Proceedings of the 11th
KACC, October 1996

다수 모터 구동 실시간 제어 시스템

Real-Time Control System for Driving Multi-Motors

김재양*, 임영신**, 정신태***

* 숭실대학교 전자공학과 시스템제어 연구실 (TEL: 817-5987; FAX:821-7653; E-Mail: basagi@syscon.soongsil.ac.kr)
** 숭실대학교 전자공학과 시스템제어 연구실 (TEL: 817-5987; FAX:821-7653; E-Mail: lys@syscon.soongsil.ac.kr)
*** 숭실대학교 전자공학과 (TEL: 820-0638; FAX:814-1652; E-Mail: cst@syscon.soongsil.ac.kr)

Abstracts In this paper, we present real-time S/W architecture and techniques such as real-time scheduling, synchronization, and etc. for implementation of a real-time control system for driving multi-motors. Simulation shows feasibility of real-time S/W techniques presented here.

Keywords Real-Time system, Induction Motor, Multi-Tasking, Real-Time Scheduling, Vector Control

1. 서 론

서보모터는 FA및 컴퓨터기기 등에 많이 필요로 하기 때문에 그 사용환경에 맞는 고 정밀성이 요구된다[7]. 이 때문에 지금까지는 제어가 용이한 DC모터가 사용되어 왔으나, 유지 보수가 어렵고 대출력을 얻기 힘들기 때문에 최근에는 AC 서보모터의 고정밀 제어가 연구되어지고 있다. 그런데, AC 서보모터의 구동 제어시스템의 구현은 제어방식이 복잡하고 계산량이 많아서 보통의 범용 마이크로 프로세서는 그 구현이 쉽지 않다[2,3,4,6]. 더구나 fault detection, diagnostics, monitoring 기능등 부가적인 기능이 요구되는 경우는 더욱 쉽지 않게 된다. 하지만 최근에는 계산이 빠른 고속 마이크로프로세서인 DSP chip을 이용하여 필요로 하는 모든 기능을 만족하는 제어 알고리즘을 실시간으로 처리 가능하도록 구현하고 있다[6]. 그런데, 현재 고속의 DSP chip을 이용하여 설계하고 있는 모터구동용 제어시스템은 보통 서보기능만을 충실하게 구현하는 데에 치중하고 있으며 이 경우 대개의 DSP 시스템은 이를 충분히 수행하고도 CPU time의 여력이 생긴다. 때문에 대부분의 경우에는 monitoring 등의 다른 부가기능을 실현하는 알고리즘을 추가적으로 수행하도록 설계하지만, 이를 잘 이용하면 또 다른 모터를 구동하는 기능을 수행할 수 있다. 한 개의 구동제어시스템으로 다수의 모터를 구동하게 되면 여러 개의 모터가 사용되는 복잡한 공정에 적용하기가 쉬워지며, 이로 인한 경제적인 이득을 볼 수 있다[7].

다수의 모터를 구동하기 위한 방법으로는 여러 가지가 있을 수 있다[1,3,5]. 첫째 sequence한 program의 수행을 통해 다수의 모터의 제어를 위한 알고리즘을 동시에 수행하도록 하는 방법인데, 이 방법은 주기가 다른 여러 개의 모터를 구동하는 경우에 이를 처리할 수 있도록 하기 위해서는 program의 작성 시에 많은 노력이 필요하다. 또한 구동 시에 발생할 수 있는 많은 event를 처리하기가 매우 어렵게 된다. 두 번째 방법으로는 interrupt를 이용한 방법이 있는데, 이 방법은 timer 등의 주기적인 interrupt를 발생시켜서 이를 기준으로 하여 필요한 알고리즘을 수행하도록 하는 방법인데 이 방법은 구현이 간단하고 대부분의 경우 시스템이 안정적으로 운영되기 때문에 많이 사용되는 방법이다. 하지만 서로 다른 주기를 가진 알고리즘이 동시에 수행될 때에는 각 알고리즘의 주기를 유지하기가 어렵고, 많은 event가 동시에 발생할 때에는 시스템이 폭주할 수도 있다. 세 번째 방법으로는 시스템에 요구되는 여러기능을 태스크들로 나누고, 각 태스크들을 multi-tasking 할 수 있도록 하는 실시간 제어기법을 이용하여 이를 구현하는 방법이다. 이 방법은 구현이 어렵고 다른 두 가지 방법에 비해 CPU의 사용효율이 떨어진다는 단점이 있다. 하지만 실시간 기법을 이용한 경우에는 시스템이 수용할 수 있는 한도 내에서는 각 태스크의 주기와 무관하게 어떠한 경우에도 실행이 가능하며 이때 발생할 수 있는 어떠한 event라도 원활하게 처리할 수 있는 장점이 있다

