2014년 5월 전자공학회 논문지 제 51권 제 5 호
Journal of The Institute of Electronics and Information Engineers Vol. 51, NO. 5, May 2014 http://dx.doi.org/10.5573/ieie.2014.51.5.207

논문 2014-51-5-23

CAN과 RTOS를 내장한 소형 실시간 시스템 설계 기법

(Design Scheme of A Micro Real-Time Control System with CAN and RTOS)

임 영 규*, 김 동 성**

(Young-Gyu Lim and Dong-Seoung Kim©)

요 약

본 논문은 초소형 센서노드(이하 노드)에서 인터럽트 처리와 데이터 전송에 대한 지연에 대한 문제들을 해결하기 위해 Micro Real-Time Control System (MRTCS)을 제안한다. MRTCS은 제어노드와 Controller Area Network (CAN) 기반의 노드로 구성되어졌다. 제어노드는 소형 마이크로 제어기 (sMCU)에 Real-Time Operating System (RTOS)를 내장하여 설계하였다. 노드들은 sMCU 없는 CAN 기반의 디바이스이며, 다중 디지털 입출력과 CAN 제어기를 가지고 있다. 소형 실시간 시스템 설계를 위해, 오픈소스인 OCTAVE v3.6.4를 이용하여 시스템 성능에 대한 모의실험을 실시하였다. 모의실험을 통해 제안된 설계 기법을 이용할 경우 인터럽트 처리와 데이터 전송에 대한 지연이 감소하여 시스템 성능이 증가함을 알 수 있었다. MRTCS이 다양한 실시간 제어 시스템에 적용 가능함을 검증하였다.

Abstract

In this paper, we propose a Micro Real-Time Control System (MRTCS) for decreasing the delay during interrupts processing and data transfer on sensor nodes. The MRTCS consists of a control, sensor nodes based on Controller Area Network (CAN) device. The control node was designed with Real Time Operating System (RTOS) on top of the small Micro Control Unit (sMCU). Sensor nodes have the CAN device without sMCU, which have multiple Digital Inputs, Outputs (DI/DO) and the CAN controller. We have evaluated with OCTAVE v3.6.4 from open source for system performance. Simulation results show that the system performance was increased through the delay reducing for interrupt processing and internal data transfer. We verify that a proposed MRTCS approach will be adapted to various real-time control system.

Keywords : Micro Real-Time Control System, CAN, RTOS

I. 서 론

실시간 임베디드 시스템의 다양한 응용 및 기술적 발전으로 소형 마아미크로 제어기 (sMCU)의 설계 및 구현기법이 연구되어져 오고 있다. 또한 sMCU에 탑재 가능한 여러 종류의 RTOS들이 연구되었다[1~2]. 8/16 비트 sMCU들은 다양한 센서들의 디지털 입력 (DI) 처리와 제어 신호들에 대한 디지털 출력(DO)을 할 수 있는 기능을 가지고 있다. 그리고 통신을 위한 CAN 제어기를 내장하고 있어 제어 시스템과 센서노드(이하 노드)들 간의 데이터 전송을 위한 통신 수단을 제공한다[1].

실시간 처리는 제어 시스템과 각 노드 간의 데이터

* 학생회원, ** 정회원, 금오공과대학교
(Department of IT Convergence Engineering, Kumoh National Institute of Technology)
© Corresponding Author(E-mail: dskim@kumoh.ac.kr)
※ 본 연구는 교육부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임 (NRF-2012H1B8A2026109).
접수일자: 2014년1월13일, 수정일자: 2014년3월3일
수정완료: 2014년5월 2일

결함을 배제하여야 하며 신뢰성 있는 실시간 데이터 전송을 요구한다. 최근에는 노드 구성비용이 최적화 되고 동시에 단순한 기능을 가진 디바이스 형태의 소형 마이크로 제어기들이 임베디드 시스템에 응용되고 있다.

기존의 연구들[3~16]은 펌웨어 기반의 소프트웨어를 이용한 병행처리로 노드 내에서 타임 슬롯을 이용한 방식으로 실시간 제어를 하였다. 그러나 이러한 방식은 인터럽트 처리와 내부 데이터 처리에 대한 지연을 증가시키는 문제들이 발생할 수 있다. 이를 해결하기 위해 각 노드를 디바이스 형태를 사용함으로써 노드 증가에 따른 비용감소 및 결함 발생으로 인한 인터럽트 처리와 내부 데이터 전송에 대한 지연을 감소시킬 수 있다.

제안하는 MRTCS은 언급한 문제들을 해결하기 위해 CAN[17]과 차량용 RTOS를 적용하고 각 노드를 디바이스 구조로 단순화시켰다[18~19].

