단일개발환경을 위한 제어용 실시간 운영체제의 개발

° 박세진
쌍용자동차 (Tel 0333-610-2343)

Development Of Controller Area Operating System For Uniform Developing Environment

Sejin Park
Ssangyong Motor Company

Abstract

The concept of unifomity in control implementation is exploled for improving efficiency of design procedure. A controller area operating system which includes real time kernal and control specific shell are developed. Three examples are discussed for the validationn of the system.

1. 서론

디지탈 제어기는 아날로그 제어기에 대한 여러가지 장점으로 산업계 전반에 걸쳐 사용되고 있다. 이러한 괄목한 성장은 의심할 여지 없이 마이크로프로세서 등의 하드웨어의 성능 향상에 기인한 것으로 오늘날의 대부분의 제어기에는 마이크로프로세서가 주요 부품으로 자리잡고 있다.

반면에 제어용 소프트웨어의 설계와 개발 기술은 발전된 하드웨어를 쫓아가지 못하고 있다. 이런 소프트웨어의 개발 문제들 중에는 잘알려진 것들도 있는데, 실시간 시스템의 설계, 제어기 알고리즘의 구현 및 편리한 시험환경의 구축과 같은 것이 그것이다.

첫째, 실시간 시스템의 설계상의 어려움을 개발자들은 제어기 프로그램의 단순화로 해결하고 있다. 예를 들어 한개 정도의 일정한 주기의 샘플링 시간으로 인터럽트 처리를 하며, 다른 기능을 수행할 필요가 있을 때는 먼저 수행하고 있는 프로그램을 정지하고 수행한다. 그러나, 복잡한 기능의 요구가 실시간 문제와 연결된 다면 새로운 방안이 필요하다. 즉 실시간 운영체제와 같은 커널을 이용하는 것이다[1]. 둘째, 제어기 알고리즘의 구현 및 편리한 시험환경의 구축을 위해서는 효율적이고 유연한 인터페이스 도구가 요구된다. 중요한 점은 이러한 도구가 모든 제어 분야에 적용될 수 있는 범용의 것이라야 한다. 기존에는 제어기 개발자가 새로운 프로젝트가 시작할 때마다, 새로운 개발환경과 도구에 익숙해져야하고 새로운 소프트웨어를 처음부터 작성해야 한다. 반복적인 작업은 개발시간을 길게하고, 비용의 상승과 신뢰도의 저하를 초래한다.

본 논문에서는 상기한 문제점을 해결할 수 있는 실시간 커널과, 제어용 쉘을 이용한 제어용 운영체제의 개발을 설명하며, 세가지 적용예를 통해 이의 타당성을 검증한다. 개발된 제어용 운영체제의 명칭은 NANOOS로 한다.

NANOOS 운영 시스템은 커널(KERNEL), 시스템콜 라이브러리, 쉘프로그램 세부분으로 되어 있다. NANOOS 운영체제의 기본 구성을 그림 1에 나타내었다. NANOOS를 사용하면 복잡한 환경을 유지해야하는 단점이 있지만, 다음과

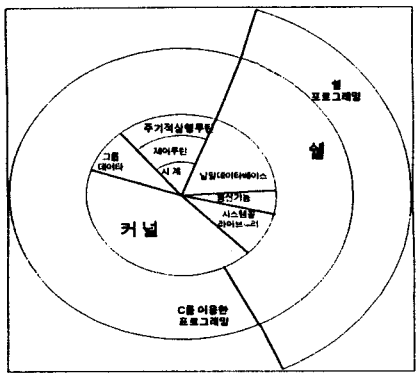

그림 1. NANOOS의 기본 구성

같은 당일환경의 장점을 기대할 수 있어 개발자의 경쟁력을 현저히 향상시킬 수 있다.

- 빠른 개발 시간 - 최단 시간에 상품화
- 편리하며 제어기 개발에 공통적으로 사용할 수 있는 프로그램 방법 제공
- 새로운 프로젝트를 시작할때 매번 새로운 환경을 구축하거나 배울 필요가 없음
- 개발환경에 대한 신뢰도 향상

2. NANOOS 커널의 개발

NANOOS는 한가지 이상의 프로그램(타스크)을 동시에 처리할 수 있도록 고안되었다. 이는 멀티태스킹커널에 의해 가능해진다. NANOOS커널은 제어용 목적으로 개발되었다. 이러한 제한 조건으로 화일관리기능, 복잡한 메모리 관리 기능 등은 초기 설계에서 제외되었고, 커널 내에 정보 교환 장소인 날말데이타베이스와 스케쥴링시 때마다 주기적으로 실행되는 루틴을 두었다. NANOOS커널은 다음과 같은 조건을 만족하도록 설계하였다.

