논문 2017-1-7

다이나믹 디스플레이 회로 구동을 위한 인터럽트 소프트웨어 기법에 관한 연구

이현창*, 권성열**

A Study on the Interrupt Software Technique for Driving Dynamic Display Circuit

Hyun-Chang Lee* Sung-Yeol Kwon**

요 약

본 논문에서는 타이머 인터럽트를 이용해 다이나믹 디스플레이 회로를 구동하는 소프트웨어 기법을 제시하였다. 제시한 방법은 메인 소프트웨어의 진행에 관계없이 독립적으로 디스플레이 소프트웨어가 동작하기 때문에 메인 소프트웨어의 구성이 쉬워지고 알고리즘이 단순해짐에도 불구하고 디스플레이 타이밍은 프로세서의 타이머 회로에서 생성되는 인터럽트에 의해 정밀하게 결정되므로 다이나믹 디스플레이에서 발생되는 얼룩현상이나 깜박임 현상이 제거될 수 있다. 제시한 방법의 효과를 확인하기 위해 4자리의 7-Segment로 구성된 회로에 제시한 알고리즘을 적용해 실험한 결과 메인 소프트웨어에서는 디스플레이 소프트웨어에 대한 특별한 관리가 필요 없으면서도 정확한 디스플레이 전환이 이루어져 메인 소프트웨어의 알고리즘이 매우 간단해짐을 확인하였다.

Abstract

In this paper, the software technique for driving dynamic display circuit using timer interrupt is proposed. As proposed technique makes the display software independent of main program, the algorithm of main software becomes simple. On the other hand, as display timing is fixed by the interrupt from timer circuit of micro-controller, the timing of switching becomes precise. In order to verify the effectiveness of proposed software technique, experimental circuit is designed as Timer Interrupt generator into the 4 MHz micro-controller. Experimental results show that proposed software technique controls switching digits of 7-segment exactly, and main software algorithm become very simple.

한글키워드 : 다이나믹 디스플레이, 인터럽트 소프트웨어, 타이머 인터럽트, 소프트웨어 블록

* 공주대학교 정보통신공학부
(email: hclee@kongju.ac.kr)
** 부경대학교 공과대학 전기과
접수일자: 2017.06.02 수정완료: 2017.06.23

1. 서 론

스마트-폰, 가전제품 및 IT 기기를 비롯한 다

양한 전자기기들에는 사용자에게 기기의 동작상태, 설정상태 등을 표시하기 위한 여러 가지 형태의 표시장치가 사용된다. 이 중 가장 많이 사용되는 것은 LCD(Liquid Crystal Display)를 기반으로 한 표시장치들로서, 예를 들어 냉장고의 경우 기기가 어떤 상태로 설정되어 있는지, 그리고 현재 내부 상태가 어떠한지를 표시하고, 스마트-폰의 경우 TFT-LCD(Thin-Film Transistor LCD)를 사용해 고해상 컬러화면을 표시한다. 이러한 LCD 기반의 표시장치는 중량이 매우 가볍고 두께가 얇아 기기에 내장하기 쉬우며, 대량생산이 유리한 장점이 있지만, 기본적으로 수광형(受光形) 소자, 즉 자체에서 빛을 발생하지 못해 보조조명을 이용해 화면을 표시하기 때문에 매우 밝은 화면을 얻기 어렵고, 대량생산에 유리한 특징으로 인해 소량으로 필요한 경우 오히려 적용하기 어렵다는 단점이 있다.[1,2] 최근 개발된 OLED(Organic Light Emitting Diode) panel의 경우에는 발광형(發光形) 소자이기 때문에 보조조명 없이 필요한 곳만 표시되므로 전력소비가 획기적으로 줄어드는 장점이 있지만, 이 또한 소량으로 사용되는 경우에는 적용하기 어려운 점이 있다. 이에 비해 비록 오래 전에 개발된 소자이기는 하나 7-Segment LED의 경우 최근의 고휘도 LED의 발전에 힘입어 고휘도 7-Segment가 사용가능함에 따라 LCD나 OLED의 적용이 어려운 분야, 즉 소량이면서도 상당한 밝기를 요구하는 분야에서 꾸준히 사용되고 있다. 예를 들어 햇빛이나 실내조명의 밝기를 극복하며 표시되어야 하는 디지털 벽시계나 택시의 요금미터, 엘리베이터의 층 수 표시용, 자동판매기의 요금표시 등에서는 여전히 7-Segment LED가 사용된다.[3,4,5,6]

