용한 배터리 관리 시스템은 개별 셀 전압 측정 및 보호동작, 팩 전압 측정 및 보호동작, 잔존용량, 배터리의 노화상태 측정 등의 배터리 모니터링을 위한 사용자 인터페이스를 쉽고 효율적으로 관리 할 수 있다. 하지만 배터리 관리를 제외한 운영체제를 구동하는데 많은 전력을 소모하여 배터리의 사용 시간이 짧아지는 단점이 있다.

본 논문에서는 실시간 운영체제의 도움이 없이 FSM을 이용한 H/W로 구현이 어려운 상당한 복잡성을 갖는 휴대용 전자기기의 배터리 관리를 위해 SBS-IF(Smart Battery System Implementers Forum)에서 표준화한 SBS v1.1의 구현이 가능함을 보인다. 이를 위해 각각 Command를 필요에 따라 발생하는 Bus Interrupt, 주기적으로 배터리의 상태를 모니터링 하는 Timer Interrupt, Power Device로부터 받는 Device Interrupt를 구분하여 이벤트 드리븐 프로그래밍을 이용한 배터리 관리 시스템을 제안하고, 실시간 운영체제를 이용한 배터리 관리 시스템과 비교하여 전체 프로그램 및 메모리 크기, 소모하는 전력 등의 성능을 비교하여 임베디드 프로세서를 이용하여 SBS v 1.1의 모든 Commands를 구현 할 수 있음을 증명하고, 운영체제에 쉽게 결합하여 구현이 가능함을 보인다.[3][4][8]

본 논문의 구성은 2장의 서두에는 배터리 관리 시스템의 운영방식에 대해서 설명한다. 또한 임베디드 프로세서를 이용한 배터리 관리 시스템의 시스템 모델을 제안하고 배터리 관리 환경 구성, 프로그래밍 모델에 대해서 설명한다. 3장에서는 실험 장비 및 방법과 전체코드 크기와 메모리 사용량을 결과로 제시하고, 마지막으로 4장에서는 결론을 맺는다.

II. 본론

1. 배터리 관리 시스템 운영방식

가. 실시간 운영체제에 의한 운영

실시간 운영체제는 최근 정보기기의 급속한 발전에 따라 소형의 임베디드 시스템에 이식하기 위한 실시간 응답이 가능한 운영체제이다. 일반적인 운영체제(General Purpose Operating System)의 자원관리, 멀티태스팅 지원 등의 기본적인 기능에 추가적으로 실시간 처리를 위해 실시간 스케줄러가 포함되어 있고, 저 전력 설계가 가능하다. 종류로는 Vxworks, Vrtx, pSOS, µC/OS-Ⅱ 등이 있다.

실시간 운영체제에 의한 배터리 관리 시스템은 운영체제에서 제공하는 API(Application Programming Interface)와 운영체제 자체에서 자원의 효율적인 관리가 가능하기 때문에 정확한 배터리의 상태를 예측하고, 사용자 편의에 따른 GUI(Graphical User Interface)의 구현이 쉽다. 또한 추가적으로 구현을 해야 하는 기능이 있을 경우 쉽게 업데이트가 가능하여 유지보수가 용의한 장점이 있다. 반면에 시스템 구성이 커져서 소형의 모바일 시스템 장치에는 부적합하고, 응용 프로그램이 아닌 실시간 운영체제 동작에 의해 소비되는 전력이 큰 비중을 차지해서 소형의 모바일 제품에서 한번 충전으로 장시간 배터리를 유지하는데 힘든 어려운 점이 있다.

나) Dedicated S/W 또는 H/W에 의한 운영

임베디드 시스템에서 특정한 기능만을 수행하는 Dedicated S/W를 이용한 배터리 관리 시스템은 작은 용량으로 모바일 환경에 적합하고, 전력 소모가 적다는 장점을 가지고 있다. 하지만 이미 특정한 기능만을 수행하도록 구현을 하였기 때문에 다양한 응용에 제한이 되고, 다른 기능을 수행하기 위해서는 새롭게 다시 설계해야 된다는 단점을 가지고 있다. 또한 실시간 운영체제와의 결합이 어렵다.

또 다른 방법인 FSM을 이용한 특화된 기능만을 지원하는 H/W로 배터리관리 시스템을 구현하는 방법은 상업적 제품의 가격적인 장점을 위한 방법으로 현재 많이 이용되고 있다. 이를 통해 저 전력 구현이 가능하고, 대량생산이 가능해지는 장점을 가지고 있다. 반면에 한번 H/W로 구성을 할 경우에 시스템 구성 변경에 따른 업데이트가 불가능하고, 특화된 기능만 수행이 가능하기 때문에 복잡하고, 다양한 응용이 힘든 단점을 가지고 있다.