그런데, 다수 모터를 구동하게 되는 경우는 취급해야할 태스크가 여러 개가 되고, 단순 서보기능 외에 다른 기능을 부가하는 경우에는 각각의 태스크 수행이 외부의 event에 맞게 실시간으로 처리되도록 할 수 있는 적절한 scheduler가 필요하게 되며, 또한 필요한 memory, I/O등의 자원도 효율적으로 잘 관리되어야 하는 등 Real-Time OS의 기법이 지원되는 실시간 제어기법에 관한 연구가 필요하게된다[1,5]. 따라서 본 논문에서는 다수 모터 구동을 위한 실시간 제어 시스템을 구현하기 위한 다수 모터 구동 실시간 제어 시스템의 S/W 구조 및 기법을 제시한다. 먼저 다수 모터 구동 실시간 제어 시스템의 개요에 대하여 2절에서 살펴보고, 그리고 이를 구현하기 위한 실시간 S/W의 구조와 구현방법을 3절에서 살펴본다. 4절에서는 시뮬레이션의 결과를 보이며, 마지막으로 5절에서는 결론이 주어진다.

2. 다수 모터 구동 실시간 제어 시스템의 개요

이 절에서는 본 논문에서 구현 대상으로 하는 다수 모터 구동 실시간 제어 시스템의 전반적인 구성과 고려된 AC 유도 모터 벡터 제어 기법등에 대하여 간략히 설명한다.

2.1 시스템의 전체 구성

본 논문에서 사용한 모터는 AC 유도 모터(AC induction motor)이고 제어기법은 '자속추정방식에 의한 속도센서 없는 벡터제어 기법'[4]이며, 실제 시스템의 구현은 2개의 CPU를 이용하여 구현되어졌다. AC 유도 모터를 벡터 제어기법을 이용하여 구동하려면 상당히 많은 양의 연산이 필요하며, 또한 이러한 시스템을 PC based system이 아닌 embeded system으로 구현하기 위해서는 user interface와 같은 부가적인 기능이 필요하다. 따라서 실제 H/W의 구현은 연산을 위주로 하는 제어알고리즘의 수행을 위해서는 DSP CPU로 분류되어지는 TMS320C31을 사용하였고, user interface, alarm의 처리와 같은 부가적인 기능을 처리하기 위하여 i80C196KC를 이용하였다. 구현되어진 시스템의

H/W를 운용하기 위한 S/W역시 두 개의 부분으로 나누어지는데 user interface를 담당하는 부분의 Kernel에는 일반적인 multi-tasking의 기법을 이용할 수 있으나, 연산을 담당하는 부분의 Kernel은 일반적인 multi-tasking 기법과는 다른 실시간 처리 기법이 필요하다. 따라서 여기서는 다수 모터 구동을 위한 시스템에서 필요한 실시간 S/W의 구조에 중점을 두어 설명한다. 다음의 그림 1은 본 논문의 연구에서 설계되어진 다수 모터 구동 시스템의 전체 구조이다.

