임베디드 시스템 타이머 동기화

이형봉*
*강릉원주학교 컴퓨터공학과
e-mail:hblee@gwnu.ac.kr

Synchronization of Timers in Embedded Systems

Hyung-Bong Lee*
*Dept of Computer Science, Gangneung-Wonju National University

요 약

임베디드 시스템 구성 요소 중 가장 빈번하게 사용되는 디바이스들 중의 하나로 타이머를 들 수 있다. 대부분의 임베디드 시스템 MCU 들은 3~5 개의 타이머를 제공하므로 설정시간 별로 독립된 타이머를 할당하여 구현할 수 있다. 그러나 TDMA 기반 무선 통신 프로토콜 등과 같이 10 개 이상의 타이머를 필요로 하는 경우가 있는데, 이런 경우에는 하나의 물리적 타이머에 여러 개의 논리적 타이머를 구현해야 한다. 이 때, 논리적 타이머들 사이에 물리적 타이머의 분해능에 따른 오차가 존재하여 시간 동기화 오차를 유발하는 원인이 된다. 이 논문에서는 이러한 논리적 타이머 사이에 존재하는 오차를 자세하게 분석하여 제기하고, 이를 극복하는 방안을 모색한다.

1. 서론

유비쿼터스 센서 네트워크(USN: Ubiquitous Sensor Network)에 이어, 최근 스마트폰의 광범위한 보급으로 MCU(Micro Controller Unit)를 탑재한 임베디드 시스템의 활용과 응용에 대한 수요가 급속도로 팽창하고 있다. 이로 인하여, 과거부터 늘 중요하게 여겨져 왔던 소프트웨어에 대한 중요성이 오늘날에 와서는 단순한 응용 소프트웨어를 벗어나, 하드웨어를 통제하는 시스템 소프트웨어에 대한 중요성으로 그 무게 중심이 급격하게 이동하고 있다. 즉, 대부분의 소프트웨어들이 운영체제나 개발도구가 제공하는 단순한 API(Application Programming Interface) 기반만으로는 구현되지 못하고, 하드웨어를 직접 구동시켜야만 구현이 가능하게 된 것이다. 이러한 하드웨어로는 인터페이스 방법이 매우 다양한 각종 센서뿐만 아니라, MCU 내에 내장된 타이머(timer), ADC(Analog Digital Converter), UART(Universal Asynchronous serial Receiver Transmitter) 등을 들 수 있다. 이 논문에서는 특별히 TDMA 기반 무선 통신 프로토콜[1] 등 엄격한 시간 동기화를 요구하는 어플리케이션이 MCU의 타이머를 다루는 과정에서 나타난 타이머 펄스 동기화 문제를 제기하고 이를 극복할 수 있는 방안을 제안하되, 그 사례로써 Atmega2560 MCU[2] 및 AvrStudio[3]]를 활용한다.

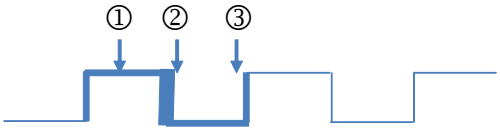
2. 일반적인 타이머 구조 및 사용법

보통 타이머는 입력되는 클록 펄스에 맞추어 타이머 카운터(레지스터)를 1씩 증가시켜 일정 값(최대 혹은 설정값)에 도달하면 인터럽트를 일으킨다. 따라서 소프트웨어는 원하는 시간을 클록 펄스(틱, tic) 수로 변환하여 카운터 레지스터에 설장한 후 인터럽트를 기다리게 된다.