2. 제안하는 배터리 관리 시스템

가. 시스템 모델

그림 1는 제안하는 임베디드 프로세서의 주변블록 구성을 나타내고 있다. 시스템은 임베디드 프로세서 역할을 수행하는 Cortex-M0와 Flash, SRAM, DMA, GPIO 등을 관리하기 위한 AMBA AHB Lite Protocol, 배터리의 상태를 측정하기 위한 아날로그 IP와 측정한 값을 디지털로 바꿔주는 ADC, 외부 IP와 통신을 하기위한 Serial Interface인 I2C Protocol로 구성되어 있다.

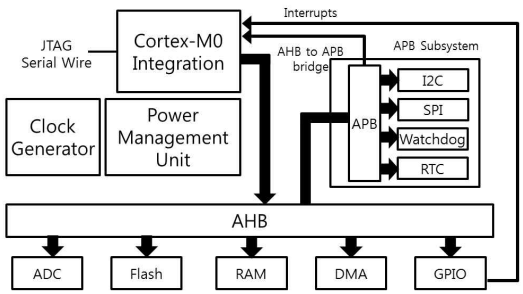


Fig. 1. The Peripheral Block Organization of the Embedded Processor to be Proposed
그림 1. 제안하는 임베디드 프로세서의 주변블록 구성

(1) 시스템 동작

본 시스템은 이벤트 드리븐 모델로 SBS v1.1을 구현하여 펌웨어 형태로 내장되어진다. 모든 동작은 Interrupt에 기초 하여 동작하게 된다. 호스트 시스템으로부터 Command를 입력 받는 Command Interrupt, Timer로부터 일정 시간이 지나면 발생하게 되는 Timer Interrupt, Power Device로부터 받은 Device Interrupt의 3가지 동작을 수행하게 된다. 동작을 수행하지 않을 시에는 Sleep Mode를 유지하여 전력 소모를 최소로 줄인다. Command Interrupt는 SBS v1.1의 모든 Command를 Interrupt로 발생시켜 해당 동작을 수행하기 위해서 배터리의 상태를 측정하는 아날로그 IP로부터 배터리의 전압, 전류, 온도를 측정하여 ADC에서 변환된 디지털 값을 이용하여 배터리의 상태를 업데이트 한다. Timer Interrupt는 배터리의 상태를 실시간으로 감지하기 위하여 일정 시간마다 발생하여 배터리의 상태 업데이트를 위한 Command를 처리한다. Device Interrupt는 ADC에서 아날로그 값이 디지털 값으로 변환 되었을 때 발생하여 변환된 디지털 값을 받을 때와 보호회로에서 배터리의 과 충·방전 시 발생하여 배터리의 상태이상을 알리기 위하여 발생한다.

(2) 메모리 구성

그림 2는 Cortex-M0의 제안한 구성을 구현하기 위한 메모리 구성을 보여준다. 시스템은 128KB의 Flash와 16KB의 SRAM을 가지고 있다. 128KB의 Flash는 Interrupt Vectors, SBS v1.1에 필요한 파라미터를 저장한다. APB Control 영역은 WDT(Watchdog Timer), RTC(Read Time Clock), I2C, SPI, UART 등을 컨트롤 할 수 있는 레지스터를 가진다. AHB Control 영역은 GCR(System Global Control Registers)와 GPIO, Flash Memory, ADC를 컨트롤 할 수 있는 레지스터를 가진다. System Control 영역은 NVIC(Nested Vectored Interrupt Controller)와 ST(System Timer)를 컨트롤 하는 레지스터로 구성된다.

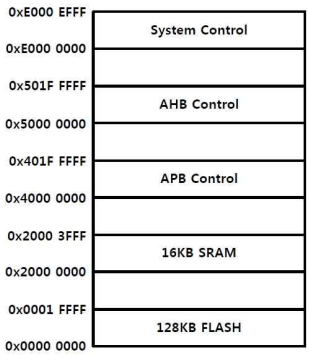


Fig. 2. Memory Map for Implementing the Organization of Suggesting the Cortex0-M0
그림 2. Cortex-M0의 제안한 구성을 구현하기 위한 메모리 구성

나. 배터리 관리 환경 구성

(1) 임베디드 프로세서(Embedded Processor)