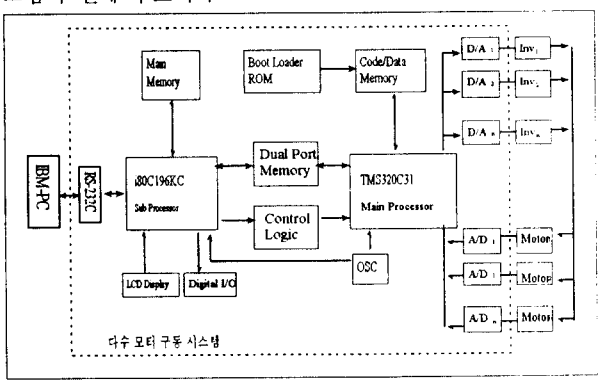

그림 1. 전체 시스템의 구조

2.2 고려된 AC 유도모터 벡터제어 기법

본 논문에서 고려한 AC 유도 모터 벡터제어 기법은 고가의 속도센서 대신 속도추정을 통해 속도센서가 없는 벡터제어기법이다. 그림 2는 본 논문에서 고려된 속도센서없는 벡터제어기법의 AC 유도모터 속도제어시스템을 나타낸다.

그림 2의 자세한 내용은 참고문헌 [2,4]를 참조하기 바란다.

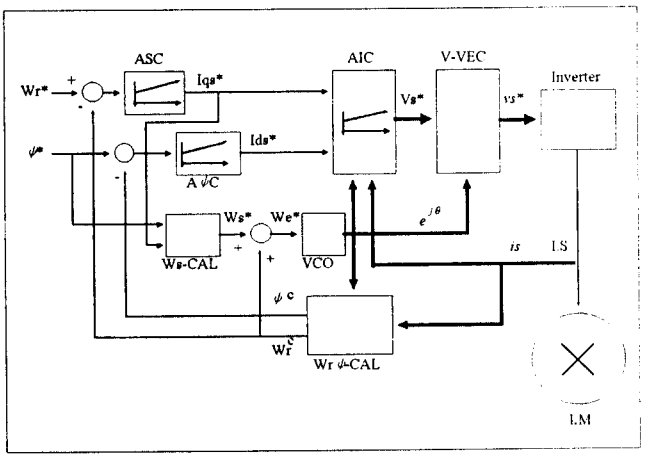

그림 2. 자속 추정방식에 의한 속도센서없는 유도전동기의 벡터제어

*, c : setting value and calculated value
ϕ : rotor flux
V-VEC : voltage vector generator
Wr : rotor angular speed
Ws : slip angular frequency
ASC : speed cotroller
AϕC : flux controller
AIC : current controller
VCO : voltage controll oscilator
Ws-CAL : slip angular speed calculation
We : angular speed of refernce frame
Ids, Iqs : d-q component of stator current in the rotating frame
Vs : vector of stator voltage in the rotating frame
vs : the phase voltage
is : the phase current
Wrϕ-CAL : speed and flux calculation

3. 다수 모터 구동을 위한 실시간 S/W의 구조

3.1 실시간 S/W의 구조

다수 모터 구동을 위한 실시간 제어 시스템의 구현을 위해 가장 핵심이 되는 Kernel은 다음과 같은 조건을 만족하여야 한다.

첫째, 대부분의 모터 구동용 제어 시스템은 정확한 제어가 이루어지지 않을 경우 주변환경에 치명적인 영향을 끼칠 수 있는 Hard-Real Time 시스템이다. 따라서 이를 운영하기 위한 운영체제는 각각의 제어 알고리즘을 수행할 때 제한된 deadline내에 결과가 나올 수 있도록 scheduling을 하여야 한다.