제안하는 MRTCS은 기존 연구들[3~16]보다 높은 신뢰성과 실시간 데이터 처리를 보장하였고 이를 통해 인터럽트 처리 및 데이터 전송 지연을 감소시켰다.

본 논문은 I장의 서론에 이어, Ⅱ장에서는 기존 연구들에 대한 문제들을 살펴본다. Ⅲ장에서는 MRTCS의 구성을 소개하고, Ⅳ장에서는 모의실험을 통한 결과를 기술한다. 마지막 V장에서는 본 논문에서 제안하는 MRTCS의 적용 가능함에 대한 결론을 내리고, 향후 연구 방향을 기술한다.

Ⅱ. 기초 설계의 문제점 분석

Ⅱ장에서는 기존 연구들에 대해 제어노드와 노드의 구성 및 확장성에 관한 문제점들을 살펴본다.

1. 제어노드 설계의 문제

기존 연구들에서는 펌웨어 기반에서 sMCU를 제어하였다[3~16]. 펌웨어 기반으로 작성한 제어 소프트웨어 경우 내장된 타이머/계수기를 이용하여 타임 슬롯을 만들고, 주어진 타임 슬롯 시간 내에서 각 DI/DO 들의 제어 동작들이 완료되어야 한다. 그렇지 못한 경우 다음 번 DI/DO 들의 처리에 있어 타임 슬롯을 놓치게 되어 지연이 발생하게 된다.

전송방식의 경우 제어노드에서 데이터 요청 신호를 보내고 노드에서 데이터(또는 출력완료 신호)를 전송하는 양방향 방식을 사용하였다. 이것은 요청과 응답이라는 양방향 통신이 필요하기 때문에 데이터 전송상의 오버헤드가 발생한다. 제어 노드와 노드의 sMCU는 데이터 전송과 입출력 처리를 위해 인터럽트들을 사용하며 이들에 대한 인터럽트 처리로 인하여 지연이 발생하는 문제가 발생한다.

기존 연구[14]에서는 RTOS를 적용하여 태스크 기반으로 DI/DO 와 데이터 전송을 제어하였다. sMCU의 리소스를 고려하지 않고 많은 태스크를 운영하면 리소스 고갈에 따른 오류발생 빈도를 증가 시키는 문제가 발생할 수 있다.

2. 노드 구성 및 확정성 분석

기존 연구들은 각 노드에 모두 sMCU를 이용하였고[3~16], 시스템의 데이터 전송방식은 CAN 통신을 사용하였다. 이는 노드의 경량화를 어렵게 하고 구성비용과 노드 구성시간을 증가시키게 된다.

노드는 다양한 DI/DO들을 제어하기 위해 제어노드로부터 받은 프로토콜 분석과정으로 인한 내부 처리 지연문제가 발생한다. 또한 노드에서 입력, 처리응답, CAN 프레임 제작 그리고 제어노드로의 데이터 전송에 대한 내부 처리 지연에 대한 문제가 발생한다.

CAN 2.0A[17] 명세를 사용한 기존 연구들은 200개 미만의 노드와 오직 2.0A 명세만 지원하였다. 실시간 제어 시스템의 규모가 증가하여 200개 이상의 시스템을 구성하는 경우 노드의 확장에 제한을 받는다. 2.0B[17] 명세를 지원하는 노드가 추가되는 경우 이 노드로 부터의 데이터 수신은 가능하지만, 데이터 손실 발생과 이로 인한 노드 확장의 문제가 발생한다.

Ⅲ. MRTCS

Ⅲ장에서는 MRTCS의 구성과 CAN 통신, 그리고 시스템 성능 평가 방법에 관해 논한다.

1. MRTCS 구성

Ⅲ장에서는 MRTCS을 구성하는 각 요소들에 대해 살펴보고자 한다. 처리 지연과 데이터의 신뢰성 문제들은 실시간 제어 시스템 구성에 있어 매우 중요한 요소이다. 그림 1은 제안하는 MRTCS의 구성도를 보여준다.

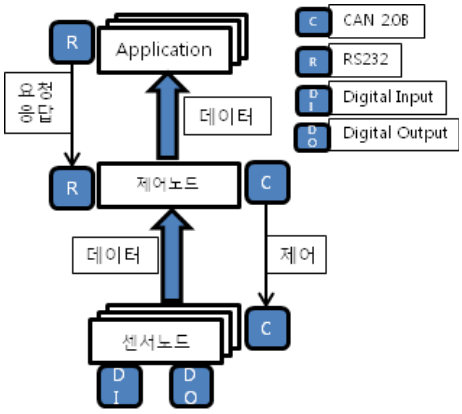

그림 1. MRTCS의 구성도.
Fig. 1. System overview of MRTCS.

