논문 2022-17-23

멀티 코어 DSP를 위한 이더넷 기반 고속 데이터 통신 구현

(Implementation of Ethernet-Based High-Speed Data Communication for Multi-core DSP)

응 우 옌 후 동, 최 준 영*
(Dung Huy Nguyen, Joon-Young Choi)

Abstract : We propose a high speed data communication method for motor drive systems with fast control cycle in order to collect state variables of motor control without degrading control performance. Ethernet is chosen for communication device, and multi-core DSP architecture is exploited for communication processing load distribution. The communication program including network protocol stack and motor control program are assigned to two separate cores, and data between two cores are exchanged using interrupt-based inter-process communication mechanism, which enables to achieve a high-speed communication performance without degrading the motor control performance. The performance of developed communication method is demonstrated by real experiments using TCP, UDP and Raw Socket protocols in an experimental setup consisting of TI's TMS320F28388D motor control card and MS Windows PC.

Keywords : Ethernet, Multi-core DSP, TCP, UDP, Raw socket

Ⅰ. 서 론

DSP (Digital Signal Processor)는 자동제어기, 영상처리, 통신, 네트워크, 가전제품 등에 널리 사용되고 있으며 다양한 산업 분야에서 DSP는 복잡한 산업 제어 시스템을 설계하고 구현하기 위해 사용되고 있다 [1, 2]. 특히, 디지털 신호 처리 기술은 비용과 에너지 효율적인 제어 시스템 설계를 가능하게 하고 DSP의 우수한 성능은 더 저렴한 전기 모터, 적은 수의 센서, 작은 크기의 EMI 필터를 사용하여 디지털 모션 제어 응용 분야의 시스템 비용을 감소시키는 장점을 나타낸다. 따라서 DSP는 모터 제어 응용 분야의 모터 드라이브 시스템을 위한 효과적인 핵심 부품으로 알려져 있다 [3].

한편 고성능 모터 드라이브 시스템을 구현하기 위해 고속 제어 루프의 상태변수를 측정, 전송, 감시하는 정보 수집 및 처리를 위한 고속 통신 기능의 요구가 점점 증가하고 있다 [4]. 그러나 현재 널리 사용되고 있는 단일 코어 DSP는 고속 통신 기능과 복잡한 제어 알고리즘을 동시에 구현하는 데 한계를 나타낸다 [5].

최근에 멀티 코어 아키텍처는 마이크로프로세서의 동작 주파수의 과도한 증가 없이 성능을 현저하게 향상 시키는 중요한 수단이 되고 있다. 특히, 멀티 코어 DSP는 추가적인 DSP 코어 및 ARM 코어를 지원하고 있어 다양한 종류의 고성능 다기능 시스템 구현이 가능하며 단일 코어 DSP에 비해 많은 장점을 나타내고 있다 [6, 7].

본 논문에서는 이러한 멀티 코어 DSP의 장점을 활용하여 모터 제어 루프의 성능 저하 없이 시스템의 상태변수를 측정하고 고속 데이터 통신을 통하여 관리 PC로 전송하기 위한 이더넷 기반 고속 데이터 통신 방법을 제안하고 구현한다. 이더넷 통신 프로그램과 모터 제어 프로그램은 별도의 ARM 코어와 DSP 코어에 할당되며 IPC (Inter-Process Communication) 메커니즘을 사용하여 두 이종의 코어 간에 데이터를 교환한다 [8]. 관리 PC와 멀티 코어 DSP 보드로 구성되는 실험 환경을 구축하고 TCP, UDP, Raw Socket 프로토콜을 사용하는 실제 데이터 통신 실험을 수행하고 실험 결과를 분석 및 비교하여 구현된 이더넷 기반 통신 방법의 성능 및 안정성을 검증한다.

전체 논문의 구성은 다음과 같다. Ⅱ 장에서는 이더넷 장치를 이용하여 고속 데이터 전송이 가능하도록 멀티 코어 DSP기반 통신 방법을 설계한다. Ⅲ 장에서는 설계된 고속 데이터 통신 방법을 TCP, UDP, Raw Socket 프토코콜 기반으로 구현할 때 각각 필요한 절차 및 핵심 기술을 기술한다. Ⅳ 장에서는 멀티 코어 DSP 보드와 PC로 구성된 통신 실험 환경을 구축하고 구현된 통신 방법을 실험하고 실험결과를 분석한다. Ⅴ 장에서는 결론을 도출한다.

Ⅱ. 고속 데이터 통신 구조 설계

일반적으로 DSP는 디지털 데이터를 처리, 분석 및 해석하기 위해 수학적 조작을 통한 디지털 데이터 처리 기능을 갖춘 특수 마이크로프로세서이다. DSP는 주어진 데이터를

*Corresponding Author (jyc@pusan.ac.kr)
Received: Feb. 7, 2022, Revised: Mar. 6, 2022, Accepted: Mar. 15, 2022.
D.H. Nguyen: Pusan National University (M.S. Student)
J.Y. Choi: Pusan National University (Prof.)
※ 본 연구는 산업통상자원부 (MOTIE) 및 산업기술평가관리원 (KEIT) 연구비 지원에 의한 연구임 (No. 20012815).

ⓒ IEMEK J. Embed. Sys. Appl. 2022 Jun. 17(3) 185-190
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2022.17.3.185

실시간으로 처리하고 높고 정확한 데이터 처리 성능을 제공하며 소프트웨어를 사용하여 기능을 재 프로그래밍 할 수 있는 특징을 나타낸다 [9].