둘째, 하나의 모터를 구동하기 위한 알고리즘을 여러 개의 module로 나누어 작성할 경우, 이를 다수의 모터를 구동하기 위한 시스템으로 확장할 경우 기존의 알고리즘을 그대로 이용하고 필요한 data만 바꾸어 사용하면 다수의 모터를 구동할 수 있으므로, 매우 유용하게된다. 하지만 이 경우에는 각 알고리즘의 module간에는 시간적인 선행(先行)조건이 있을 수 있으므로 이를 만족시켜주어야 한다.

셋째, 모터 구동 시에 발생하는 많은 alarm, event들을 제어 알고리즘의 연산에 영향을 주지 않고 수행하려면 event의 효율적인 처리가 필요하다. 이를 일반적인 interrupt service routine에서 처리할 경우 수행중인 태스크에 영향을 미칠 수 있으므로 별도의 interrupt handler를 두어 발생하는 event의 종류에 따라 적절한 처리를 해주게 된다.

본 논문에서는 semaphore와 message mail box를 적절히 이용하여 이들 각각의 태스크들을 scheduling 한다. 다음의 그림 3은 다수 모터 구동 실시간 S/W의 전체구조를 나타내는데 자세한 사항은 3.3절과 3.4절에서 설명한다.

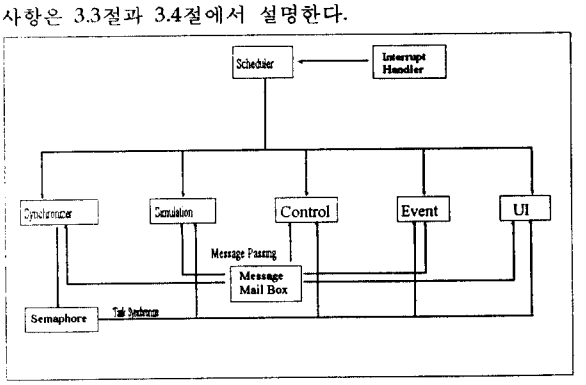

그림 3. 실시간 S/W의 구조

3.2 전체시스템의 태스크 구성과 수행 순서

다수의 AC 유도 모터를 벡터 제어하기 위해서는 많은 태스크가 서로 유기적으로 결합하여 실행되어진다. 이를 크게 나누면 제어 알고리즘을 수행하는 태스크들과 모터 시뮬레이션을 처리하는 태스크(시뮬레이션의 경우), 그리고 event들을 처리하는 태스크로 나눌 수 있다. 먼저 제어 시스템에서 수행되는 태스크를 분류해보면 크게 3가지로 나눌 수 있다(그림 1과 그림 2 참조).

첫 번째로 분류되어지는 태스크들은 주기적인 태스크들로써 주로 제어 알고리즘의 연산에 필요한 control 태스크들인데 Voltage Vector Generation을 위한 태스크, Current Control을 위한 태스크, State Estimation을 위한 태스크가 있다. 특히 시뮬레이션의 경우에는 인버터와 모터 동역학 시뮬레이션 태스크가 각각 필요하다. 이 경우 다른 어떤 태스크보다 우선 순위가 높이야 한다. 또한 이러한 태스크들은 제 주기마다 재대로 수행이 되지 않는 경우 제어 시스템 성능에 크게 영향을 끼치는 time-critical 태스크들이다. 이러한 태스크들의 scheduling은

hard real-time scheduling 기법이 필요하다.

두 번째로는 비주기적인 태스크들인데 모두가 외부의 event에 의해 수행되는 태스크들로 alarm 처리를 위한 태스크와 user interface를 위한 태스크들인데, 이들 태스크는 모두 비동기 태스크들이다. 그중 alarm 태스크는 다른 태스크들보다 우선적으로 처리하여야 하는 태스크이지만, 다른 event들은 경우에 따라 control 태스크의 수행 후에 처리하여야 하는 태스크들이다. 따라서 이들 태스크들의 scheduling은 수행되어야 하는 태스크에 따라 다른 우선 순위를 가지도록 하고 scheduler는 그 우선 순위에 따라 적절한 태스크를 수행하도록 하는 priority preemptive scheduling을 하여야 한다.