가. 제어노드 구성

제어노드에서 노드를 직접 제어하는 방식을 도입하였다. 제어노드는 sMCU 하나에 RTOS를 내장하고 4개의 태스크를 이용하여 노드들을 제어한다. 노드와의 데이터 전송은 CAN 통신을 이용하였다. 제어노드는에 적용된 sMCU는 마이크로칩사의 PIC18F4580[20]을 이용하였다. 노드와는 CAN 2.0B 명세를 이용하여 통신한다. 상위 응용과의 확장을 위해 RS232 통신을 추가로 제공한다. 제안하는 MRTCS에서의 데이터 전송에 따른 시스템의 단방향 통신 시간의 관계는 식 (1)과 같다.

$$T_c = \frac{L(N-1)}{2} * C. \quad (1)$$

T_c : 시스템통신시간
L : 전송 데이터 길이
N : 전체 노드 수
C : CAN통신 속도$(bytes)$

나. RTOS 선정 및 분석

MRTCS에서 제어노드의 sMCU에 RTOS인 PICOS18[18]을 적용하였다. PICOS18은 자동차용 임베디드 시스템의 제어와 데이터 처리의 표준을 제공하는 Open Systems and their Interfaces for the Electronics in Motor Vehicles (OSEK/VDX) 를 정의한 구조로 설계되었다[18~19]. 차량용 임베디드 시스템에 적용되는 이점을 고려하면 PICOS18은 실시간성과 데이터 전송의 무결성, 그리고 높은 신뢰도를 가진다. OSEK/VDX는 CAN 통신을 지원하기 때문에 실시간 제어 시스템에 매우 적합하다. OSEK/VDX에서 소개한 공개 아키텍처는 3 분야로 다음과 같다.

- 통신(제어장치 내부 및 외부 데이터 교환)
- 운영체제(ECU 소프트웨어의 실시간 실행과 다른 OSEK/VDX 모듈을 위한 기반)
- 네트워크 관리(구성 결정 및 모니터링).

OSEK/VDX의 적용으로 인한 장점은 다음과 같다.

- 비용과 개발시간의 절약
- 다양한 회사들의 제어장치 소프트웨어의 질적 향상
- 다른 아키텍처로 설계된 제어장치를 위한 표준화된 인터페이스
- 하드웨어의 추가 없이 종합적인 시스템 성능 향상을 위한 순차적이고 분산처리 된 지능화된 시스템.

OSK/VDX에서 제안하는 태스크의 4가지 상태는 그림 2와 같다. PIC18OS의 가장 큰 특징은 선점형 멀티태스킹을 지원하며 오직 하나의 태스크 만이 우선순위에 따라 동작한다. 특정 시간대에 가장 높은 우선순위를 가지는 태스크를 실행한다.

실행된 태스크는 이벤트를 기다릴 수 있고, 우선순위가 낮은 태스크 실행이 스케쥴러에 의해 요구되면 높은 우선순위를 가지는 태스크는 정해진 실행시간 동안 실행된 다음 대기 상태에 놓이게 된다. 이벤트가 발생하는 시점에서 다시 동작 상태로 된다. 이때 우선순위가 낮은 태스크는 대기 상태가 된다. 우선순위 기반의 태

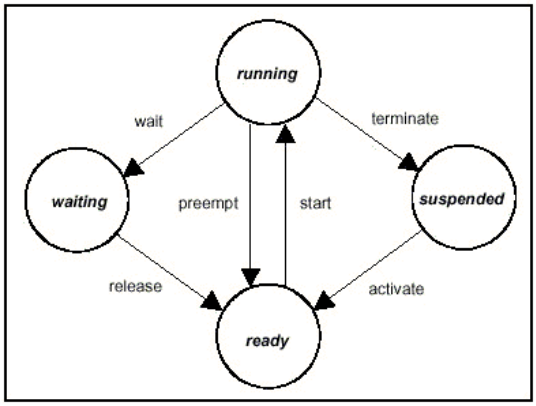

그림 2. OSK/VDX의 태스크 상태도.
Fig. 2. State of a task on OSK/VDX.

스크 운용이 가능하기 때문에 sMCU 자원을 적게 사용하며 실시간 처리에 대한 신뢰도는 더 증가한다.

다. PICOS18 커널의 구조

제안하는 논문에 이용한 PICOS18의 커널 구성은 그림 3과 같은 계층구조를 가진다[18]. PICOS18은 PIC18F4580 내의 제어기들의 기능을 커널과 하나의 독립된 태스크로 운영한다.