◉ 빠른 콘텍스트스위칭 시간을 갖는다.

일반 프로세서를 채택하고도 MOTION CONTROL 분야에 적용 가능하도록 설계하였다.

◉ 주기적 실행 루틴

빠른 샘플링 타임이 요구되는 응용에 적합하며, 상용 라이브러리가 준비되었다.

◉ 날말데이타베이스

운영체제 내에 존재하는 주요 파라메타, 프로그램들을 데이타베이스로 관리한다. 또한, 커널과 쉘의 통신 창구로써 활용된다.

◉ 프로그램 전체 사이즈가 작다.

제어용의 전용 운영체제를 지향하며 최종 시스템에 필요한 기능만을 골라서 탑재할 수 있다.

◉ 외부 인터럽트에 빠르게 응답한다.

인터럽트가 금지되는 시간을 최소화 하는등의 고려를 하였다.

◉ 우선순위를 갖는 스케쥴링 메커니즘을 제공한다.

사용자가 타스크 우선순위를 정할 수 있고, 높은 우선순위의 타스크가 발생하면, 현재 커널중에 있는 낮은 우선순위의 타스크는 즉시 PREEMPTION 된다. 각 타스크의 우선순위는 고정되있다(STATIC PRIORITY SYSTEM).

◉ 타스크가 수행을 일시 멈출 수 있는 기능 제공

타스크가 일정 시간 동안 수행될 필요가 없을 때, SUSPEND 상태로 있게해 다른 타스크가 CPU TIME을 갖게 한다. 또한, 딜레이등 시에 타임아웃 기능을 제공한다.

◉ 그룹데이터 관리 기능

많은 양의 메모리 관리를 위해 그룹데이터 관리 기능을 제공하고 메모리내의 데이터나 코드를 처리, 압축, 코드화할 수 있는 기능을 제공한다. 또한, 빠른 속도로 데이터를 축적할 수 있는 방법을 제공한다.

◉ 타스크간의 통신과 동기화를 제공한다.

이벤트(EVENT), 세마포어(SEMAPHORE), 메시지 전송과 같은 프리미티브를 제공한다.

NANOOS커널의 콘텍스트스위칭과 타스크의 상태 설명을 통해 커널의 주요한 동작방식에 관해 설명하고 타스크간의 정보교환 및 동기화 방법을 통해 타스크들의 공동작업방식과 NANOOS커널의 특징인 주기적 실행루틴과 날말데이타베이스에 관해 설명한다.

2.1. 콘텍스트스위칭

NANOOS는 타스크 스케쥴링을 시분할(TIME SHARING)기법으로 각각의 타스크에 일정한 CPU 시간을 할당하고 타스크를 바꾸는 방법을 사용한다. 이때 하나의 타스크에서 CPU의 제어권이 다른 타스크로 넘어가는 것을 콘텍스트스위칭이라 하며, 콘텍스트라는 것은 각 타스크에 대한 몇 가지의 정보를 갖는 테이블을 말한다. 이 방법을 사용하면 각 타스크는 고정된 CPU TIME을 할당받게 된다(CLOCK의 규정된 TICK수만큼). 그리고 할당받은 CPU TIME의 끝에 동작을 멈추게 되고 타스크리스트(TASK LIST)에 있는 다른 타스크를 수행하게 된다. 그리하여 교체된 타스크가 같은 CPU TIME을 공유하여 수행되는 방식이다. 만일 한 타스크가 할당된 CPU TIME이내에 끝난다면 리스트내의 다른 타스크가 즉시 수행된다.

NANOOS는 우선순위 스케쥴링 메카니즘을 사용한다. 각 타스크는 우선순위에 따라 할당되고 미리 지정된 시간의 끝부분에서 실행될 준비가 되어 있는 가장 높은 우선 순위의 타스크가 선택되고 CPU TIME을 가지게 된다.