세 번째로는 back ground 태스크인데 시스템의 상황을 알려주기 위한 Graphic Display 태스크와 시스템의 진단을 위한 Diagnostic Task가 있다. 이러한 태스크들은 특별한 시간의 제약성을 가지지 않으므로 back ground 태스크라고 불리는데 흔히 시분할 기법을 이용하여 scheduling하게 된다.

위에서 설명한 다양한 종류의 태스크들은 semaphore와 mail box를 이용하여 동기화 된다. 그림 4는 전체 태스크의 구성도인데, 실제 구동에서는 가장 먼저 모터의 상태를 읽어오지만 시뮬레이션에서는 실제 모터의 역할을 수행하는 모터 및 인버터 시뮬레이션 태스크가 먼저 수행된다. 그 결과를 이용하여 Estimator, Controller, Voltage Vector Generator가 각각 수행하게 된다. 이때 태스크간의 data의 교환과 선행조건을 유지하기 위한 동기화는 mail box를 이용하는데 자세한 내용은 3.3절과 3.4절에서 설명되어질 것이다.

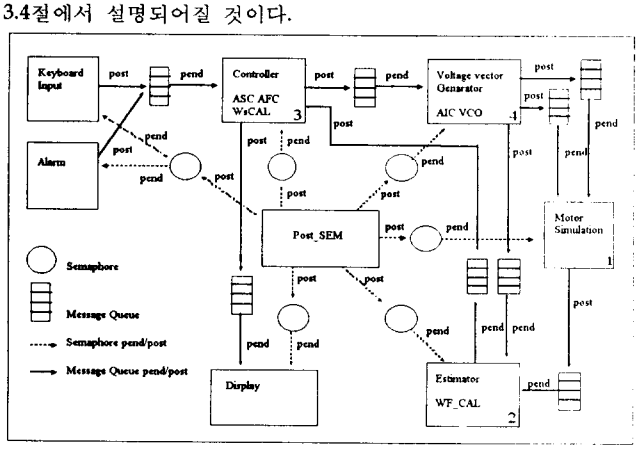

그림 4. 시스템의 Task 구성도

3.3 *Scheduling*

다수의 모터를 구동하는 시스템에서는 위에서 설명한 것처럼 Control 태스크들로 이루어진 주기적인 태스크와 Display 태스크 같은 back ground 태스크, 그리고 event 태스크나 alarm 태스크처럼 interrupt에 의하여 구동되어지는 event 태스크의 세 가지가 있는데, 이들은 각 태스크의 특징에 따라 scheduling 방법이 달라야 한다. 특히 Control 태스크와 같은 태스크들은 Hard Real-Time의 특성을 가지는 태스크들이므로 Fault 발생을 감지하고 처리할 수 있어야 한다. 하지만, 이러한 특징을 모두 처리하는 scheduling 방법을 지원하는 scheduler는 CPU의 수행시간을 너무 많이 차지하는 비효율적인 scheduler가 되기 쉽고, 심지어는 구현되어진 scheduler가 시간 제약조건을 만족하지 못하거나 하는 등의 이유로, 실시간 시스템에는 적합하지 않을 수도 있다. 따라서 본 논문에서는 Kernel이 수행하는 scheduler는 단순히 실행할 준비가 된 태스크들 중에서 가장 높은 우선 순위의 태스크를 실행하는 priority based preemptive scheduler의 역할만을 가지도록 하여 매우 짧은 수행시간을 가지도록 설계하며, Control 태스크와 같이 hard real-time의 특성을 가지는 태스크에 대해서는 다음과 같은 방법을 이용하여 각각의 태스크에 대한 scheduling을 수행한다.

