TMO-eCos 기반 클럭 동기화 설계 및 2족 보행 로봇 제어 응용

오용석[O] 김정국 이승연

한국외국어대학교 컴퓨터 및 정보통신공학과

justgn[O]@hufs.ac.kr jgkim@hufs.ac.kr sylee6298781@gmail.com

Clock Synchronization and Biped Robot control application based-on TMO-eCos

Yong Seok Oh[O], Jung Guk Kim, Seung yun Lee

Dept. of Office, Computer and Information Communication Engineering,

Hankuk University of Foreign Studies

요 약

분산처리 시스템은 네트워크로 연결된 프로세서들로 구성되며, 시스템 내의 각 프로세서는 고유한 클럭을 갖는다. 글로벌 시간 기준으로 볼 때 수행중인 프로세스가 유지하는 시간은 분산시스템 각각 차이가 있을 수 있으므로 일관성 있는 시간관리가 필요하다. 본 논문에서는 TMO-eCos를 기반으로 하는 분산 처리 시스템에서 각 분산 시스템간 발생할 수 있는 클럭의 불일치 문제를 해결하기 위한 클럭 동기화 기법에 관해 논한다. 점진적인 클럭 동기화 알고리즘을 구하기 위해 마스터 노드의 클럭을 글로벌 클럭으로 가정하고 슬레이브 노드들은 마스터 노드의 클럭으로 동기화하는 방법에 대하여 정의하였다. 정의한 알고리즘을 시현하기 위한 분산 노드 간 로봇 제어 프로그램을 소개 한다.

1. 서 론

네트워크로 연결된 분산처리 시스템은 각 노드마다 독립적인 클럭을 가지고 있으며 대체로 정확한 로컬 시간을 유지하고 있다. 하지만 각 노드에서 수행 중인 프로세스들의 로컬 시간은 글로벌 시간 기준으로 볼 때 여러 가지 원인으로 인해 정확히 일치하지 않으므로 단일 사건에 대한 여러 프로세서들의 시간은 미세한 차이가 있을 수 있다. 이러한 미세한 차이는 시간이 경과함에 따라 그 정도가 벌어져 치명적인 오류를 일으키게 된다. 이를 해결하기 위하여 주기적으로 적절한 한계치 이내로 클럭들을 동기화시켜 각 프로세스들의 시간을 일관성 있게 유지하는 일이 필요하다. 본 논문[1]에서 프로세스간 유지되는 클럭의 불일치의 원일을 조사하고 적절한 한계치를 고려한 클럭 동기화 기법 설계를 위하여 기반으로 한 TMO(Time-triggered Message-Triggered Object) 모델은[1] 정시보장, 객체지향, 분산환경 등의 특징을 통합하는 대표적인 분산 실시간 객체 모델로 경성 또는 연성 실시간 응용에서 사용될 수 있으며, TMO 실시간 수행을 위해 개발된 TMO 엔진으로는 미들웨어로 윈도우 환경을 지원하는 WTMOS(Windows TMO System)[2], 리눅스 환경을 위한 LTMOS(Linux TMO System)[3]가 있고, 커널 형태로는 리눅스 커널을 수정하여 커널API와 내장 실시간 스케줄러로 직접 분산 실시간 컴퓨팅을 지원하는 TMO-Linux[4]와 임베디드 커널용인 TMO-eCos[5]가 있다. TMO-eCos는 내장형 오픈 소스 마이크로 운영체제인 eCos를 기반으로 개발된 TMO지원 임베디드 커널로 로봇 제어 시스템과 같은 제어 시스템에 적용될 때 모듈화 구성이나 용량 면에서 큰 이점을 갖는다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해 소개하고 3장에서는 동기화 모델의 설계, 4장에서는 응용의 구현에 대하여 기술하며, 5장에서는 결론 및 향후 연구 방향에 대해 논한다.