2.2. 타스크의 수행 상태

NANOOS커널은 각각의 타스크(프로그램)가 수행되는 상태의 명확성과 타스크 제어를 가능하게 하기위해 다섯가지의 타스크 상태를 정의한다.

◉타스크 상태(TASK STATES)

① ACTIVE : 이 상태에 있는 타스크는 CPU의 제어권을 가지며 수행된다. 일반적으로 이 타스크는 실행할 준비가 되어 있는 타스크 중 가장 높은 우선 순위의 타스크이다.

② READY : 이 상태에 있는 타스크는 모든 필요한 자원을 이용할 수 있으나 현재 수행되지 않는 상태이다.

③ SUSPEND : 이 상태에 놓여져 있는 타스크의 수행은, 타스크가 현재 이용할 수 없는 자원을 요구하거나 혹은 타스크가 플랜트로부터 일부 SIGNAL을 기다리고 있기 때문에, 혹은 타스크가 경과 시간을 기다리고 있기 때문이다.

④ EXIST : 운영체제는 이 상태의 타스크의 존재를 인식하고 있으나 이 타스크는 우선순위가 부여되지 않았고 수행할 준비를 갖추지 않았다.

⑤ NON-EXIST: 비록 컴퓨터의 메모리 내에 상주되어 있더라도 운영체제는 아직이 타스크의 존재를 인식하지 않은 상태이다.

2.3. 타스크간의 정보교환 및 동기화

멀티 타스크 시스템에서 타스크간에 메모리 또는 각종 자원을 공유해야 할 때가 있다. 이때 발생하는 문제는 어떤 타스크가 공유 메모리를 억세스 하고 있는 도중에 보다 우선순위가 높은 타스크가 CPU의 제어권을 빼앗아 그 메모리를 갱신하고자 하는 경우 발생한다, 그래서 메모리를 억세스하는 프로그램이 완전히 종료할 때까지 다른 타스크가 접근하지 못하게 해야 하는데 이러한 프로그램 코드를 임계 영역(CRITICAL REGION)이라 한다. 임계 영역을 보호하기 위한 방법으로 세마포어를 사용할 수 있다. 앞의 두 타스크간의 임계영역을 세마포어와 그 카운터 값의 조정으로 해결할 수 있다. NANOOS는 세마포어, 이벤트, 메시지 전달 등의 통신 기능을 보유하고 있다.

2.4. 날말데이타베이스

운영체제 내에 존재하는 주요 파라메타, 프로그램들을 관리하는 데이타베이스이다. 또한, 커널과 쉘의 통신 창구로써 활용된다.

2.5. 주기적 실행모듈

제어기 구현, 디지탈 신호처리와 소프트웨어 타이머 등에 응용할 수 있는 일정주기 샘플링 마다 실행되는 루틴이다. 사용자 정의 루틴을 넣을 수도 있고, 각종 상용의 라이브러리를 구비하고 있다. 라이브러리에 포함된 모듈은 PI, PD, LMFC, FIR 필터, 임의 함수 발생기, 데이타 획득, 적분기 등이다. 또한, 각각의 모듈은 파라메타, 리미터 등을 날말데이타베이스를 통해 선택할 수 있게 한다. 예를 들어 식 (1), (2)와같은 형태의 PI, PD 제어기를 구현한다. 이를 이산 시간 제어기로 나타내면 각각 식 (3), (4)와 같다. 제어기 구현시에는 1 STEP 지연을 갖게 한다.

$$G_{PI}(S) = K_P \left(1 + \frac{1}{\tau_I S} \right) \quad (1)$$

$$G_{PD}(S) = K_P(1 + \tau_D S) \quad (2)$$

$$G_{PI}(Z) = \frac{K_P}{2\tau_I} \frac{(2\tau_I+T)Z + (T-2\tau_I)}{Z - 1} \quad (3)$$

$$G_{PD}(Z) = K_P\left(1 + \tau_D \frac{Z - 1}{TZ}\right) \quad (4)$$

PI, PD 제어기를 이용하여 성능 사양을 만족하지 못할 경우 논문[2]에서 제안된 방법을 이용하여 마찰력 보상등을 병행하여 사용할 수 있다. 이 방법은 속도의 부호와 제어기 출력의 부호에 따라 조정된 값으로 FRICTION을 보상(OVER COMPENSATION, UNDER COMPENSATION)한다.