본 논문에 ARM사의 Thumb와 Thumb2를 모두 지원하는 저 전력을 목적으로 만들어진 Cortex-M0를 이용하여 배터리 관리 시스템을 구현하였다. Cortex-M0는 인터럽트 처리를 위한 NVIC, AMBA AHB Lite Bus Protocol을 가지고 있다. 인터럽트는 16개의 NMI(Non-maskable Interrupt)와 32개의 Physical Interrupts로 총 48개가 중첩적으로 발생이 가능하다. AMBA AHB List를 통하여 Flash Memory, SRAM 및 주변장치들과 통신을 하고, AHB to APB Bridge를 통하여 속도가 느린 주변 장치와의 통신을 한다. Sleep Mode와 Deep Sleep Mode는 Clock Gating를 통하여 불필요한 Core의 동작을 방지하기 때문에 저 전력 구현이 용이하다.

(2) ADC(Analog-Digital Converter)

정확한 배터리 관리 시스템을 위하여 배터리에서 측정된 전압, 전류, 온도의 아날로그 신호를 디지털 신호로 변환하기 위하여 ADC를 사용하였다.

(3) I2C(Inter Integrated Circuit)

아날로그 시스템과 통신을 위하여 Serial Interface인 I2C를 사용하였다. I2C는 SDA(Serial Data)와 SCL(Serial Clock)를 이용한 통신으로 두 선을 이용하여 마스터가 각각 고유 주소를 가지고 있는 여러 개의 I2C 슬레이브 디바이스 중 주소를 비교하여 해당 슬레이브와 통신을 하는 프로토콜이다. SBS v1.1의 표준은 SMBus(System Management Bus)가 사용되지만 I2C로 대체하여 사용한다. 그림 3은 제안하는 배터리 관리 시스템의 전체 블록 다이어그램이다. 이는 구현된 임베디드 프로세서가 호스트 시스템에 의해 컨트롤 되는 것을 보여준다.

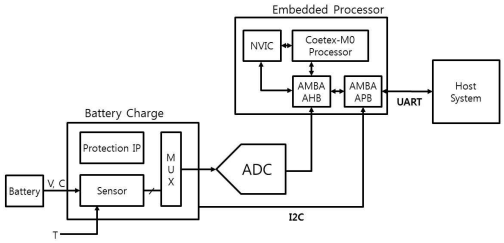


Fig. 3. The Block Diagram of the Battery Management System to be Proposed

그림 3. 제안하는 배터리 관리 시스템의 전체 블록 다이어그램

(4) Protection IP

현재 대부분 에너지 밀도가 높고, 효율이 좋은 리튬이온 배터리를 사용한다. 이러한 리튬이온 배터리는 과 충·방전을 했을 경우에 전지의 수명이 짧아지는 단점을 가지고 있다. 또한 리튬이온 배터리는 휘발성이 강한 액체가 들어 있어 과 충전 시 과전류가 발생할 경우 화기에 접근하면 전지발화가 일어나는 안전상의 위험이 있으므로 반드시 보호회로가 필요하다.

다. 프로그래밍 모델

그림 4는 이벤트 드리븐 프로그램 모델에 따른 전체적인 SBS v1.1의 구현 상태도이다.. 시스템을 초기화하는 'Initialization' 상태, 인터럽트가 요청이 없을 시 최소의 전력을 소모하기 위한 'Sleep Mode' 상태, Bus Interrupt가 들어왔을 경우 동작을 수행하는 'SBS Commands' 상태, Power Device로부터 응답이 있을 경우 처리하는 'Power Event' 상태, 정해진 시간마다 반복적인 동작을 하는 'Repeating Programming' 상태로 구성된다.

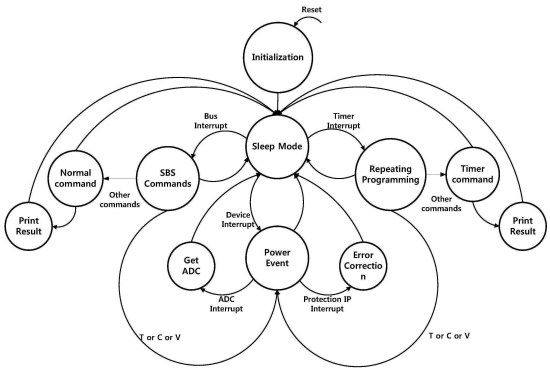


Fig. 4. The State Diagram of whole SBS Implementation according to the Event-Driven Program Model

그림 4. 이벤트 드리븐 프로그램 모델에 따른 전체적인 SBS 구현의 상태도

(1) Initialization

PoR(Power on Reset)이나 Reset시 시스템을 초기화 하는 상태이다. Cortex-M0의 메모리 영역, 배터리 관리 시스템에 필요한 Interface를 초기화 한 후 프로그램에 시작주소로 PC(Program Counter)를 이동하여 동작을 가능하게 하는 역할을 한다. 초기화가 끝나면 바로 'Sleep Mode'로 천이하여 대기하게 된다.