2. 관련 연구

2.1. TMO 모델

TMO(Time-triggered Message-Triggered Object) 모델은 규격적 실시간 프로그래밍 설계를 위해 제안된 모델로 객체 지향 프로그래밍, 시간과 메시지에 의해 구동하는 동적 실시간 스레드의 객체 멤버화, 데드라인 스케줄링 및 분산

[1] 본 논문은 정보통신부 ITRC 및 국과연의 지원에 의한 것임

IPC 등을 통합적으로 제공하는 모델로 정시 보장성을 위한 TMO 어플리케이션들은 여러 운영체제에 미들웨어와 커널형태로 개발 되었다. TMO로 설계된 시스템은 분산 환경에서 주어진 시간 조건에 의해 수행 된다. TMO-eCos는 이러한 TMO로 구성된 프로그램을 단일 또는 복수개의 Embedded 플랫폼에서 수행하기 위해 개발된 엔진이다. 다음은 TMO의 구조<그림1>이다.

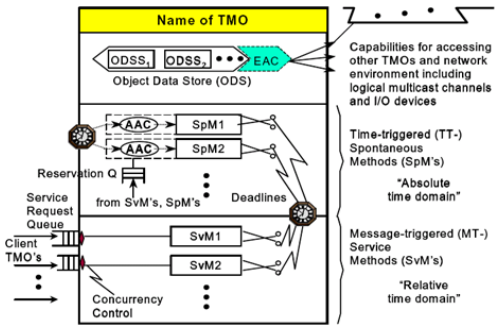

그림 1 TMO의 구조[1]

[그림 1]은 TMO의 구조를 도식화한 것으로 TMO는 일반적인 객체 모델과 같이 객체 내 자료 저장소(ODS)와 이 자료를 제어하는 SpM과 SvM으로 구성된다. SpM은 실시간 클럭에 의해 주기적으로 구동 되며, SvM은 리모트 또는 로컬 노드로부터 전달된 메시지에 의해 구동 된다. 구동된 SpM과 SvM은 주어진 데드라인 안에 그 수행을 마치도록 스케줄링 된다.

TMO는 경성 실시간 응용 프로그램뿐만 아니라 병렬 컴퓨팅 응용 프로그램에서도 사용할 수 있는 유연한 구조를 가졌으며 시스템 설계 시 정시 서비스를 보장한다. TMO의 특징은 다음과 같다.

- TMO는 ODS(Object Data Store)와 이들을 공유하는 method인 thread 군으로 구성된다.
- TMO의 method는 그 특성에 따라 두 개의 그룹으로 나누어진다. 하나는 시간 조건에 의해 구동되는 SpM(Spontaneous Method)이고 또는 분산 환경에서 클라이언트가 보내는 메시지에 의해 구동되는 SvM(Service Method)이다. SpM은 실행주기와 데드라인이 주어지며 SvM은 수행 데드라인을 가진다.
- SpM과 SvM이 객체내의 공유 데이터에 동시에 접근하여 충돌이 발생할 경우 SpM은 SvM보다 높은 우선순위를 가지는데, 이것은 설계시 시간보장의 개념을 도입하기 위해 SpM과 SvM의 시간선점을 계층화한 것으로 BCC(Basic Concurrency Constraints)라 한다.
- SpM과 SvM의 ODS에 대한 병행 접근의 동기화를 위해 CREW(Concurrent Read Exclusive Write)모니터를 함께 제공한다.
- SpM과 SvM의 설계 시 필요한 실행주기와 데드라인은 ACC(Autonomous Activation Condition)에 정의되며 기본 단위는 1/1000초 이다.

2.2. TMO -eCos

TMO-eCos는 TMO기반의 분산 실시간 객체의 실행을 위해 다음과 같은 기능을 제공한다.

- CPU의 성능에 따라 최소 30us에서 10ms까지의 정밀로도 Time-triggered 스레드와 message-triggered 스레드의 데드라인 구동 실시간 스케줄링 제공 한다.
- Time-triggered 실시간 스레드의 정시 구동(on-time activation)을 제공 한다.
- 논리적인 멀티 캐스트 IPC 서브시스템을 제공하며, 분산 및 로컬메시지를 통한 message-triggered 실시간 스레드 구동 기능을 제공 한다.
- TMOSL(TMO Support Library)을 이용한 TMO 기반의 프로그래밍 방법을 제공 한다.

TMO-eCos커널을 이용하여 분산 실시간 어플리케이션을 개발하는 방법에는 2가지가 있다.

- 스레드 기반의 방식으로 TMO-eCos가 제공하는 기본적인 API를 이용
- TMO 기반 방식으로 TMOSL를 이용

TMO-eCos의 구조는 다음<그람2>와 같다.