먼저, 제어 알고리즘을 연산하는 주기적인 태스크들은 필요한 sampling time에 따라 수행되도록 하되, 각 태스크의 선행조건을 만족시킬 수 있도록 하는 부가적인 태스크를 두어 처리한다. 즉, 원래의 Kernel의 scheduler는 priority를 이용한 scheduling을 하도록 하고, 가장 높은 priority를 가지는 Synchronizer 태스크를 만들고 Kernel의 scheduler가 scheduling할때 우선적으로 수행되도록 한다. 이 Synchronizer task는 control 태스크들만을 scheduling하게 되는데, 각 control 태스크들의 주기와 선행조건에 따라 수행되어야 할 태스크를 선택하고 필요한 시간내에 수행될 수 있도록 semaphore를 이용하여 태스크에게 알려준다. 하지만 다른 태스크들에 대해서는 이러한 방법의 scheduling이 필요하지 않으므로 모든 제어 알고리즘이 필요한 주기 내에 한 번씩 수행되지 않았을 경우에만 수행되도록 하였다. 또한 이 태스크는 semaphore의 상태를 관찰함으로써 현재 수행중인 태스크, 이미 실행을 마친 태스크, 또는 아직 실행되지 못한 태스크들의 구분을 하고, 이들 각각의 태스크들의 선행조건을 계산하여 각각의 태스크를 동기화 시키는 역할도 수행하는데, 이때 각각의 태스크에서 Fault가 발생 하였는지의 여부를 판단하고 이를 alarm 태스크에 전달하는 일도 수행한다.

그 다음으로는, back ground 태스크들인데 이들 태스크는 주로 System의 상황을 사용자에게 알려주기 위한 태스크로 특별한 시간 제약조건을 가지지 않는다. 따라서 이러한 태스크들은 CPU가 다른 태스크를 모두 수행하고 남는 시간 동안 수행되어야 한다. 이것은 이들 태스크의 priority를 가장 낮은 priority를 가지게 함으로써 별도의 부가적인 scheduler없이 해결한다. 즉 scheduler는 우선 순위에 따라 scheduling을 하므로, 이들 태스크보다 우선 순위가 높은 태스크가 있을 때에는 수행되지 않도록 한다.

마지막으로는 event 태스크들인데 이들 태스크는 대부분 interrupt에 의해 구동되는 태스크이다. 이러한 태스크는 interrupt routine에서 모두 처리될 경우 다른 태스크의 수행에 영향을 줄 수가 있으므로 이들 태스크는 각각의 event를 처리하는 interrupt handler에서 필요한 처리를 하는 태스크를 생성시키는 방법을 이용한다. 이때 일반적인 event는 제어 알고리즘보다 낮은 priority를 가지도록 하고, 이상발생을 처리하는 alarm event는 제어 알고리즘보다 높은 priority를 가지도록 함으로써 제어 알고리즘이 Sampling Time을 놓치는 경우가 없도록 설계하였다.

3.4 *태스크간의 통신 (IPC)*

다수 모터 제어 시스템의 경우에는 여러 개의 module로 나누어진 제어 알고리즘들이 각각의 module에 전달하여야 할 parameter들이 매우 많다. 이들 parameter들은 global 변수를 이용하여 처리할 수도 있지만 global변수를 이용하여 이를 구현하려면 여러 개의 모터가 가지는 parameter들이 서로 다른 이름을 가져야 하며, 각각의 알고리즘들은 경우에 따라 다른 변수를 참조할 수 있도록 구성되어져야 하므로 많은 memory를 차지하며 program도 복잡해진다. 따라서 각 알고리즘간의 parameter 전달에는 message mail box를 이용하는 것이 간단해진다. 하지만 mail box를 사용하는 경우에는 단 하나의 parameter만을 전달할 수 있으므로 각 알고리즘사이에 전달되는 많은 parameter들은 모두 structure를 이용하여 하나로 묶을 필요가 있다.