▣ 타이머 클록 펄스

(그림 1)에 타이머 클록 펄스와 소프트웨어가 타이머 카운터에 클록 수를 설정하는 시점 ①, ②, ③을 보였다. 이 그림에서 타이머의 내부 카운팅 시점이 펄스의 폴링 에지(Falling Edge)이므로, 타이머 카운트에 대응되는 정확한 시간이 경과하기 위해서는 시점 ②에 타이머 카운터 설정이 이루어져야 한다. 만약 설정이 시점 ①이나 ③에서 이루어지면 첫 번째 카운터 하나가 일찍 감소되기 때문에 원하는 시간보다 그만큼 짧은 시간을 카운트한다. 평균오차는 클록 주기의 1/2, 최대 오차는 클록 주기만큼이 된다. 그러나, 이 오차는 대부분의 개발자들이 예상하고 있는 부분이고, 주파수가 높은 CPU 클록을 사용하는 타이머를 사용하는 경우 클록 주기가 매우 짧기 때문에 무시해도 무방하다. 이를테면 7.3728MHz 클록의 경우 클록 주기가 0.135us이기 때문에 이정도의 오차라면 거의 모든 어플리케이션을 만족한다.

그러나 이러한 오차 개념을 분주(Prescaling)된 클록 펄스나 외부 입력 클록 펄스에 그대로 적용하여, 시행착오 끝에 뒤 늦게 설계변경을 실시하는 경우가 있다.



(그림 1) 타이머 클록 펄스 및 타이머 카운터 설정 시점

▣ 타이머 클록의 분주

대다수의 타이머는 CPU 클록을 공유하므로 클록 주파수가 매우 높다. 이 주파수를 그대로 사용하는 경우 인터럽트가 너무 자주 일어 CPU에게 부담이 되므로 일정 비율로 주파수를 낮추어 사용하는 것이 좋다. 이를테면 7.3728Mhz 클록에 8 비트 카운터를 사용하면 최대 설정 가능 클록 수가 256이고, 이에 대응되는 시간은 약 34*us*에 불과하여 ms 단위의 타이머가 필요한 경우 많은 수의 인터럽트를 치러야 한다. 만약 7.3728MHz를 1/32로 분주한다면 8 비트 카운터로 최대 약 1ms까지 설정할 수 있다.

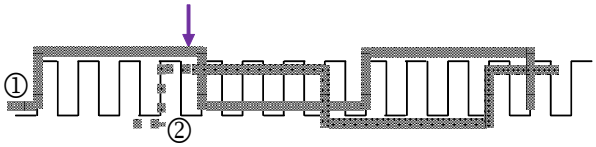
3. 타이머 펄스 동기화 문제

여기서 다루는 타이머 펄스 동기화 문제는 분주된 클록 주파수로 인한 클록 주파수가 낮은 외부 클록 소스에서 발생한다.

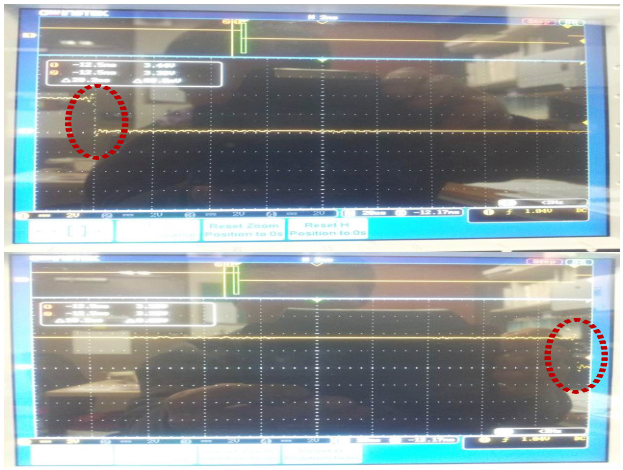
▣ 분주된 타이머 펄스의 동기화 문제

(그림 2)와 같이 1/8로 분주된 타이머 펄스에서 화살표 표시 지점에서 타이머 카운터가 설정되면 분주된 타이머 펄스가 원래의 ① 번 파형(펄스 패턴)이 ② 번 파형으로 적응되는 것으로 기대하고 있으나, 실은 그렇지 않다. 즉, 분주된 타이머 펄스 오차가 원래 펄스의 8 배(분주 배율)가 되는 것이다. (그림 3)에 Atmega2560의 32768Hz를 1/8로 분주한 4096Hz 타이머에 50 클록(약 10.3ms)을 설정했을 때의 오차 측정 모습을 보였는데, 오실로스코프의 그리드 분해능이 20*us*이므로 이 그림에서 보이는 오차는 약 180*us*이다. 이는 4096Hz 클록 주기 244*us*에 가깝다.