(2) Sleep Mode

저 전력을 위하여 Cortex-M0의 Sleep Mode 기능을 이용하여 Interrupt가 발생 할 때까지 Clock Gating을 통하여 Core의 동작을 멈추어 대기 동안의 전력 소모를 줄이는 상태이다. Interrupt 발생하면 wake-up 신호가 발생하여 Core가 동작하게 되고, Interrupt에 따라서 'SBS Commands', 'Power Event', Repeating Programming' 상태로 천이하게 된다.

(3) SBS Commands

사용자로부터 입력이 있을 시 Interrupt를 통하여 Core에 전달이 된다. 이때 'Sleep Mode'에서 'SBS Commands' 상태로 천이하게 되고, 호스트 시스템으로부터 받은 SBS v1.1 Command를 비교하여 소프트웨어로 구현된 Command를 수행하게 된다. 전압, 온도, 전류의 값을 업데이트 할 때에는 Power Event 상태로 천이하여 ADC로부터 변환된 디지털 값을 가져온다. Command 수행 후 호스트 시스템에 결과를 출력하고 처리가 완료 되었을 경우에 다시 'Sleep Mode'로 천이 된다.

(4) Power Event

APD(Analog Power Device)에서 Interrupt가 발생하였을 경우 'Power Event' 상태로 천이 된다. 배터리에 이상이 있을 때 (과 충·방전) 인터럽트가 발생하여 배터리 상태를 확인하여 문제를 해결한다. 또한 ADC에서 정보를 보낼 준비가 되었을 때 발생하여 해당 값을 가지고 오는 역할을 수행한다.

(5) Repeating Programming

배터리 관리 시스템은 사용자가 원하는 정보를 얻을 때뿐만 아니라, 실시간으로 배터리 상태가 업데이트 되어야 한다. 이러한 경우를 위하여 Timer를 이용하여 일정 시간마다 SBS v1.1의 Command에서 배터리 상태 업데이트에 필요한 Command를 동작시켜 배터리의 상태를 업데이트 한다. 전압, 온도, 전류의 값을 업데이트 할 때에는 Power Event 상태로 천이하여 ADC로부터 변환된 디지털 값을 가져 온다. Command 수행 후 'Sleep Mode'로 천이 된다.

Table 1. The Smart Battery Commands Organization of Implemented

표 1. 구현된 전체 스마트 배터리 함수 구성

#	Command	Description	Interrupt	Access	Data
1	ManufactureAccess()	manufacturer specific	Command	r/w	word
2	RemainingCapacityAlarm()	최소 배터리 용량(%) 설정, 설정 이하 용량시 알람	Command	r/w	mAh or 10mWh
3	RemainingTimeAlarm()	최소 배터리 남은시간 설정, 설정 이하 용량시 알람	Command	r/w	minutes
4	BatteryMode()	배터리 모드 설정(16bit)	Command	r/w	bit flages
5	AtRate()	AtRate 값를 설정하여 현재 배터리의 충/방전률 계산	Command	r/w	mA or 10mW
6	AtRateTimeToFull()	배터리 완충시간 예측	Command	r	minutes
7	AtRateTimeToEmpty()	배터리 남은 작동시간 예측	Command	r	minutes
8	AtRateOK()	AtRate 값에 따라 추가적인 시간 제공(10sec)	Command	r	Boolean
9	Temperature	배터리 셀의 온도값 확인	Command / Timer	r	0.1K
10	Voltage()	배터리 셀의 전압값 확인	Command / Timer	r	mV
11	Current()	배터리 내/외부 전류값 확인	Command / Timer	r	mA
12	AverageCurrent()	배터리 내/외부 1분동안의 평균 전류값 확인	Command / Timer	r	mA
13	MaxError()	에러율	Command	r	percent
14	RelativeStateOfCharge()	남아있는 배터리 예측 용량 확인(현제 배터리 전체 용량대비)	Command / Timer	r	percent
15	AbsoluteStateOfCharge()	남아있는 배터리 예측 용량 확인(설계된 배터리 전체 용량대비)	Command	r	percent
16	RemainingCapacity()	남아있는 배터리 예측 용량 확인	Command / Timer	r	mAh or 10mWh
17	FullChargeCapacity()	충전이 완료 되었을때의 배터리 예측 용량 확인	Command	r	mAh or 10mWh
18	RunTimeToEmpty()	현재 속도로 방전시 남아있는 배터리 시간 확인	Command / Timer	r	minutes
19	AverageTimeToEmpty()	1분동안의 평균값의 속도로 방전시 남아있는 배터리 시간 확인	Command	r	minutes
20	AverageTimeToFull()	1분동안의 평균값의 속도로 충전시 충전이 완료되는 시간 확인	Command	r	minutes
21	ChargingCurrent()	충전시 최대 전류	Command	w	mA
22	CharhingVoltate()	충전시 최대 전압	Command	w	mV
23	BatteryStatus()	배터리 상태 확인	Command	r	bit flages
24	CycleCount()	배터리 충전 횟수	Command / Timer	r	count
25	DesignCapacity()	설계된 배터리 용량	Command	r	mAh or 10mWh
26	DesignVoltage()	설계된 배터리 전압	Command	r	mV
27	SpecificationInfo()	스마트 배터리 스펙 버전 확인	Command	r	unsigned int
28	ManufactureDate()	제조날짜 확인	Command	r	unsigned int
29	SerialNumber()	제품번호 확인	Command	r	number
30	ManufacturerName()	제조업체명 확인	Command	r	string
31	DeviceName()	모델명 확인	Command	r	string
32	DeviceChemisty()	배터리의 화학적 특성 확인	Command	r	string
33	ManufacturerData()	제조업체 정보 확인	Command	r	data