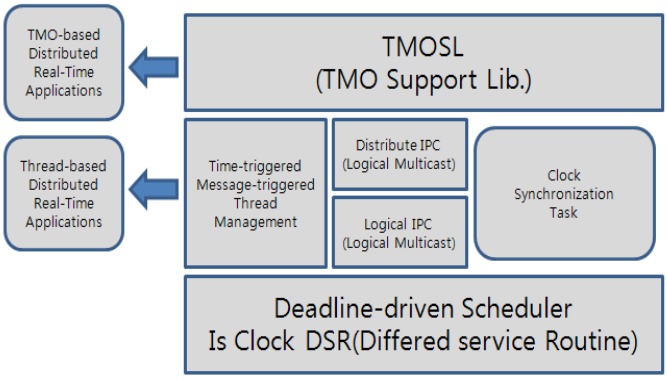

그림 2 TMO-eCos 구조

2.3 기존 분산 시스템에서의 동기화 문제점

분산 시스템에서 발생하는 사건 시각을 정확히 설정할 수 있다는 것은 제반 기능의 동기화, 순서화 및 일관성 유지에 큰 영향을 준다. 이에 따라 분산된 시스템 자원이 하나의 전체적인 시스템을 이루는 경우 각 자원간에는 더욱 정밀한 시각 동기화 방법이 요구되게 되었다. 분산 시스템에 있어서, 응용프로그램은 시스템의 여러 노드에서 concurrent하게 수행되는 많은 프로세스를 가지며 올바른 결과를 얻기 위해서는 분산 응용프로그램은 각각의 노드간에 클럭 동기화가 반드시 필요하다.

분산 처리 시스템에서는 각 분산 노드간 클럭이 글로벌 클럭으로 정확한 시간에 작업을 처리해야만 한다. 그러나 네트워크로 연결되어 있는 분산 시스템에서 각 분산 노드마다 독립적인 클럭을 가지고 있으나 각 분산 노드에서 유지하고 있는 로컬 클럭은 글로벌 클럭에서 봤을 때 다음과 같은 원인에 의하여 노드간 시간차이가 발생할 수 있다.

- 서로 상이한 시스템의 클럭 오차 누적
- 동작 환경에 따른 각 시스템의 클럭의 미세한 변화
- 시스템 재 가동 시 글로벌 시간과의 편차

3. TMO-eCos 기반의 분산시스템 동기화 모델의 설계 및 구현

3.1 동기화 기법의 설계

클럭 동기화는 글로벌 클럭의 불일치로 인한 실시간 분산 시스템에서 작업의 처리 불능과 처리 지연을 미연에 방지하기 위하여 아래와 같이 설계 되었다.

- 마스터 노드 : 마스터 노드의 클럭을 글로벌 클럭으로 가정한다. 또한 주기적으로 자신의 클럭을 슬레이브 노드에게 동기화 메시지를 통하여 주기적으로 전송한다.
- 슬레이브 노드: 마스터 노드의 동기화 메시지를 수신하면 자신의 클럭을 검사한 후, 자신의 클럭이 글로벌 클럭과 오차가 발생했을 경우 자신의 클럭을 점진적으로 감소시키거나 증가시키는 방법을 선택한다.

3.2 동기화 기법의 구현

클럭이 불 일치하는 경우 동기화 과정은 두 가지로 구분 할 있다.

- 슬레이브의 클럭이 빠른 경우
- 슬레이브의 클럭이 느린 경우

슬레이브의 클럭을 C 라 하고 글로벌 클럭을 t 이고 노드 p 의 클럭이 Cp(t) 이라 하고 동기화 수행을 빠른 클럭을 기준으로 할 때 슬레이브 노드의 클럭과 글로벌 클럭 사이의 진행 비율에 대한 식을 얻을 수 있다. 현재 i 번째 클럭에 대하여 슬레이브의 동기화 i+1번째 클럭을 구하는 식은 다음과 같다.

$Cp(t) > t$

$$Cp(t)_{i+1} = Cp(t)_i - (1 - (t_i \ / \ Cp(t)_i)) * t_i \quad --(1)$$

$Cp(t) < t$

$$Cp(t)_{i+1} = (1 - (Cp(t)_i / t_i)) * Cp(t)_i + Cp(t)_i \quad --(2)$$

첫째 Cp(t) >t 인 경우 t 의 진행에 대한 Cp(t)의 진행 비율은 t/Cp(t) 이다. 즉 슬레이브의 클럭이 빠른 경우이므로 글로벌 클럭으로 동기화 하기 위해서 매 클럭 마다 진행 비율을 산정하고 식(1)을 이용하여 점직적으로 클럭을 줄여 나간다.