각 계층은 매니저로 표시하며 기능에 대한 설명은 표 1 에 나타내었다.

그림 3. PICOS18의 계층구조.
Fig. 3. Hierarchical structure of PICOS18.

표 1. PICOS18 계층구조 설명.
Table 1. Summary of Hierarchical structure for PICOS18.

기능	설명
커널 코어	Init, Scheduler, 태스크 Manager로 구성되며, 우선순위 기반의 태스크 스케줄링을 담당
알람매니저	sMCU 내부의 TIMER0 인터럽트를 이용하여 주기적 일람을 발생시킴
후킹루틴	개발자로 하여금 프로그램의 함수 호출이 가능하도록 지원
태스크매니저	생성된 태스크의 상태를 관리
이벤트매니저	이벤트 생성 및 태스크의 이벤트 대기, 상태 읽기 기능을 제공
인트럽트 매니저	인터럽트 서비스를 관리

라. 태스크 기능

제어노드는 4개의 태스크를 가지며 표 2에 태스크의 기능들과 우선순위를 표시하였다. 태스크 1번은 상위 모니터링 시스템으로의 확장 및 디버깅을 관리한다. 태스크 2번은 노드로부터 수집되는 데이터를 처리하며, 태스크 3번은 노드에서 대해 출력을 지시할 수 있도록 설계하였다. 태스크 4번은 CAN 통신 제어 역할을 담당한다. 우선순위 값이 작을수록 더 높은 우선순위를 가진다. 따라서 DI에 대한 처리를 한 후 이에 대한 DO 처리를 하므로 태스크 2번의 우선순위를 가장 높게 설정하였다.

그림 4는 PICOS18에서 태스크를 생성하는 예제 코드를 나타내었다.

표 2. 태스크 별 기능과 우선순위.
Table 2. Function and priority of each Tasks.

태스크	DI	DO	CAN	RS232	우선순위
#1				O	4
#2	O				1
#3		O			2
#4			O		3

```
#define DEFAULT_STACK_SIZE        128
Declare TASK(TASK0);
#pragma                udata        STACK_A
volatile unsigned char stack0[DEFAULT_STACK_SIZE];
#pragma                udata
#pragma                romdata                DESC_ROM
const rom unsigned int descromarea;
//* ----------------------------  태스크 0
rom_desc_tsk rom_desc_TASK0 = {
        태스크0_PRIO,          // prioinit from 0 to 15
        stack0,                // stack address (16 bits)
        태스크0,               // start address (16 bits)
        READY,                 // state at init phase
        태스크0_ID,            // id_tsk from 0 to 15
        sizeof(stack0)         // stack size     (16 bits)
};

rom_desc_tsk end = {
        0x00,                  // prioinit from 0 to 15
        0x0000,                // stack address (16 bits)
        0x0000,                // start address (16 bits)
        0x00,                  // state at init phase
        0x00,                  // id_tsk from 0 to 15
        0x0000                 // stack size     (16 bits)
};
volatile rom unsigned int * 태스크desc_addr =
(&(descromarea)+1);
```

그림 4. PICOS18의 태스크 예제 코드.
Fig. 4. Sample codes of a task in PICOS18.

2. CAN 통신

CAN 통신은 현재 산업 데이터 표준으로 자리매김하고 있다[17]. CAN 통신은 여러 개의 CAN 디바이스가 서로 데이터를 전송할 수 있는 경제적이며 안정적인 네트워크를 제공한다. 높은 우선순위의 데이터를 먼저 처리하기 때문에 전기적인 노이즈에 매우 강하며 2023까지 노드를 연결할 수 있다. 그리고 멀티 마스터 구성이 가능하여 모든 노드들이 제어 노드가 될 수 있으며, 하드웨어적인 오류 보정 기능이 있고 필터링을 이용하여 설정된 ID만 수신 할 수 있다.

그림 5. CAN 2.0B 확장 프레임 형식.
Fig. 5. CAN 2.0B Ext. frame format.

본 논문에서 사용하는 CAN 데이터 확장 포맷 2.0B는 그림 5에 나타내었다. 노드 ID를 29비트까지 표현하기 때문에 확장이 용이하다. MRTCS에서 각 노드들은 MCP250XX[20] 범용 CAN 디바이스를 이용하였다.

CAN 디바이스의 특징은 다음과 같다.

- sMCU 코어를 가지지 않음
- CAN 규약 2.0B 명세, 1 Mbps 전송 속도
- 범용 DI/DO 8 개, 펄스 출력
- 메시지 수신 버퍼 2개, 메시지 자동 전송 버퍼 3개.