이 문제를 해결하기 위한 방법으로 타이머를 중지시켰다가 다시 시작시키는 방안을 들 수 있다. 그러나 이 방법은 해당 타이머에 이미 설정되어 진행 중이던 경과 시간에 영향을 미치기 때문에 적용할 수 없다.



(그림 2) 분주된 타이머 클록 펄스의 예



(그림 3) 32768Hz를 1/8로 분주한 타이머의 50 클록 수 동안의 오차 측정(상:최대, 하:최소)

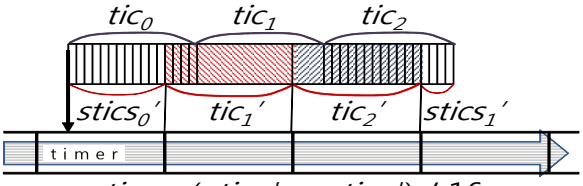
▣ 외부 클록 소스에서의 동기화 문제

대부분의 MCU는 파워절약 모드를 제공하는데 이 모드에서는 CPU 클록이 정지하여 CPU를 깨워줄 타이머를 사용할 수 없다. 이 경우를 위하여 외부 클록을 사용하는 타이머가 제공되고, 파워절약 모드 지속 시간은 짧지 않으므로 외부 클록의 주파수는 낮다. 즉, 외부 클록은 주파수가 낮으am로 분주를 하지 않더라도 오차가 크다.

4. 타이머 펄스 동기화 문제 해결 방안

분주되거나 클록 주기가 긴 타이머 펄스에 동기를 맞추기 위한 기본 방법은, 설정하고자 하는 주 타이머의 인터럽트 주기에 맞추어 클록 수를 설정(Setting)하고, 그 시점까지의 시간을 분해능이 더 높은 보조 타이머로 측정하는 것이다. 첫 번째 카운터 중 인터럽트까지 소요된 시간을 제외한 시간은 타이머 종료 후 처음에 사용했던 보조 타이머로 카운팅한다.

(그림 4)에 1/16으로 분주된 타이머 펄스 동기화 예를 보였다. 이 그림에서 어플리케이션이 요구한 클록 수는 tic_0~tic_2의 3 개이고, 그 중 tic_0는 주파수가 16 배인 보조 타이머를 위해 주 타이머의 펄스 주기(인터럽트)까지의 $stic_0$ 개와 주 타이머 종료 후 $stic_1$ 개의 클록 수로 분리되어 처리되고 있다.



(그림 4) 주 타이머와 보조 타이머의 개념

5. 결론

32768Hz를 8 분주한 4096Hz의 주 타이머와 921600Hz의 보조 타이머를 사용해서 주 타이머의 50 클록을 실험한 결과 최대 오차가 4*us*를 넘지 않았다. 이는 주 타이머만 사용했을 때의 180*us*보다 획기적으로 개선된 결과이다.

참고문헌

[1] Hyung-Bong Lee, et. al, "A Lightweight Lap Time Measurement System for Alpine Ski Sport using a TDMA-based Linear-Wireless Snensor Network", International Journal of Distributed Sensor Network (IJDSN), Volume 2012, pp. 1-15, Mar. 2012

[2] http://www.atmel.com/devices/ATMEGA2560.aspx, accessed Mar. 2013

[3]http://www.atmel.com/tools/ATMELSTUDIO.aspx, accessed Mar. 2013