AC모터를 여러개의 알고리즘으로 제어할 경우에는 연산을 수행하는 태스크들의 sampling time이 서로 다른 경우가 있다. 이때 주기가 짧은 태스크가 주기가 긴 태스크의 결과를 계속 기다리게 된다면 필요한 deadline을 놓치게 된다. 이 때문에 주기가

짧은 태스크는 message box를 사용할 때 필요한 data가 도착할 때까지 기다리지 않고 일정한 시간을 두어 이때까지 data가 도착하지 않으면 이전의 data값을 이용하여 연산을 수행하게된다.

4. 시뮬레이션

다음의 결과들은 본 논문에서 설명되어진 실시간 S/W의 기법을 적용하여 구현 되어진 시뮬레이션 program을 IBM-PC에서 수행시킨 후 각각의 태스크의 수행시간을 계산하고, 이를 이용하여 실제 다수 모터 구동 시스템에서 각각의 태스크가 어떻게 scheduling되어 실행되는지를 보이기 위한 것이다. 실제 구동시와는 달리, 시뮬레이션을 하는 경우에는 모터 시뮬레이션 태스크가 필요한데, 시뮬레이션의 경우에는 매우 많은 시간이 소요된다. 또한 IBM-PC의 경우 CPU의 연산시간은 TMS320C31의 연산시간과 비슷하지만, I/O 및 운영체제와 같은 부가적인 영향으로 time tick을 1ms이하로는 하기가 힘들다. 따라서 이를 시뮬레이션하기 위해서는 기준이 되는 시간을 늘려야 한다. 다음의 표 1은 시뮬레이션에서 사용되어진 시간을 나타내는데, 시뮬레이션을 할 때에는 모터의 시뮬레이션을 위한 태스크가 필요한데, 표에서 시뮬레이션 태스크의 수행 시간은 Inverter의 시뮬레이션 (0.2ms) 시간에 모터의 시뮬레이션시간 (1.4ms)을 더한 것이다. 이 때문에 시뮬레이션시에는 실제 구동할 때와 비교하여 CPU의 사용율이 같도록 했는데 실제 구동시에는 40.5%이며, 시뮬레이션시에는 44.5%인에 약간의 시간차이는 scheduler등의 영향으로 정확한 시간을 측정하지 못한 이유인 것으로 보인다.

Task / 종류	Simulation	V-VEG	Control	Estimator
실제 수행시간	2μs	17μs	8μs	16μs
Simulation의 주기	4.6ms	4.6ms	9.2ms	27.6ms
Simulation의 수행시간	1.6ms	0.3ms	0.2ms	0.3ms

표 1. Simulation의 결과 Data

다음의 그림 5는 실제 scheduler의 scheduling을 시뮬레이션하여 그 결과 data를 이용하여 그림으로 나타낸 것인데, 그림에서 가로로 그려진 시간축의 윗 부분은 모터 1의 수행을 나타내며, 아랫부분은 모터 2의 수행을 나타낸다. 그림 5에서 화살표로 표시되어진 부분은 alarm의 발생으로 인해 발생된 태스크이다.

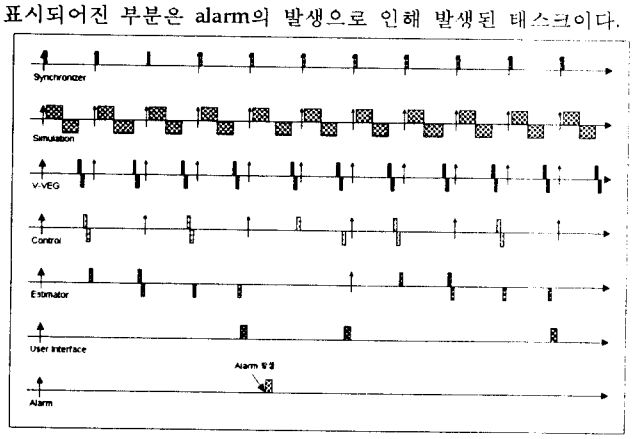

그림 6. Scheduling의 Simulation

다음의 그림7은 시뮬레이션 program을 이용하여 모터가 지령치에 도달하는 것을 그린 그림인데, 두 개의 모터가 정확하게 입력된 지령치에 수렴하는 것을 볼 수 있다.