7-Segment LED는 1개를 단독으로 사용할 때는 회로가 간단하고 구동 소프트웨어도 간단한데 비해 2자리 이상으로 표시 자리가 늘어나면 하드웨어만 이용할 경우 회로가 지나치게 복잡해지므로 최소의 하드웨어와 이를 구동하는 소프트웨어를 함께 이용하는 다이나믹 디스플레이(Dynamic Display) 기법을 사용한다.[7] 그런데, 일반적인 기기의 경우 기기 자체의 동작 중에 디스플레이 또한 꾸준히 표시되고 있어야 하기 때문에 소프트웨어를 구성할 때 기기의 주요 제어 도중 디스플레이 제어를 지속할 필요가 있어 소프트웨어 전체적인 구성이 매우 복잡해지고, 또한 메인 소프트웨어의 처리에 지연이 발생할 경우 디스플레이에 밝기의 얼룩이 발생해 기기의 미관이 크게 떨어진다. 이러한 다이나믹 디스플레이 시간 오차를 줄이기 위해 타이머를 이용한 인터럽트(interrupt)를 이용할 수 있지만, 디스플레이를 처리하는 인터럽트 소프트웨어와 메인 소프트웨어의 데이터 공유 및 타이밍 문제, 소프트웨어 스케줄링 문제 등이 발생하므로 소프트웨어 제작 시 세심한 주의를 요한다.

본 논문에서는 다이나믹 디스플레이를 구동하는 소프트웨어의 구성에 있어 메인 소프트웨어에서는 디스플레이 소프트웨어에 대한 제약을 받지 않으면서도 항상 부드러운 디스플레이가 유지되도록 하기 위한 향상된 소프트웨어 기법을 제시하고자 한다.

2. 다이나믹 구동 소프트웨어의 개선

2.1 다이나믹 디스플레이

그림 1은 1개 7-Segment를 이용해 숫자를 표시하는 회로의 예로서, 이를 구동하기 위한 소프트웨어는 특별한 기법을 필요로 하지 않는다.