둘째 Cp(t) < t인 경우는 Cp(t) 의 진행에 대한 t 의 진행 비율은 Cp(t)/t 이다. 즉 슬레이브 클럭이 느린 경우이므로 글로벌 클럭으로 동기화 하기 위해서 매 클럭 마다 진행 비율을 산정하고 식(2)와 같은 방법으로 점진적으로 클럭을 증가해 나간다.

다음은 위 식 (1)과 (2)를 사용한 동기화 수행이다.

- 초기 동기화는 마스터가 초기화 메시지를 송신하면 모든 슬레이브는 동일한 시간에 동기화 메시지를 수신한다고 가정하고 TMO의 handshake[6]를 수정하여 분산 노드 간 초기 동기화 작업을 수행한다. 초기화 단계에서는 마스터로부터 오는 초기화 단계 동기화 메시지를 통해서 모든 분산 노드들은 마스터 노드의 클럭으로 초기화를 수행한다.

- 작업 수행 중 동기화는 마스터가 주기적으로 동기화 메시지를 전송하고 동기화 메시지를 수신한 슬레이브는 동기화 작업 수행 여부를 결정하고 슬레이브의 클럭이 글로벌 클럭보다 느리다면 식(2)를 사용하여 점진적으로 동기화 작업을 수행하며 슬레이브 클럭이 글로벌 클럭 보다 빠르다면 식(1)을 사용하여 점진적으로 동기화를 수행한다.

- 작업 수행 도중 새로운 노드가 분산 시스템에 추가가 되면 마스터로부터 주기적으로 오는 동기화 메시지를 수신하게 되고 이때부터 새로운 노드는 동기화를 수행하고 동기화 수행 완료 후에 작업을 수행하게 된다.

본 논문에서는 수행 중인 작업의 일관성 문제를 해결하기 위하여 현재 수행중인 작업의 처리 완료 시점부터 동기화를 시작하는 방법을 사용하여 구현하였다.

SvM은 메시지에 의한 구동이므로 동기화에 대한 고려 없이 처리되는 구조 이다. TMO의 ORT(Official Release Time) 적용하여 SvM의 메시지의 처리는 다음 연구과제로 남긴다.

<그림 3> 은 동기화 수행 과정을 도식화 한 것이다.

4 동기화 모델을 따른 TMO-eCos를 이용한 2족 보행 로봇 제어 프로그램

TMO-eCos 기반의 동기화 모델을 적용시킨 이족로봇 제어 프로그램은 분산 시스템에서의 동기화된 동작을 유도하는 응용프로그램이다. 다음은 동기화 모델을 적용시키기 위한 노드 간 동기화 구조 이다.

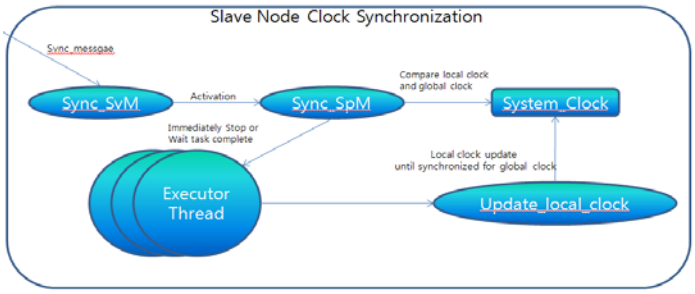

그림 3 Slave Clock Synchronization

- 마스터 노드
  - 마스터 노드의 Clock_SpM은 주기적으로 자신의 시스템 클럭을 슬레이브에게 실시간 분산 동기화 메시지로 전파한다.
  - 마스터 노드의 Sync_SvM은 슬레이브 노드로부터 오는 Reply메시지를 수신하고 슬레이브의 정보를 갱신하는 작업을 수행한다.
- 슬레이브 노드
  - 슬레이브 노드의 Sync_SpM은 Sync_SvM의 활성/비활성화에 따라 주기적으로 자신의 System 클럭을 갱신한다.
  - 슬레이브의 Sync_SvM은 마스터 노드로부터 동기화 메시지를 수신하고 동기화 작업을 하는 SpM의 활성/비활성을 결정하는 작업을 수행한다.