3. NANOOS쉘의 개발

NANOOS 운영 프로그램은 새로운 명령어를 만드는데 사용될 수 있는 자체의 사용자 인터페이스인 NANOOS쉘 이 담당 한다. NANOOS쉘은 제어용 사용자 인터페이스 기능이 강조된 범용의 쉘이다. 일반적인 쉘의 기능 뿐 아니라 디버거 기능등과 프로그램이 가능한 인터프리터 언어로써의 기능이 통합된 것이다. 여러가지 기능의 유기적인 통합으로 내장형 시스템의 특성을 최대화시킬 수 있으며, 단일한 개발체제로써 모든 제어분야에 적용될 수 있다. NANOOS쉘은 언어이면서 동시에 프로그램 환경이다. 이에 포함된 기능은 다음과 같다.

- 디버거(DEBUGGER)
- 로더를 포함한 모니터 프로그램
- 데이타 IDENTIFICATION
- 신호 처리 (행렬연산포함)
- 제어기 구현
- 파라메타 튜닝
- 제어 성능 평가
- 최적화
- 일괄처리(BATCH PROCESSING)
- 인터프리터 형식의 프로그래밍 기능

NANOOS쉘은 비 프로시저 형태의 언어이며 객체지향 특성을 갖고있는 스택 연산에 기반을 둔 언어이다. 함수 또는 모듈간의 매개변수, 결과 등의 교환과 지역변수등 저장장소는 스택을 이용하면 각각의 모듈이 독립적으로 정의될 수 있고, RECURSVE 와 RE-ENTRANCE 가 보장된다. 각각의 모듈이 독립적인 기능을 수행하므로 그 자체로 시험이 가능하고 에러를 수정할 수 있어 테스트, 디버깅, 유지 보수가 쉬워진다. 단순한 것들을 피라미드 식으로 계속 쌓아 나감으로써 복잡한 일도 처리할 수 있는 처리계이다. 폭대기에서 내려오는 TOP-DOWN 프로그램법과 바닥에서 올라가는 (BOTTOM-UP) 프로그램법의 장점을 결합할 수 있다. 작성된 프로그램은 모듈화 되있기 때문에 유지 보수가 쉽다. NANOOS쉘는 후위 표기법을 지원하므로 목적어 다음에 동사가 우리말과 가깝다. 예를 들면 "3과 4를 더해서 보여라" 는 NANOOS쉘로는 "3 4 + ?" 로 표현한다. 또한 자료로 얻은 것을 마치 코드인 것처럼 실행시킬 수 있는 LISP에서 구현되는 기능을 흉내낼 수 있다. 실행할 때 사용자로부터 문자열을 입력받는 프로그램을 짰다고 하자. NANOOS쉘은 이 문자열을 NANOOS쉘 코드로 간주하고 번역, 실행시킨 다음 원래의 프로그램으로 돌아올 수 있다.

4. 내구 시험 장비에의 응용

특정 기기 또는 부품의 내구시험을 위해 반복 시험하고 결과를 보고하는 내구 시험 시스템에 NANOOS를 적용한다. 그림 2에 제작된 기기의 실물을 보인다. NANOOS는 현재 CPU V25 와 V55 에 포팅되어 있으며 68340에 포팅 진행 중이다. 제작된 내구 시험기는 V55 를 탑재하였으며, 이의 시험 사양은 그림 3을 참조한다. 그림 3에서 ON, OFF 시간, 1회의 반복 시간, 1회 완료후 휴지기간 등은 날말데이타베이스를 이용하여 지정할 수 있다. 또한, 내구 반복횟수를 먼저 설정하고 설정치 만큼 작동시킬 수 있다.