라. SBS v1.1 구현

표 1는 구현된 전체 SBS v1.1의 Commands를 정리한 것이다. SBS v1.1의 구현은 본 내용의 프로그래밍 모델ㅇ 설명한 것과 같은 이벤트 드리븐 방식을 이용하여 구현한다. 따라서 각각의 상태에 맞게 Command Interrupt와 Timer Interrupt로 Commands를 분리하여 구현한다.

(1) Command Interrupt

Command Interrupt는 호스트 시스템에서 전달하는 입력을 수행하는 Command로써 SBS v1.1의 모든 Commands가 여기에 해당된다. 따라서 사용자가 배터리의 정보를 확인 할 수 있는 모든 SBS Commands를 Command Interrupt로 확인 가능 하도록 구현한다.

(2) Timer Interrupt

Timer Interrupt로 구현하는 Commands는 온도, 전압, 전류 값을 Power Device로부터 받아오는 Command와 받은 값을 통해 계산한 잔존용량을 표시할 수 있

는 Command, 배터리의 수명을 알기 위해 배터리의 Cycle를 출력하는 Command로 실시간으로 배터리의 상태변화를 확인하기 위한 Commands만 따로 구분하여 구현한다.

(3) 잔존용량(State of Charge)

배터리의 특성인 EMF(Electromotive Force) 곡선과 잔존용량과의 관계를 이용하여 표2와 같이 Table를 작성하여 잔존용량을 구현한다.[6]

Table 2. Table of Battery EMF Curve
표 2. 배터리 EMF 곡선

전압(V)	2.7000	3.4698	3.5829	3.6612	3.6960
잔존용량(%)	0	2	4	6	8
전압(V)	3.7047	3.7047	3.7134	3.7308	3.7395
잔존용량(%)	10	12	14	16	18
전압(V)	3.7482	3.7569	3.7656	3.7743	3.7830
잔존용량(%)	20	22	24	26	28
전압(V)	3.7830	3.7917	3.7917	3.8004	3.8004
잔존용량(%)	30	32	34	36	38
전압(V)	3.8091	3.8091	3.8178	3.8265	3.8265
잔존용량(%)	40	42	44	46	48
전압(V)	3.8352	3.8439	3.8526	3.8613	3.8700
잔존용량(%)	50	52	54	56	58
전압(V)	3.8787	3.8961	3.9048	3.9222	3.9396
잔존용량(%)	60	62	64	66	68
전압(V)	3.9483	3.9657	3.9744	3.9918	4.0005
잔존용량(%)	70	72	74	76	78
전압(V)	4.0179	4.0353	4.0614	4.0788	4.0962
잔존용량(%)	80	82	84	86	88
전압(V)	4.1049	4.1223	4.1484	4.1658	4.1832
잔존용량(%)	90	92	94	96	98
전압(V)	4.2006				
잔존용량(%)	100				