동기화 모델을 지원하기 위해서 다음과 같은 기능들을 지원한다.

- Get_current_clock : 마스터 노드의 현재 클럭을 가지고 오는 메소드
- Send_sync_msg : 마스터 노드가 슬레이브 노드로 주기적으로 클럭 동기화를 알아 보기 위해서 보내는 클럭 동기화 메시지를 보내는 메소드
- Send_reply : 동기화 메시지를 받은 슬레이브 노드가 마스터 노드에게 메시지를 수신하였다는 것을 전달하고 자신이 현재 분산 시스템의 구성에 포함되어 있다는 것을 알리는 메소드
- Update_currnet_clock : 동기화를 수행하기 위해서 자신의 시스템 클럭을 조정하고 주기적으로 실행되며 각 주기마다 점진적으로 클럭을 줄이거나 늘여나가는 메소드

다음<그림4>는 동기화 모델을 응용한 로봇제어 프로그램을 도식화 한 것이다.

- Time_triggered로 동작하는 SpM은 시리얼을 통한 로봇의 구동과 네트워크를 통한 클럭 동기화를 주기적으로 수행하며 자신의 로컬 클럭을 주기적으로 분석하는 역할을 한다.

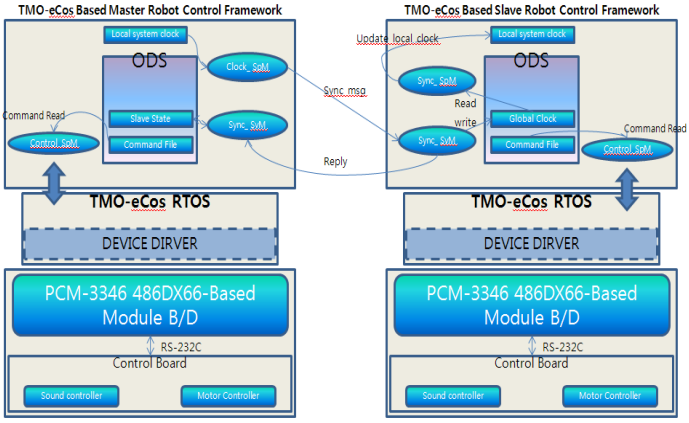

그림 4 TMO-eCos Based Control Framwork

- Message-triggered로 동작하는 SvM들은 마스터 노드로부터 오는 동기화 메시지를 수신하고 동기화 작업을 하는 SpM들을 활성/비활성화 시키는 역할과 슬레이브 노드에서 오는 상태 메시지들을 수신하고 슬레이브들의 상태는 ODS에서 갱신하는 작업을 수행한다.

TMO 기반 2족 보행 로봇 제어 응용프로그램은 로봇 구동을 위한 SpM, 동기화 작업을 위한 SvM, 종합적인 구동을 지시하는 제어 SpM들의 컴포넌트 메소드들로 구성된다. 다음은 각 메소드 구성 요소에 대한 설명이다.

- Control_SpM : 로봇의 활동 주기를 간격으로 command file 에서 명령을 입력 받아 로봇에게 구동 명령을 내린다.
- Sync_SvM : sync_SpM으로부터 동기화 메시지를 수신하고 동기화 작업을 수행하는 Csync_SpM을 구동시키는 역할을 한다.
- Csync_SpM : sync_SvM 구동을 인가 받으면 자신의 clock synchronization 을 수행한다

다음<그림5>는 2족 보행 로봇의 제어 구조도 이다.

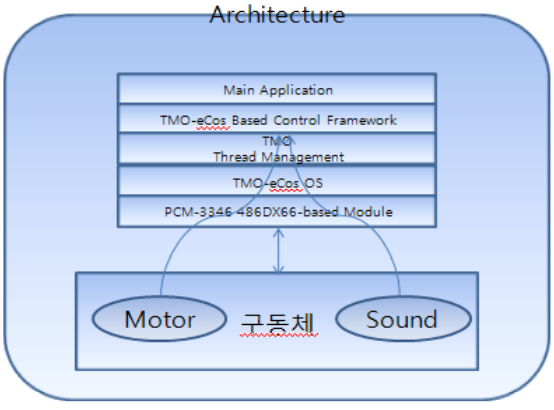

그림 5 2족 보행 로봇 제어 구조도

4.1 2족 보행 로봇 제어 시나리오

동기화모델을 적용시킨 2족 보행 로봇의 하드웨어 소프트웨어 사양은 다음과 같다.