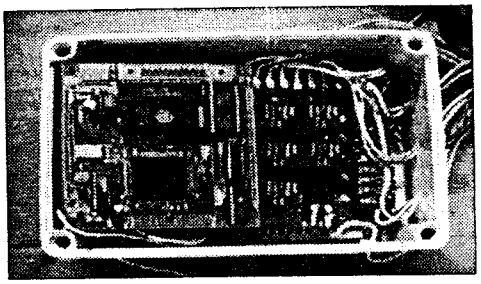

그림 2. 제작된 내구테스터

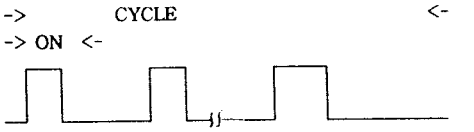

그림 3. 내구 시험 사양

4.1. NANOOS커널을 이용한 프로그래밍

그림 3의 사양과 같은 내구시험기 다섯개를 한개의 CPU로 NANOOS커널 만을 이용하여 동작 시키는 프로그램을 작성하였다. 한개의 타스크는 NANOOS쉘을 실행하여 사용자와 수시로 인터페이스 한다. 임의의 시간에 그림 3의 모든 파라메타를 수정할 수 있으며 수시로 시험진행 상태를 확인할 수 있다. 시간지연 루틴에서는 타스크를 일정시간 SUSPEND 상태로 만들므로 CPU 자원은 효율적으로 관리된다.

4.2. NANOOS쉘을 이용한 프로그래밍

그림 3의 사양을 NANOOS쉘을 이용하여 프로그램한 것을 표 1에 보인다. 이와 같이 NANOOS쉘은 적은 코드로 제어용 프로그램을 작성할 수 있다.

```
"o1 (dup "p1 s2macro 1 dout) ;
"clean (heat shake normal inhibit wait) ;
"heat (p1 &24 or o1) ;
"normal (p1 0 and o1) ;
"rain (p1 &9 or o1 inject_time wait p1 &f6 and o1) ;
"do_cycle (rain clean) ;
"run1 (init cycle "do_cycle repeat) ;
"shake (num "wave repeat) ;
"wave (do_1 do_2 do_3 counting) ;
"do_1 (p1 2 or o1 on wait) ;
"do_2 (p1 &10 or &fd and o1 on wait) ;
"do_3 (p1 &2d and o1 off on - wait) ;
"init ("cnt "0 ; "p1 "0 ; ) ;
"counting ("cnt inc cnt Lout ( COUNTER : ) cnt ftoa + ?) ;
"dem (off "2 ; shake "off "7 ;) ;
"demo (rain dem) ;
"inhibit "1 ;
"num "6 ;
```

표 1. NANOOS쉘을 이용한 내구시험기 프로그램

5. 데이타 획득 시스템에의 응용

그룹데이터 관리 기능을 시험하기 위해 NANOOS커널을 이용한다. 데이타의 획득은 두개의 타스크를 사용하여 그림 4와 같이 수행한다. 첫번째 타스크의 기능은 임시저장 장소를 갖고 있어, A/D 변환기에서 입력된 내용을 이 CIRCLE 형태의 저장 장소에 저장한다. 데이타를 계속 저장하다가 정해진 수 만큼의 데이타가 격납되면 이를 그룹으로 분리하여 시간 데이타를 첨가하여 두번째 타스크로 그룹구성완료신호를 보낸다. 두번째 타스크는 그룹단위로 데이타를 관리할 수 있는 메모리 풀(POOL)을 보유하고 있다. 적절한 판단 기준에 따라, 메모리 풀에서 공간을 확보하여 새로 받은 그룹 데이타를 저장하고 저장완료 신호를 첫번째 타스크로 보낸다. 이러한 방식의 데이타 획득 방법은 다음과 같은 장점을 이용하기 위해 채용됐다.

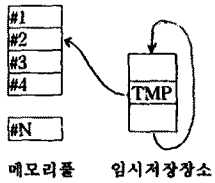

그림 4. 메모리 운용 개념도

그룹화의 장점
- 코딩 및 압축을 적용하기 쉬움
- 에러체크가 쉬움
- 그룹 마다 시간이 적혀 있으므로 시간 데이타에 의거해 불필요한 내용을 제거하기 쉬움