3. 시스템 성능 평가 방법

본 논문에서는 단 방향 데이터 전송 방식을 사용한다. 이것을 식으로 표시하면 식 (2)와 같으며 식 (1)의 결과에 대해 태스크 스케쥴링 시간과 이벤트 처리 시간을 추가하한 것으로 전체 시스템 성능을 평가할 수 있다.

$$S_p = [(L(M-1)/2]*[C][I_c * E_d].$$

S_p : 시스템성능
L : CAN프레임 길이
M: 센서노드수
T: CAN통신속도(bps)
I_c : 인터럽트개수
E_d : 인터럽트처리시간 (2)

Ⅳ. 모의실험

Ⅳ장에서는 MRTCS의 모의실험을 수행하기 위한 환경 설정과 모의실험을 수행하고 그 결과를 분석하였다.

1. 모의실험 환경 설정

모의실험을 위해 고려 가능한 환경변수들을 정의하였다. 가능한 최적의 신뢰성을 도출하기 위해 시스템 성능을 지연시키는 항목만 변수 값으로 설정하기로 하였다. 기본적으로 정의되는 환경변수는 표 3과 같다.

표 3. 모의실험 환경변수 및 변수 값.
Table 3. Environment variables and values for simulations.

순	환경변수	변수설정값
1	MCU 클럭	20 Mhz
2	태스크 주기	20 ms
3	CAN 프레임생성	CAN2.0B
4	BUS 송신시간	500 Kb/s
5	Tx 시간	전송
6	Rx 시간	수신
7	CAN 프레임분해	데이터획득
8	명령분리	파싱
9	노드의 출력처리	출력
10	DI/DO인터럽트	문맥교환
11	노드의 입력처리	DI
12	노드의 Tx 시간	전송
13	노드의 Rx 시간	수신
14	인터럽트 개수	2-10

제어노드에서 전송받은 CAN 프레임은 CAN 디바이스 내부에서 데이터로 인식되어 DO 동작이 실행된다. 입력의 경우 노드의 센서들로부터 데이터를 입력받아 CAN 2.0B 프레임 조합 없이 제어노드로 전송한다. 데이터 전송은 단방향 데이터 전송을 이용하기 때문에 노드들과 제어노드와의 전송 시간을 줄일 수 있다.

또한 각 노드들은 디바이스 기반으로 구성되기 때문에 인터럽트 처리가 없고 이로 인한 지연도 발생하지 않는다. 제어노드에서 출력을 위한 프레임을 전송하는 경우, 이를 받은 노느는 DO 동작을 바로 수행하기 때문에 프레임 분리 등에 따르는 내부 처리 지연을 감소시킬 수 있다.

모의실험을 위한 환경변수 중 CAN 통신 속도는 500 Kbps로 설정하였고 인터럽트 개수는 2, 4, 8, 10개로 제한하였다. 노드는 CAN 프레임을 생성하지 않고 송수신 시 데이터가 들어 있는 부분만 이용한다. 데이터 내부에는 노드의 내부 DI/DO들의 제어를 위한 레지스터 번호와 각각의 제어에 필요한 데이터가 들어있다. 따라서 제어노드와 노드에서의 CAN 프레임 생성 및 분리에 따른 내부 처리 지연은 발생하지 않는다. 노드의 ID 및 DI/DO 들에 대한 설정은 마이크로칩사에서 제공하는 프로그램을 이용하며, 그림 6, 7에 각 노드들의 연결 구

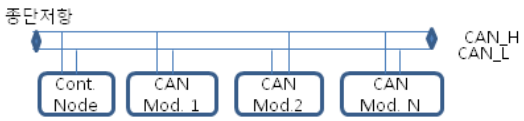

그림 6. 노드의 연결 구성도.
Fig. 6. Connections diagram of nodes.

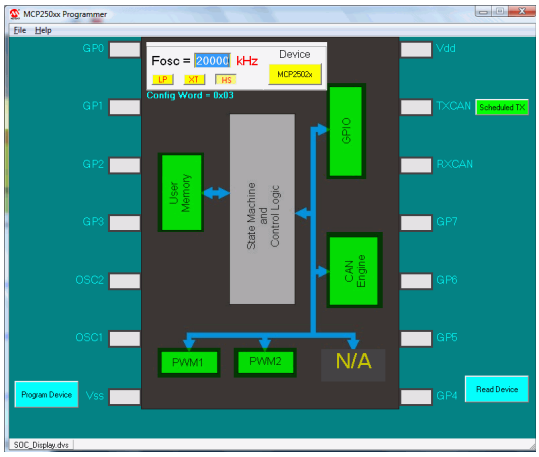

그림 7. 노드 환경설정 프로그램의 예.
Fig. 7. Example of node configuration.