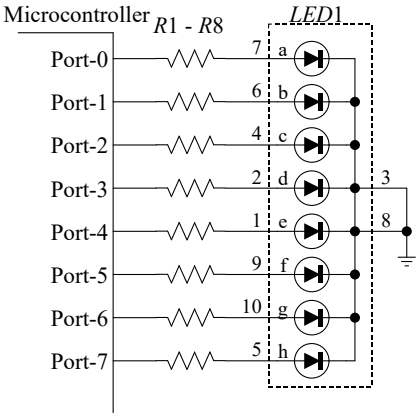


그림 1. 7-Segment LED의 스태틱 디스플레이 예

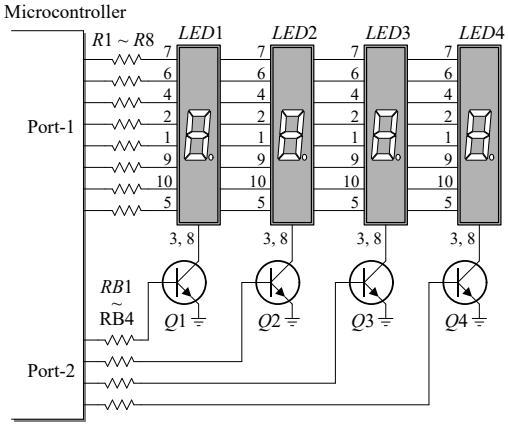


그림 2. 다이나믹 디스플레이 회로의 예

이와 같이 표시회로가 일정하게 표시를 유지하는 것을 분류상 스태틱 디스플레이(static display)라 하며, 이는 표시할 내용을 단지 포트에 출력만 하면 LED는 그대로 켜져 있기 때문에 사용법이 쉽고 LED가 가장 밝게 표시된다. 그러나 스태틱 디스플레이를 이용해 3자리 이상을 구성하려면 부품도 많이 필요하고(7-Segment 1개당 저항 8개) 특히 이를 처리할 프로세서의 포트가 대량으로 필요해 가격상승의 요인이 된다. 이와 같이 여러 자리의 LED를 표시해야 할 경우 다이나믹 디스플레이를 사용하면 부품 수는 크게 줄이고, 특히 필요한 포트 수가 크게 줄어드는 효과가 있으며, 그 예를 그림 2에 나타내었다.

그림 2의 회로에서 4자리의 숫자를 표시하기 위한 저항소자는 1자리 분량인 8개(R1 - R8)가 사용되고 부가적으로 트랜지스터와 이를 구동하기 위한 저항들만 추가되었을 뿐이다. 더구나 마이크로컨트롤러의 포트는 포트-1이라 표시한 8-비트 포트 1개와 포트-2라 표시한 4개 비트만 필요하다.

스태틱 디스플레이 시 필요한 포트 수 P_S와 저항 수 R_S는 각각 식 (1), (2)와 같이 나타낼 수 있다.

$$P_S = D \times 8 \quad (1)$$

$$R_S = D \times 8 \quad (2)$$

여기서 D는 표시할 자릿수(Digits)이다. 이에 비해 다이나믹 디스플레이의 경우 필요로 하는 포트 수 P_D와 저항 수 R_D는 식 (3), (4)와 같고, 여기에 트랜지스터 수 Q_D가 추가된다.

$$P_D = D + 8 \quad (3)$$

$$R_D = D + 8 \quad (4)$$

$$Q_D = D \quad (5)$$

따라서 다이나믹 디스플레이에 사용되는 소자 수의 합 S_D는 식 (6)과 같이 나타낼 수 있다.

$$S_D = R_D + Q_D = 2 \cdot D + 8 \quad (6)$$

표시할 자릿수에 대한 소자의 증가비율을 시뮬레이션 하면 그림 3과 같다.

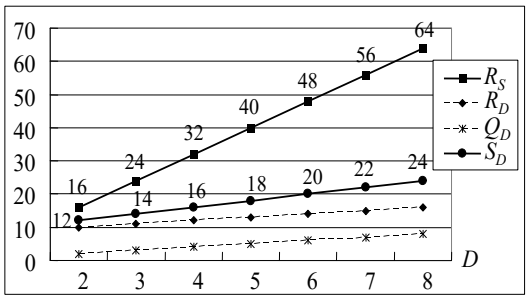
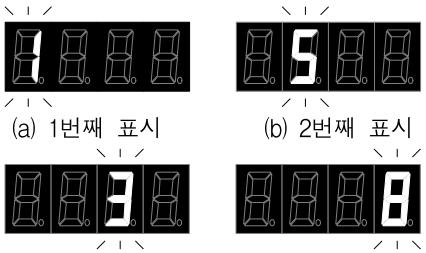


그림 3. 각 디스플레이 방법에 대한 소요 부품 수의 비교



(a) 1번째 표시 (b) 2번째 표시

(c) 3번째 표시 (d) 4번째 표시

그림 4. 다이나믹 디스플레이 회로의 구동 예

그림 3의 시뮬레이션 결과에 의하면 표시 자릿수가 증가할 때 스태틱 디스플레이의 경우 소자수가 급격히 증가하는 반면 다이나믹 디스플레이의 경우 증가폭이 작음을 알 수 있으며, 8자리 표시의 경우 스태틱 디스플레이의 소자보다 다이나믹 디스플레이의 소자 비율은 37.5%에 불과함을 알 수 있다. 만약 개별 트랜지스터를 사용하지 않고 ULN2003A나 UDN2981A 등과 같은 Darlington Transistor Array를 사용하면 7자리에서 8자리까지는 보조 부품이 1개에서 더 이상 증가하지 않아 차이는 더욱 커질 수 있다. 또한 마이크로컨트롤러의 포트 수 증가도 동일한 경향으로 증가하므로 전체적인 코스트 차는 더욱 커진다.