6. 제어기 동작중에 제어기 형태의 조정기능을 갖는 시스템

NANOOS의 사용 예로 제어기 동작 중에 제어기 형태의 조정기능을 갖는 유연성 및 가변성을 갖는 시스템을 개발해 본다. 이 시스템은 새롭게 고안된 것으로서 특히 연구용, 학습용 제어기로 적용될 경우 여러가지 시스템의 환경을 바꾸어가며 시스템을 이해할 수 있게 한다. 예를 들어 특정 파라메타의 변화에 따라 시스템의 특성이 어떻게 변화하는지를 알아볼 수 있다. 또한, 기업에서 여러개의 모델을 한가지의 제어기로 제어할 수 있는 범용 제어기를 적용하므로써, 제어기의 상품 모델 수를 줄일 수 있고, 개발 기간의 단축 등의 장점이 있다. 즉 통합된 방법으로 제어기 동작중에 제어기의 파라메타의 변경, 시스템 변수의 모니터링, 입력 센서 종류의 선택, 제어기 알고리즘의 선택, 제어 목표치의 변경, 통신 파라메타의 변경 등이 가능하다. 그림 5에 그 구성도를 보인다. 여기서, 하드웨어로는 조작자의 명령행 입력을 받는 입력수단, 명령을 제어기운영체제으로 보내는 통신단말과, 스케쥴러에 동작 시기와 실시간 시간을 제공하는 기준클럭, 제어 대상인 플랜트, 플랜트와 제어기운영체제의 인터페이스를 담당하는 입출력인터페이스장치와 통상의 제어기능은 물론 본 논문에서 제안된 매개변수 조정 등의 기능을 종합 관리하는 제어기운영체제를 구비하고, 제어기운영체제에는 NANOOS커널과 NANOOS쉘이 탑재돼있다. NANOOS쉘은 조작자의 입력내용을 임시로 받아 드려 입출력 버퍼메모리에 저장하고 입출력 버퍼메모리의 내용을 조작자에게 표시하는 조작자인터페이스 모듈, 버퍼메모리의 저장된 명령을 해석하여 다른 모듈에 명령 사항을 수행하게 하는 명령행해석기 모듈, 자료를 보관하고, 연산과 입출력 등의 데이타 임시로 기능을 담당하는 스택, 스택에 저장된 데이타에 대해 연산 처리 등의 기능을 담당하는 스택연산처리모듈이 있다. NANOOS커널에는 일련의 낱말과 명령행이 하나씩 짝을 이뤄 데이타베이스화 되여있는 낱말데이터베이스, 플랜트를 계측제어하는 제어용 알고리즘이 탑재되여 제어기능을 담당하는 주기적실행루틴과 각종 모듈의 실행 시기의 편리를 담당하는 스케쥴러로 구성된다. 그림 7에 본 예제에 적용된 주기적실행루틴의 순서도를 그림 8에 이 때의 쉘의 수행 순서도를 보인다.

7. 결론

본 논문에서는 실시간 커널과, 제어용 쉘을 이용한 제어용 운영체제의 개발을 설명하였다. 세가지 적용예를 통해 제어용 단일환경이 효과적으로 여러종류의 작업에 사용될 수 있도록 설계되어있음을 알 수 있었다.

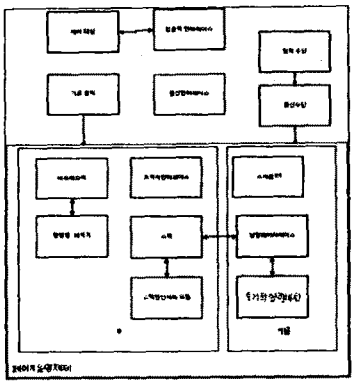

그림 6. 제어기의 구성도

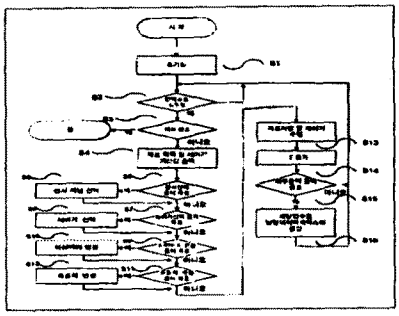

그림 7. 주기적실행루틴의 순서도

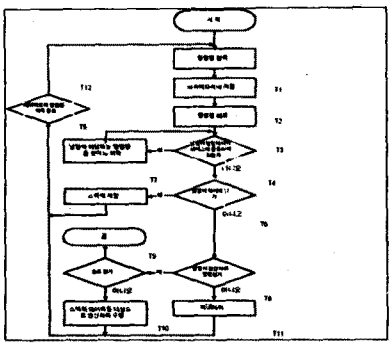

그림 8. NANOOS쉘의 순서도

8. 참고문헌

[1] WATER S.HEATH, REAL TIME SOFTWARE TECHNIQUES, VAN NOSTRAN REINHOLD, NEW YORK, 1991

[2] S.S. YANG, A STABLE FRICTION COMPENSATION SCHEME FOR MOTION CONTROL SYSTEMS, KACC, VOL. 2 OF 2. PP. 1503 ~ 1508 1991.