성도 및 노드 환경설정 프로그램을 나타내었다.

모의실험을 통해 MRTCS의 성능을 평가하였다. 성능평가는 식 (2)를 이용하고 CAN 2.0A 명세와 sMCU로 구성한 노드와 CAN 2.0B 명세와 CAN 디바이스 형태로 구성한 노드로 나누어 실험을 하였다.

CAN 2.0A 명세의 최대 데이터 비트 수는 134이며, 2.0B 명세의 최대 데이터 비트 수는 159 비트이다. MRTCS에서 제어노드와 노드 간 단 방향 데이터 전송시간을 CAN 2.0A 명세와 2.0B 명세에 대해 실험하였다.

MRTCS은 단 방향 데이터 전송방식을 사용하며 지연시간 비교를 위한 가정은 다음과 같다. sMCU를 사용하는 경우 DI/DO 동작들에 대한 인터럽트를 사용하기 때문에 인터럽트 개수를 2, 4, 8, 10개로 제한하고 이들의 인터럽트 처리 지연은 1 ms 이하로 가정하였다.

2. 모의실험 결과 분석

각 명세를 이용한 데이터 단 방향 전송 시간 결과를 그림 8에 나타내었다. 그림 8에서 보는 바와 같이 전송에 필요한 총 데이터 개수가 작은 CAN 2.0A 명세 기반의 데이터 전송 시간이 2.0B 명세를 이용하는 것보다 짧은 것을 알 수 있다.

그림 9, 10에서 CAN 2.0A 명세기반의 경우 그림 8과 같은 전송 속도를 보여준다. 인터럽트 2, 4개를 이용하는 경우 인터럽트 처리 시간이 8, 10개에 비하여 짧기 때문에 지연 시간의 변위 폭이 적은 것을 알 수 있다. sMCU 기반에서 4개 이하의 인터럽트를 사용하는 경우 최대 320 us를 넘지 않았다. 이는 가정에서 최대 인터럽트 지연시간을 1 ms 이하로 가정한 상태에서 인터럽

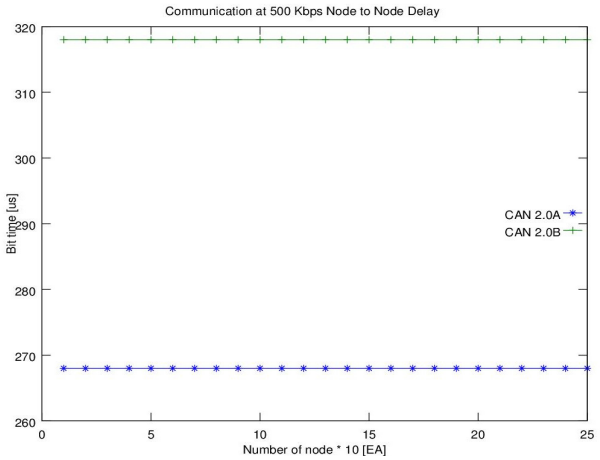

그림 8. 전송속도 500 Kbps 에 대한 제어노드와 노드 간 전송시간 비교
Fig. 8. Data Transmission time comparison between control node and node using 500 Kbps CAN speed.

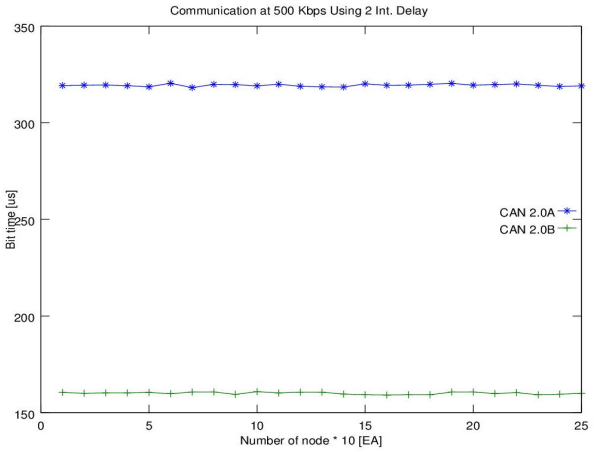

그림 9. 인터럽트 2개에 대한 단 방향 전송 시간비교.
Fig. 9. One-way Data transmission time comparison using 2 interrupts.