III 실험

1. 실험 장비 및 방법

배터리 관리 시스템을 관리하기 위한 임베디드 프로세서는 Nuvoton NUC1xx계열의 ARM Cortex-M0 Processor를 가진 Keil MCBNUC1xx 개발보드를 사용 하였고, 개발 중인 배터리의 충·방전 / 보호 / 측정을 할 수 있는 Battery Protection AFE IP와 개발 중인 16bit 해상도를 갖는 Sigma-Delta ADC를 이용하여 실험 환경을 구축하였다. 각각 그림 5의 ①, ②, ③에 나타나 있다. 그림 5는 제안된 배터리 관리 시스템이 구현된 실험 보드이다. 추가적으로 ADC의 결과 값을 FPGA로 Decimation Filter를 설계하여 Filter를 통과한 값으로 임베디드 프로세서에서 받아 연산은 수행한다. 호스트 시스템과 임베디드 프로세서의 통신을 UART를 이용하여 컨트롤 하도록 설계하였다.

전압을 계산하기 위한 식은 수식(1)과 같이 나타낼 수 있다. y는 전압을 나타내며 x는 Decimation Filter를 통해 구한 Decimal 값이 들어간다. 온도를 측정할 때 전압 값으로 결과가 넘어오게 된다. 수식(2)는 전압을 절대온도(K)로 변환하는 식을 나타낸다. x는 측정된 전압 값으로써 식(1)을 통하여 구한다.

$$y = \frac{1}{8360}x - 6.28779 \tag{1}$$

$$T = \frac{1}{\frac{1}{3500} ln \frac{30000x}{29310(3.3-x)} + \frac{1}{273}} \tag{2}$$

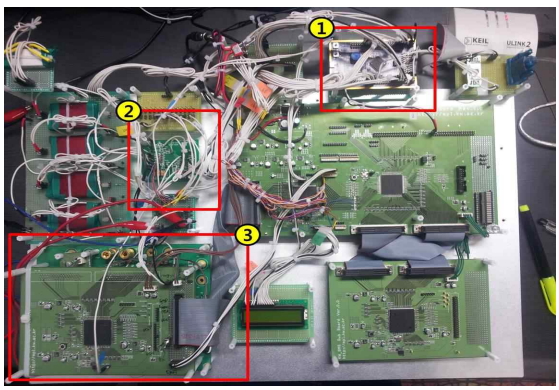


Fig. 5. The Experimental Board in the Proposed Battery Management System is Implemented on the whole
그림 5. 제안된 배터리 관리 시스템이 전체 구현된 실험 보드

결과 측정은 윈도우 하이퍼터미널을 이용하여 결과를 검증하고, 이를 바탕으로 호스트 시스템 어플리케이션을 개발하여 최종적인 확인을 하였다.

2 실험 결과

가. 최소 코드 크기

표 3은 배터리 관리 시스템을 구현한 코드의 크기를 나타내고 있다. 'system_nuc1xx' 파일은 Keil MCBNUC1xx 보드를 초기화하고, 하드웨어 레지스터와 Exception을 정의하는데 사용된다. 'etc file'은 Interrupt와 Serial Interface 등의 주변기기와의 통신을 위해 정의하는데 사용된다. BMS 파일은 SBS v1.1의 Command를 구현한 파일이고, 'main'은 전체

Table 3. Size of Object files
표 3. 오브젝트 파일 크기

File name	Size(KB)
startup	4
main	8
system_nuc1xx	4
BMS	8
etc file	15
total	39
afx file(binary)	**18**

적인 프로그램을 관리하기 위한 파일이다. 실질적으로 구현한 'main'과 'BMS'의 크기를 더하면 8KB로 전체 39KB 크기의 41%에 해당된다. 즉, 최종적으로 만들어 지는 바이너리 파일인 'afx file'의 18KB의 41%의 비중을 차지하고 있다. 이는 최적화를 통해 배터리 관리 시스템에 필요한 부분만 사용함으로써 크기를 더 줄 일 수 있다. 표 4는 실시간 운영체제의 Footprint를 나타내고 있다. 이는 운영체제 자체만을 유지하기 위한 Footprint로 추가적으로 배터리 관리 시스템 프로그램을 작성할 경우에는 적어도 18KB의 41%인 7.2KB가 추가 되어야 한다. 표 4에서 실시간 운영체제의 footprint에 7.2KB를 추가하면 더 많은 크기를 차지하게 된다. 표 4와 비교 하였을 때 임베디드 프로세서를 이용한 이벤트 드리븐 방식으로 구현을 할 경우 코드의 크기가 줄 수 있음을 확인 할 수 있다.