2.2 구동 소프트웨어

그림 4에 다이나믹 디스플레이의 구동 방법 예를 나타내었다. 그림 2의 회로를 이용해 그림 4와 같은 순서로 한 순간에 한 자리씩 표시하고, 이러한 전체의 회전을 $70Hz$ 이상의 빠른 속도로 반복하면 잔상효과에 의해 4개 숫자가 동시에 표시되는 것으로 나타난다.

따라서 각 자리를 전환하는 주파수는 식 (7)과 같이 계산될 수 있다.

$$f_{SW} = f_T \times D \quad (7)$$

여기서, f_T는 전체 회전 주파수이다. 예를 들어 그림 4와 같은 4자리 표시이고, 전체 회전주파수를 $100Hz$로 설정한다면 각 자리를 이동하는 주파수는 이의 4배인 $400Hz$가 되어야 한다.

다이나믹 디스플레이 구동을 위한 소프트웨어의 플로차트를 그림 5에 나타내었다. 그림에서 첫 번째 자리에 표시될 내용을 포트-1에 출력하고 $Q1$에 신호를 인가하면 공통단자로 전류가 유출되는 것은 첫째 7-Segment LED 뿐이므로 오직 이 자리에만 표시된다. 표시된 한 자리의 숫자가 보일 수 있도록 잠시 지연시간을 경과시킨 후 다음 자리 표시를 위해 동일한 과정을 진행한다. 이 때 시간지연에서는 식 (7)에서 계산된 주파수가 발생하도록 설정하며, 이 시간 내에 마이크로컨트롤러는 다른 처리를 진행할 수 있지만, 소프트웨어의 처리 상태에 따라 진행시간이 달라지면 디스플레이 각 자리에 할당되는 시간이 달라지므로 디스플레이에 얼룩이 발생한다.

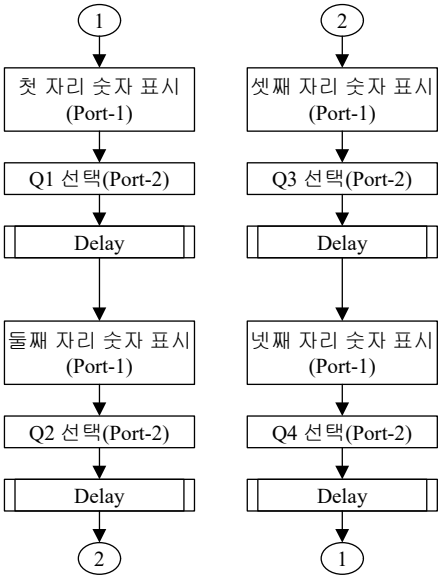


그림 5. 다이나믹 디스플레이 구동 플로차트

2.3 구동 소프트웨어의 개선

2.3.1 소프트웨어 토폴로지

제시하는 다이나믹 디스플레이 구동 소프트웨어의 토폴로지를 그림 6에 나타내었다.