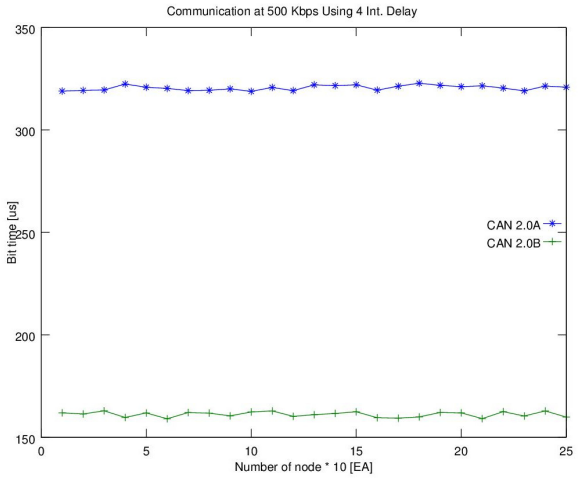

그림 10. 인터럽트 4개에 대한 단 방향 전송 시간비교.
Fig. 10. One-way Data transmission time comparison using 4 interrupts.

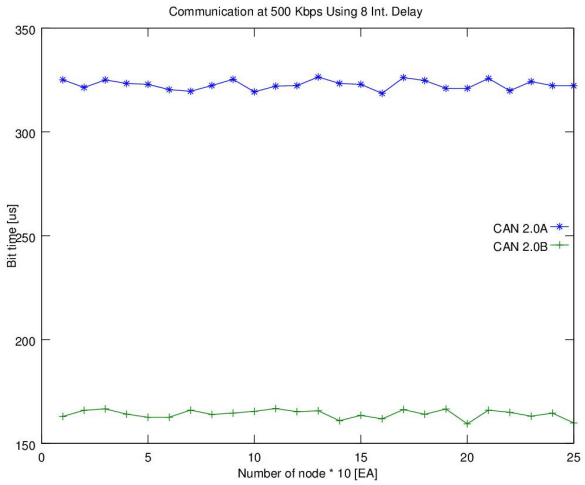

그림 11. 인터럽트 8개에 대한 단 방향 전송 시간비교.
Fig. 11. One-way Data transmission time comparison using 8 interrupts.

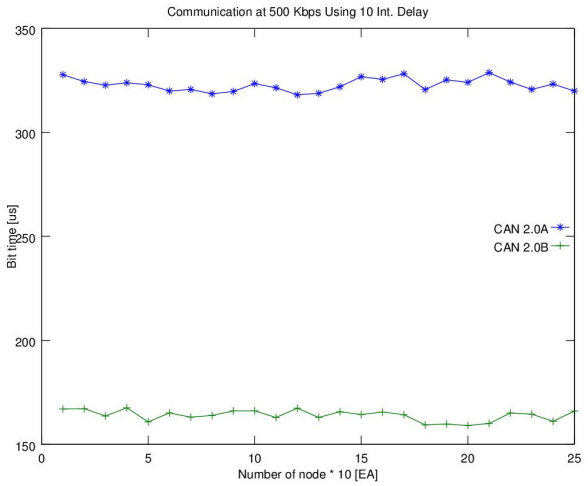

그림 12. 인터럽트 10개에 대한 단 방향 전송 시간비교.
Fig. 12. One-way Data transmission time comparison using 10 interrupts.

트 개수가 매우 작기 때문에 그림 7의 CAN 2.0A 명세에 근접함을 보여준다.

그림 11, 12에서 인터럽트 개수를 각각 8, 10개로 제한하고 분석하였다. 모의실험 결과를 통해 인터럽트 개수의 증가에 따라 CAN 2.0A 명세의 전송 시간 지연 폭이 증가함을 알 수 있다. 이 두 경우에는 320 us~330 us 까지 전송속도가 지연되는 것을 알 수 있다. 그림 8과 비교해 보면 CAN 통신에 적용한 명세가 CAN 2.0B 임에도 불구하고 CAN 2.0A 명세를 사용하는 것보다 데이터 전송 시간이 단축되는 것을 알 수 있다.

Ⅴ. 결론 및 향후 연구방향

본 논문에서 제안하는 MRTCS는 노드 구성을 최적화 하고 디바이스형태로 단순화시켜, 노드 내에서의 인터럽트 처리와 데이터 전송에 대한 지연을 감소시킬 수 있는 기능을 제공한다. MRTCS의 제어노드는 RTOS를 내장하고 4개의 독립된 태스크들로 운영하며 노드는 sMCU가 없는 CAN 디바이스로 구성하였다.

실시간 처리를 위해 노드들과의 CAN 통신을 이용하여 신뢰성을 제공하였다. 모의실험을 통해 MRTCS의 시스템 성능이 50 % 향상 되는 것을 알 수 있었다.

모의실험 결과를 분석해 보면 제안된 MRCTS 설계 기법은 다양한 실시간 제어 시스템에 적용 가능하다는 것을 알 수 있었다. 향후 연구에서는 다양한 산업용 시스템에 MRTCS을 적용하고 데이터 전송 지연과 내부 처리 지연에 대한 성능평가에 대한 연구를 진행하고자 한다.