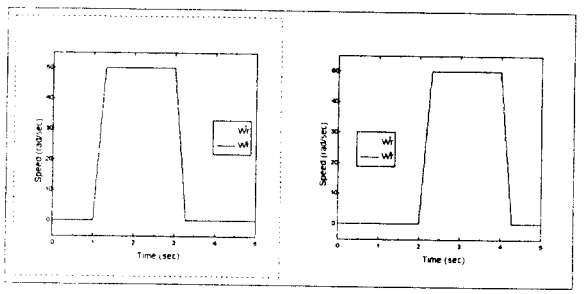

그림 7. 모터 시뮬레이션 결과

5. 결론

본 논문에서 구현되어진 다수 모터 구동 실시간 제어 시스템과 같은 시스템의 경우에는 주어진 제한시간 이내에 필요한 제어 알고리즘의 수행 및 임의로 발생하는 모든 event 및 alarm을 처리할 수 있는 scheduler가 필요하며, 자원의 공유, 태스크 간의 통신 등을 지원할 수 있는 실시간 S/W의 기법을 사용한 Kernel의 구현과 이를 지원할 수 있는 H/W 가 필요하다. 따라서 본 논문에서는 다수 모터 구동 실시간 제어 시스템을 위한 H/W를 고속의 연산기능을 가진 TMS320C31과 다양한 I/O를 처리할 수 있는 i80C196KC를 이용하여 구현하고, 이의 운용을 위하여 필요한 Kernel을 구현하는 방법을 설명하였다.

다수 모터 구동 지원 실시간 S/W 을 구현하기 위해서는 먼저 AC 모터 벡터 제어알고리즘을 분석하여 이를 여러개의 태스크로 분류하고 이를 시뮬레이션을 통하여 확인하는 과정이 필요하다. 또한 다수모터를 구동하는데 부수적으로 필요한 monitoring, alarming 등의 태스크를 제어 알고리즘의 연산에 필요한 태스크들과 함께 scheduling할 수 있는 실시간 scheduler 및 태스크의 동기화, data의 교환에 필요한 semaphore, mail box 등의 필요한 Kernel의 기능을 실시간 S/W기법을 이용하여 구현하였고, 이를 IBM-PC에서 시뮬레이션을 통하여 확인하였다.

참고문헌

[1] Auslander D.M. and C.H. Tham, "*Real-Time Software for Control : Program Examples in C*", Prentice-Hall, Englewood Cliffs (NJ), 1990

[2] B.K. Bose, "*Power Electronics and AC Drives*", Prentice-Hall, Englewood Cliffs (NJ), 1986

[3] B.K. Bose and P.M. Szczesny, "A Microcomputer-Based Control and Simulaton of an Advanced IPM synchronous Machine Drive System for Electric Vehicle Propulsion", *IEEE Trans, Industrial Electronics, Vol. 35, No. 4*, Nov. 1988

[4] T. Ohtani and N. Takada and K. Tanaka, "Vector Control of Induction Motor Without Shaft Encoder", *IEEE Trans. on Ind. Appl., vol. IA-28, No.1, pp 157-164*, 1992.

[5] A. Van Tilborg and G.M. Koob (Eds.), "*Foundations of Real-Time Computing. Scheduling and Resource Management*", Kluwer Academic Publishers, Boston, 1991

[6] 이상욱, 하인중, 고명삼, "고속 Micro-processor 를 이용한 Digital AC Servo Motor Controller 개발", 서울대 자동차 연구소, 1991.

[7] 하인중, "FA용 AC 서보모터의 고성능 제어에 관한 최근 기술경향", 전자공학회지, vol.20, No.3, pp.22-29, 1993.