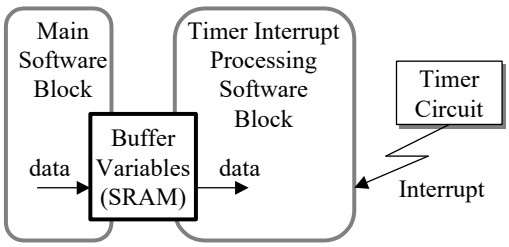


그림 6. 제시하는 다이나믹 디스플레이 구동 소프트웨어의 토폴로지

그림 6은 메인 소프트웨어 블록과 타이머 인터럽트 처리 소프트웨어 블록, 그리고 이들을 중계하는 버퍼용 변수들(SRAM)로 구성되며, 타이머 회로에서는 주기적으로 타이머 인터럽트 처리 소프트웨어를 호출한다. 각 소프트웨어 블록들 사이에는 캡슐화(capsulize) 되어 중계 SRAM을 경유하는 데이터 이외에는 서로 간섭할 수 없도록 구성한다. 타이머 회로에서는 식 (7)에 의해 계산된 주기로 인터럽트가 발생하고, 타이머 인터럽트 처리 소프트웨어에서는 인터럽트가 발생할 때마다 자동으로 7-Segment LED의 자릿수를 전환한다. 즉 그림 4에 나타내었던 자릿수의 전환을 타이머 인터럽트 처리 소프트웨어에서 담당하고, 각 자릿수의 전환은 타이머 인터럽트에 의해 진행된다. 이와 같은 토폴로지로 소프트웨어를 구성한 경우 앞서 소프트웨어에 의해 구현했던 자릿수 전환이 자동으로 진행된다.

한편, 메인 소프트웨어 입장에서는 표시할 데이터가 발생한 경우 해당 자릿수에 대응하는 SRAM 버퍼에 내용을 기록만 하면 이후 인터럽트 처리 소프트웨어에서 자동으로 하드웨어에 표시할 것이므로 소프트웨어 작성이 매우 편리해지고 복잡성이 크게 경감되는 효과가 있다.

2.3.2 타이머 인터럽트 처리 소프트웨어

타이머 인터럽트 처리 소프트웨어는 메인 소프트웨어의 동작과 달리 평상시 동작하지 않다 인터럽트 신호가 발생하면 동작을 개시하고, 현재 해야 할 임무가 완료되면 최대한 빨리 종료해 메인 소프트웨어로 복귀하도록 구성된다. 즉 항시 동작하는 것이 아니라 동작의 개시와 종료가 항상 반복되기 때문에 현재 디스플레이의 진행상태를 기억할 하는 할 수 있도록 조치해야 한다. 이러한 특성을 반영한 인터럽트 처리 소프트웨어의 플로차트를 그림 7에 나타내었다.

타이머 인터럽트가 발생하면 프로세서의 각종 레지스터들을 퇴피(退避, retreat)시키고 현재 몇 번째 자리를 표시할 차례인지 확인해(D_CNT 변수) 이에 따라 분기해 표시 처리를 행한다. 처리가 완료되면 다음 처리할 카운터 값을 설정하고 레지스터들을 복원한 후 메인 소프트웨어로 복귀한다.