Table 4. Footprint of RTOS
표 4. 실시간 운영체제 Footprint

File name	Footprint(KB)
µC/OS-Ⅱ	5 ~ 24
Nucleus PLUS	5 ~ 40
VxWorks	36 ~ 100
pSOS	12

나. 메모리 최소 사용량

ROM을 사용하는 영역은 RW Data의 영역의 경우 ELF Image와 ROM으로 나뉘기 때문에 RW Data영역의 1/2과 Code영역, RO Data 영역의 합으로 나타내면 총 12,134Byte가 됨을 확인 할 수 있다. 즉, 이 프로그램을 실행하기 위해서 12KB의 ROM 영역이 필요하다.

표 5는 Memory에 적재되어 사용되어지는 공간을 나타낸 것이다. 표에서 보면 실질적으로 프로그램 한 오브젝트 파일보다 라이브러리 파일이 많은 메모리 공간을 차지하고 있는 것을 확인 할 수 있다. 이는 MCBNUC1xx의 기본적인 라이브러리를 가져다가 사용하여 사용하지 않은 부분까지 포함 되는 것을 확인 할 수 있었다. 차후 필요한 부분만 사용하도록 최적화 한다면 메모리 소모를 더욱 더 줄 일 수 있다.

Table 5. Using the Memory Space
표 5. 메모리 사용 공간

Member	Code (inc. data)	Data	RO Data	RW Data	ZI Data
Object file	4,884	964	224	301	2,490
Library	6,620	268	256	0	100
Total	11,504	1,232	480	301	2,590

Table 6. Using the Memory Space in Object File
표 6. 오브젝트 파일 메모리 사용 공간

Member	Code (inc. data)	Data	RO Data	RW Data	ZI Data
startup	52	22	192	0	512
main	2,216	574	0	52	270
system_nuc1xx	392	48	0	8	0
BMS	1,060	184	0	212	1,608
etc file	1,164	136	32	29	100
Total	4,884	964	224	301	2,490

표 6은 표5의 오브젝트 파일의 메모리 사용 공간을 각각의 오브젝트 별로 나타낸 것이다. 실질적으로 프로그램 한 'BMS' 부분과 'main'의 코드 크기가 큰 것을 확인 할 수 있다. 이에 각각의 함수를 주소를 이용한 매핑을 이용하면 메모리 공간을 덜 차지 할 수 있다. 이와 같은 방법으로 최적화하여 최소의 공간을 차지하게 배터리 관리 시스템의 구현이 가능하다.

라. 배터리 관리 동작 결과

SBS v1.1의 모든 Command를 Command Interrupt로 구현을 하였다. 그림 6는 Command Interrupt를

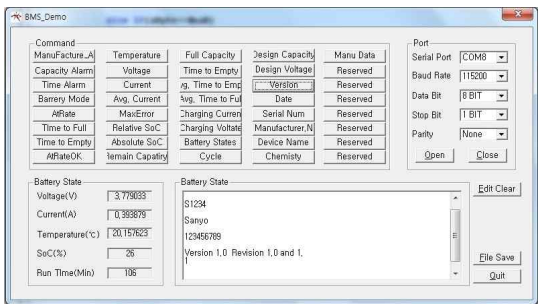


Fig. 6. Result of SBS v1.1 Command Interrupt
그림 6. SBS v1.1 Command Interrupt 결과

위하여 각각의 Command에 해당하는 버튼을 누를 경우 해당하는 Command가 임베디드 프로세서에 전달되어 디코딩 후 해당 Command를 수행하고 결과 값을 UART를 통하여 호스트 시스템 창에 출력한다.

그림 7은 Command Interrupt를 수행 중에 임베디드 프로세서 내부에서 Timer Interrupt에 의해 배터리의 상태가 바뀌는 것을 관찰 할 수 있다. Timer Interrupt를 확인하기 위하여 임베디드 프로세서에서 “Timer Interrupt”를 문자열을 호스트 시스템에 전송하여 Timer Interrupt가 수행 중임을 알려준다. 기본적인 Timer Interrupt에 의해 전압, 전류, 온도 값이 변화 되고, 그에 따라 잔존용량과 남아있는 시간이 변환 것을 확인 할 수 있다.