- 구동부 : MR-C3024 컨트롤러를 탑재한 HSR-8498HB 모터 및 IR(적외선)센서로 구성된 2족 보행 로봇
- 제어부 : 메인 로직이 있는 보드로 PCM-3346 486DX66-base module보드로 본 연구에서 개발된 제어 프레임워크 탑재

TMO-eCos 기반의 동기화 모델을 적용시킨 2족 보행 로봇 제어 응용 프로그램

- 응용 프로그램의 동작은 control_SpM에 의해서 구동되며 Serial에 의해서 제어된다.
- Control_SpM은 명령을 받아 응용 프로그램을 구동시킨다.
- 응용 프로그램은 제어 프레임워크와 인터페이스 하의 TMO 객체의 실시간 객체를 기반으로 2족 보행 로봇을 제어 한다.

2 족 보행 로봇의 동작은 마스터 노드는 로봇의 동작 주기에 해당하는 소리를 내며 다른 분산 노드들은 그 소리에 마추어 같은 동작을 일관성 있게 하는 동작을 예로 구현하였다.

마스터 노드에 연결된 2족 보행 로봇은 마스터 노드로부터 일정한 주기로 소리를 내는 명령어를 받아서 그 소리를 낸다.

슬레이브 노드에 연결된 2족 보행 로봇은 슬레이브 노드로부터 일정한 주기로 동작을 하는 명령을 수신하고 동작을 수행한다.

로봇의 수행 도중 동기화 문제가 발생했을 때 로봇에게 명령을 내리는 분산 시스템은 즉시 동기화 수행을 시작하게 되고 로봇은 동기화에 맞추어 동작을 수행하게 된다.

5. 결론 및 향후과제

본 논문은 실시간 제어 시스템에 적합한 모델로 개발된 마이크로 임베디드 운영체제인 TMO-eCos를 기반으로 하는 실시간 분산처리 시스템의 클럭 동기화 모델과 이를 이용한 2족 보행 로봇의 제어 프레임워크 응용 모델에 대하여 기술하였다. TMO 모델을 이용한 실시간 분산 시스템의 클럭 동기화 방법은 분산 시스템 사이의 동기화 문제를 해결하고 나아가 분산 시스템간 시간 보장성이 있는 작업의 처리를 위한 구조로 분산 시스템간의 동기화 작업 처리 문제를 해결할 수 있을 것으로 본다.

향후 연구 과제로는 현재 클럭 동기화는 TMO 분산 시스템의 동기화이므로 일반 분산 시스템 상에서의 동기화 문제의 해결과 clock 동기화로 인한 일반 시스템의 문제에 대한 보정이 필요하다.

참고문헌

1] Kim, J. G., et al, “TMO-Linux: A Linux-based Real-time Operating System Supporting Execution of TMO’s,” Proc. IEEE Int’l Symposium, ISORC2002, Washington DC, Apr. 28, 2002.

2] J.G. Kim, M.H. Kim, B.J. Min and D.B. Im, “A Soft Real-Time TMO Platform – WTMOS – and Implementation Techniques.” Proc. 1st IEEE International Symposium on Object-oriented Real-Time Distributed Computing, pp.256-264, April 1997

3] J.G. Kim and Sang-Young Cho, “LTMOS: An Execution engine for TMO-Based Real-Time Distributed Object.” Proc. PDPRA’00 Vol. V, pp2713-2718, Las Vegas, June 2000.

4] 박상현, “분산 TMO 리눅스 운영체제”, 한국외국어대학교 컴퓨터공학과 석사학위논문, 2001년 12월

5] 김광 “TMO-eCos : 분산 실시간 객체 모델을 지원하는 eCos 기반의 마이크로 운영체제”, 한양대학교 컴퓨터공학과 박사학위논문 2005년 12월.

6] J.G. Kim, M.H. Kim “TMO-eCos: An eCos-based Real-time micro Operating System Supporting Execution of a TMO Structured Program” Proc. ISORC2005, May. 2005, Seattle