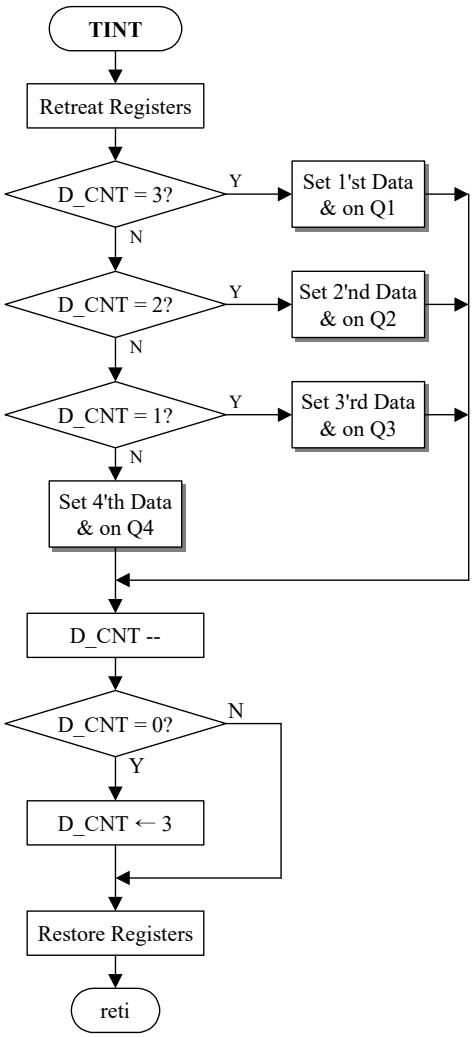


그림 7. 타이머 인터럽트 처리 소프트웨어의 플로 차트

3. 실험 및 고찰

제시한 다이나믹 디스플레이 제어 소프트웨어의 성능을 확인하기 위해 그림 2의 회로와 그림 7의 플로차트를 적용해 그림 8과 같이 4자리의 7-Segment 시계를 구성하였다.

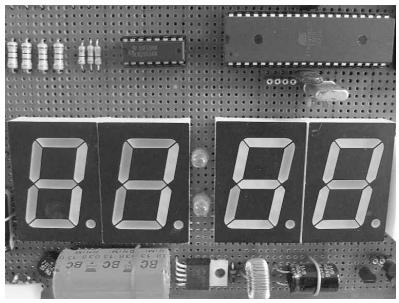


그림 8. 실험용 디스플레이 회로

타이머로부터 발생해야 할 주파수는 식 (7)로부터,

$$f_{SW} = 100Hz \times 4 = 400Hz \quad (8)$$

이며, 마이크로컨트롤러는 ATmega8535[8]를 $4MHz$로 설정해 사용했다. 이 주파수를 이용해 최적의 다이나믹 디스플레이 주파수를 얻기 위한 주파수 조합은 표 1과 같다. 여기서 각 항들은 다음 식들에 의해 계산하였다.

프리스케일러로부터 출력되는 주파수 f_{PRE}는 식 (9)와 같이 계산할 수 있다.,

$$f_{PRE} = \frac{f_{OSC}}{N_{PRE}} \quad [Hz] \quad (9)$$

필요로 하는 타이머의 카운터 값 N_N은,

$$N_N = \frac{f_{PRE}}{2^b} \quad (10)$$

이며, 여기서 b는 카운터의 비트 수로서 8-비트를 적용했다. 또한 최종적으로 발생하는 인터럽트 주파수 f_{INT}는

$$f_{INT} = \frac{f_{PRE}}{N_{CNT}} \quad (11)$$

이며, 따라서 식 (9)를 식 (10)과 (11)에 각각 대입하면 식 (10), (11)과 같은 필요 카운트 숫자와 인터럽트 주파수를 구할 수 있다.

$$N_N = \frac{f_{OSC}}{2^b \cdot N_{PRE}} \quad (10)$$

$$f_{INT} = \frac{f_{OSC}}{N_{CNT} \cdot N_{PRE}} \quad (11)$$

표 1. 타이머 인터럽트 주파수 조합(f_{OSC} = 4*MHz*)

Pre-scaler	f_{PRE} [*KHz*]	Needed Counter N_N	Counter Value N_{CNT}	f_{INT} [*Hz*]
1	4,000	10,000	-	-
8	500	1250	-	-
64	62.5	156.25	156	400.64
256	15.625	39.063	39	400.64
1,024	3.963	9.766	10	390.63

표 1의 결과에서 Pre-scaler가 1, 8일 때 출력되는 주파수를 이용할 경우 8-비트 카운터의 한계를 넘어서므로 적용할 수 없고, 64, 256, 1,024 중에서 선택한다. 따라서 본 실험에서는 그림 9에 나타낸 바와 같은 타이밍 계통을 거쳐 100*Hz*에 유사한 다이나믹 디스플레이 주파수를 생성하였다.

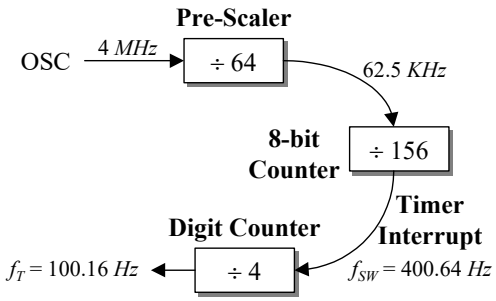


그림 9. 실험용 회로의 주파수 계통도

그림 10은 동작 중인 회로의 각 트랜지스터 구동신호를 관찰한 것으로서, 신호들의 주파수는 각각 정확한 100*Hz*를 유지하면서 전체 주기의 25% duty를 담당함을 알 수 있다.