REFERENCES

[1] YoungGyu-Lim , DongSung-Kim , “Design of Load Balanced DCS for Large Scaled Ship,” Proceedings of the Institute of Electronics Engineers of Korea Conference, pp. 1672-1675,

2012.
[2] Jiyong Park , Seongsoo Hong, “HEART: A Highly Customizable Real-Time Operating System for Diverse Embedded Systems,” Proceedings of the Institute of Electronics Engineers of Korea, pp. 323-324, 2006.
[3] Moonvin Song , Yunmo Chung, “A Study on the Hardware Architecture for Silicon RTOS,” J. IEEK-Semiconductor and Devices, vol.43, no. 11, pp.19-25, 2006.
[4] M. Abdelsalam Hassan, Keishi Saknushi, Yoshinori Takeuchi and Masaharu Imai, “A System ITRON Profile for RTOS Centric Co-Simulation; Performance Modeling in SystemC,” 대한전자공학회 논문지, vol. 1, pp. 181-182, 2005.
[5] Jae-Hyung Lee, Dong-Sung Kim, “Design and Implementation of Distributed Control System based on Dual Field-bus for Ship Engine,” J.IEEK-System and Conteol, vol.49, no.2, pp.1-9, 2012.
[6] Seung-Jun Lee, et.al., “Design and Implementation of Shipboard Monitering System”, Proceedings of the Korea Multimedia Society Conference, pp.59-63, 2008.
[7] C. U. Lee, et. al., “Development of embedded vessel monitoring system using NMEA 2000,”J. KSME, vol. 33, no. 5, pp. 746-755, 2009.
[8] Hyun Lee, et.al., “Marine Engine State Monitoring System using DPQ in CAN Network,” J.ICROS, vol. 18, Issue 1, pp. 13-20, 2012.
[9] Lee, Sang-Jae, Park, Gyei-Kark, Kim, Do-Yeon, “Study on applying Quad-Tree & R-Tree for building the analysis system using massive ship position data,” J.KIIS, vol. 21, Issue 6, pp.698-704, 2011.
[10] Presi, T. P., “Design and development Of PIC microcontroller based vehicle monitoring system using Controller Area Network (CAN) protocol,” IEEE International Conference on Information Communication and Embedded Systems, pp. 1070-1076, 2013.
[11] Mubeen, S., et al., “Response-time analysis of mixed messages in controller area network with priority-and FIFO-queued nodes,” IEEE International Workshop on Factory Communication Systems, pp. 23-32, 2012.
[12] Zhao Yu-zhuang, et al., “Design of Real-Time and Multi-task Control System for Semi-active Suspension Based on PICOS18,” IEEE International Conference on Embedded Software and Systems, pp. 565-571, May. 2009.
[13] Braescu F.C., Ferariu L., Lazar C., “OSEK-based multiple controllers with schedule feasibility self-testing,” IEEE International Symposium on Power Electronics Electrical Drives Automation and Motion, pp. 1237-1242, Jun. 2010.
[14] Huseinbegovic, S., Kreso, S., Tanovic, O., “Development of a Distributed elevator control system based on the microcontroller PIC 18F458,” IEEE International Conference on Computational Technologies in Electrical and Electronics Engineering, pp. 858-863, Jul. 2010.
[15] Xin Qiao, Wang Z., et al., “A CAN and OSEK NM Based Siren for Automobiles,” IEEE International Conference on Networking, Sensing and Control, pp. 868-873, Apr. 2007.
[16] Nisar Shahzada Khayyam, et al., “Operating System Performance Analyzer for Low-End. Embedded Systems,” IEEE International Journal of Computer Science, vol. 8, no. 6, 2011.
[17] Control Area Network 2.0A/2.0B Specification, http://www.bosch-semiconductors.de, 2013.
[18] OSEK/VDX, http://www.ms-osek.org, 2013.
[19] PICOS18, http://www.pragmatec.net, 2013.
[20] PIC18F4580, PIC250xx, http://www.microchip.com, 2013.

저 자 소 개

임 영 규(학생회원)-주저자
2009년 경북대학교 전기전자공학과 학사.
2002년 금오공과대학교 컴퓨터공학과 석사
2012년 금오공과대학교 IT융복합대학원 박사과정
2012년 前 대한상공회의소 충북인력개발원 정보통신과 교수
<주관심분야 : 실시간처리, 임베디드 시스템>

김 동 성(정회원)-교신저자
Jounal of IEEK vol. 49, no.2, pp.103-111, 2012 참조