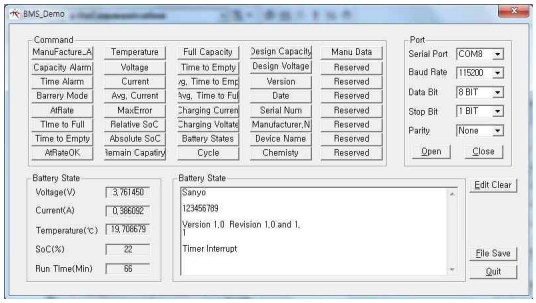


Fig. 7. Result of SBS v1.1 Timer Interrupt
그림 7. SBS v1.1 Timer Interrupt 결과

잔존용량은 측정은 ADC의 정확도에 의해서 결정된다. 본 문의 EMF 곡선과 비교 하였을 때, ADC의 허용오차 안인 0.02%(약 0.03V)의 오차가 생긴다. 이는 잔존용량 계산에 있어 8% 이하의 오차로 균일하게 측정되는 것을 확인하였다.

Ⅳ 결론

본 논문에서는 임베디드 프로세서를 이용하여 이벤트 드리븐 방식으로 제안한 배터리 관리 시스템을 구현 할 수 있음을 보였고, 시스템에 SBS v1.1을 모두 만족함을 확인하였다. 임베디드 프로세서를 이용하여 구현함으로써 실시간 운영체제를 이용한 배터리 관리 시스템에 비해 코드가 줄어드는 것을 결과를 통해 확인 할 수 있었다. 이는 소형의 저용량 모바일기기에 적용하기가 용이하며, FSM을 이용한 H/W로의 구현보다 많은 기능을 포함 할 수 있고 유연하게 추가구현이 가능하다는 장점을 가지고 있다. 또한 Idel 상태일 때 Cortex-M0의 기능인 Sleep Mode로 전환하여 CPU를 Clock Gating을 통해 소모하는 전력이 거의 없어 배터리의 소모를 줄 일 수 있다. 그러나 본 논문에서 사용한 Sigma-Delta ADC와 측정과 보호를 위한 Protection IP가 상용 제품과 다르게 다소 부정확함이 배터리의 상태를 정확하게 측정하는데 어려움이 있었다. 차후 정확성이 보장되는 상용 IP를 이용할 시 정확한 배터리 상태 측정이 가능하다.

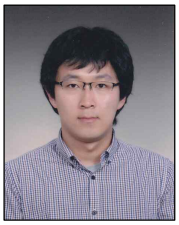
향후 연구과제로 본 논문에서 구현한 배터리 관리 시스템을 최적화 하고, 이를 토대로 배터리를 효율적으로 관리 할 수 있는 알고리즘 개발을 위하여 다양한 배터리 모델링의 연구가 필요하다. 또한 차 후 멀티셀에 적용 가능한 배터리 관리 시스템 연구를 통하여 큰 시스템에 적용하는 연구가 필요하다.

참고문헌

[1] J. Garche, A. Jossen, Battery management systems (BMS) for increasing battery life time, Telecommunications Energy Special, 3, 81 - 84 (2000)
[2] H.J. Bergveld, V. Pop, P.H.L. Notten, Method of estimating the State-of-Charge and of the use time left of a rechargeable battery, and apparatus for executing such a method, Patent WO2005085889 A1 filed February 23 (2005)
[3] “Smart Battery Data Specification” http://sbs-forum.org/specs/sbdat110.pdf, 1998
[4] J. Fischer, R. Majumdar, T Millstein, Task : Language Support for Event-driven Programming
[5] V Pop, J. Bergveld, D Danilov, P.P.L Regtien, P.H.L. Notten, Battery Management System(Accurate State-ofCharge Indication for Battery Powered Applications, Philips Research Book, 9 (2008)
[6] S. Hoenig, H. Singh, T.G. Palanisamy, Method for determining state of charge of a battery by measuring its open circuit voltage, US Patent 6,366,054, filed May 2 (2001)
[7] Texas Instruments, Single-cell Li-ion and Li-Pol battery gas gauge IC for handheld applications (bq junior family), Doc. I.D. SLUS567A (2003)
[8] Texas Instruments, SBS 1.1-Compliant gas gauge for use with the bq29312 (bq2084-V143), Doc. I.D. SLUS732 (2005)

저 자 소 개

오 창 록 (학생회원)



2011년 : 광운대학교 컴퓨터공학과 졸업
2011년 3월~현재 : 광운대학교 컴퓨터공학과 (석사과정)
<주관심분야> 저 전력 SoC, Asynchronous Circuit

이 성 원 (정회원)

1988년 : 서울대학교 제어계측공학과 졸업 (공학사)
1990년 : 서울대학교 제어계측공학과 석사졸업
2003년 : University of Southern California 전기공학과 박사졸업
2005년 3월~현재 : 광운대학교 전자정보공과대학 컴퓨터공학과 부교수
<주관심분야> 미디어프로세서 및 SoC 설계, 영상신호처리, Power-Aware Computing