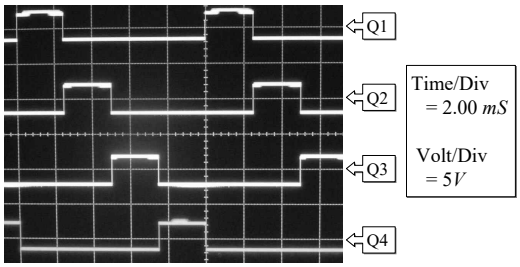


그림 10. 각 트랜지스터 구동파형의 측정결과

4. 결론

본 논문에서는 타이머 인터럽트를 이용해 다이나믹 디스플레이 회로를 구동하는 소프트웨어 기법을 제시하였다. 제시한 방법은 메인 소프트웨어의 진행에 관계없이 독립적으로 디스플레이 소프트웨어가 동작하기 때문에 메인 소프트웨어의 구성이 쉬워지고 알고리즘이 단순해짐에도 불구하고 디스플레이 타이밍은 프로세서의 타이머 회로에서 생성되는 인터럽트에 의해 정밀하게 결정되므로 다이나믹 디스플레이에서 발생되는 얼룩현상이나 깜박임 현상이 제거될 수 있다. 제시한 방법의 효과를 확인하기 위해 4자리의 7-Segment로 구성된 회로에 제시한 알고리즘을 적용해 실험한 결과 메인 소프트웨어에서는 디스플레이 소프트웨어에 대한 특별한 관리가 필요 없으면서도 정확한 디스플레이 전환이 이루어져 메인 소프트웨어의 알고리즘이 매우 간단해짐을 확인하였다.

참고자료

[1] Steven F. Barrett and Daniel J. Pack; Embedded Systems - Design and Applications with the 68HC12 and HCS12, Pearson-Prentice Hall, 2005.

[2] Frank Vahid and Tony Givargis; Embedded System Design - A Unified Hardware/ Software Introduction; John Wiley & Sons, 2002.

[3] 坂巻 佳壽美; 實戰的 マイコン 應用技術; CQ 出版, 1982. 7.

[4] Ronald J. Tocci and Frank J. Ambrosio; Microprocessors and Microcomputers : Hardware and Software, 6th Edition, Pearson-Prentice Hall, 2003.

[5] Helge Seetzen et. al; High dynamic range display systems; Proceedings of ACM SIGGRAPH 2004, Vol. 23, No. 3, pp. 760-768, 2004. 8.

[6] Shek Fai Lau et. al; Dynamic display night light; US Patent US6431719B1, 2002.8.

[7] 이현창, 이종언; 멀티미디어 회로; 상학당 , 2011.9.

[8] Atmel Corp.; ATmega8535(L) - Complete Data-sheet; http://www.atmel.com, 2006.

저 자 소 개

이현창(Hyun-Chang Lee)

1986 단국대 전자공학과 학사
1989 단국대 전자공학과 석사
1996 단국대 전자공학과 박사
1996～2004 국립천안공업대학 정보통신과 부교수.
2005～현재 국립 공주대학교 공과대학 정보통신공학부 교수.

<관심분야: 멀티미디어회로, 전동기제어회로, 마이크로프로세서, 임베디드 소프트웨어 >

권성열(Sung-Yeol Kwon)

1990 수원대 전자재료공학과 학사
1993 경북대 전자재료전공 석사
2000 경북대 센서공학과 박사
1994～1998 안동과학대학 전자계산과 조교수.
2000～현재 국립 부경대학교 공과대학 전기과 교수.

<관심분야: MEMS sensor, 전자재료, 전기제어계측, 신재생에너지 >