최근 시스템 제어 이론과 산업 통신 장치 및 프로토콜의 지속적인 발전으로 인하여 DSP 기반 시스템에서도 고속 통신 및 복잡한 제어 알고리즘을 동시에 처리할 수 있는 기능이 필요하게 되었다. 이러한 상황에서 기존 단일 코어 DSP는 성능의 한계로 이러한 요구 사항을 만족시킬 수 없었으며 이러한 문제를 극복하기 위해 멀티 코어 DSP가 개발되어 출시되었다. 멀티 코어 DSP는 더욱 높은 컴퓨팅 성능을 나타낼 뿐만 아니라 비용 및 전력 소비량 측면에서도 장점을 나타내고 있다 [10]. 이러한 멀티 코어 DSP를 기반으로 이더넷 장치를 이용하여 고속으로 데이터를 전송할 수 있는 통신 방법 및 구조를 다음과 같이 제안한다.

1. 통신 장치 및 프로토콜

본 논문에서는 멀티 코어 DSP 보드로서 TI 사의 TMS320F28388D MCU (Microcontroller unit)를 장착한 상용 F28388D 제어 카드를 채택하여 사용한다. TMS320F28388D MCU는 2개의 32-bit C28x DSP 코어를 포함하고 있어 각 코어에서 200MHz 속도로 신호 처리 기능을 제공한다. 또한 125MHz에서 동작하는 ARM Cortex-M4 프로세서를 기반으로 하는 독립적인 CM (Connectivity Manager) 코어도 포함하고 있다. CM 코어는 자체 전용 Flash 메모리 및 SRAM을 통해 동작하므로 F28388D 제어 카드의 외부 장치와의 데이터 통신 인터페이스를 완전히 독립적으로 제어할 수 있고 이는 CM 코어에서 외부 장치와의 데이터 전송을 완전히 독립적으로 처리할 수 있다는 것을 의미한다. 따라서 CM 코어가 외부 장치와의 통신 처리를 수행하고 있더라도 C28x DSP 코어는 실시간 제어에 집중할 수 있고 최대 대역폭의 성능을 달성한다.

이더넷은 효율적인 구현, 고속 통신, 유연한 토폴로지와 같은 다양한 장점으로 인해 산업 현장에서도 많이 채택되고 있는 통신 장치 및 프로토콜이다. 이러한 이더넷은 본 논문에서 제안하는 고속 데이터 통신 구현에 적합한 기술이라 판단하고 멀티 코어 DSP 기반으로 고속 데이터 통신을 구현하기 위하여 이더넷 기술을 채택한다.

2. 멀티 코어 기반 통신 구조

멀티 코어 DSP를 위한 이더넷 기반 통신 방법의 전체적 구조는 그림 1과 같다. 통신 방법 구조는 이더넷 통신을 처리하기 위한 CM 코어 부분과 모터 제어 프로그램 실행을 위한 C28x 코어 부분으로 구성된다. 이더넷 기반 통신 프로그램은 CM 코어에서 실행되고 모터 제어 프로그램은 C28x DSP 코어에서 실행된다. CM 코어와 C28x 코어 사이에서는 공유 메모리를 사용하여 데이터 전송이 수행되고 공유 메모리는 RxData 공간과 TxData 공간으로 분리되어 구현된다.

관리 PC와 C28x 제어 코어 사이의 데이터 전송 지연을 최소화하기 위해 다음과 같은 방법으로 데이터를 전송한다.

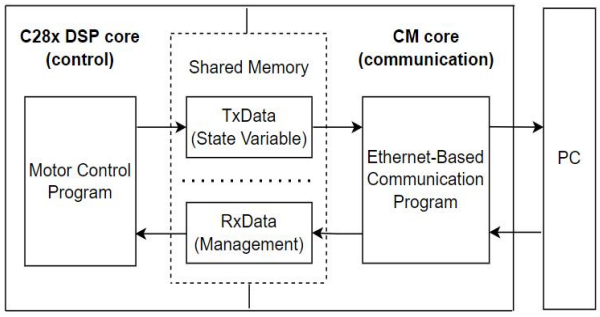

그림 1. 멀티 코어 기반 통신 구조
Fig. 1. Multi-core DSP-based communication structure

관리 PC는 모터 제어 관리용 데이터를 이더넷을 통해 CM 코어로 전송하면 CM 코어는 데이터를 수신하고 공유 메모리의 RxData 공간에 저장한다. 그러면 IPC 메커니즘에 의해 C28x 제어 코어에서는 인터럽트가 발생하고 ISR (Interrupt Service Routine) 에서 Rxdata 공간에 저장된 데이터를 읽어 필요한 처리를 수행한다. 한편 C28x 제어 코어는 측정한 상태변수 데이터를 공유 메모리의 TxData 공간에 저장하면 CM 코어에서 IPC 메커니즘에 의해 인터럽트가 발생하고 ISR에서 TxData 공간에 저장된 상태변수 데이터를 읽어 이더넷을 통해 관리 PC로 전송한다. 이렇게 IPC 메커니즘에서 제공하는 공유 메모리 인터럽트를 이용함으로써 두 코어 간 데이터 전송 지연을 최소화할 수 있다.

Ⅲ. 고속 데이터 통신 구현

II 장에서 설계된 멀티 코어 DSP를 위한 이더넷 기반 고속 데이터 통신 방법을 TCP, UDP, Raw Socket 프로토콜을 기반으로 구현한다.

1. C28x 제어 코어를 위한 통신 프로그램

C28x 제어 코어에서는 모터 제어 프로그램이 실행된다고 가정하고 CM 코어와 통신하기 위한 프로그램이 필요하고 핵심적인 실행 과정을 기술하면 다음과 같다.

첫 번째 전송 속도를 정밀하게 측정할 수 있는 CPU-Timer 설정이 필요하다. CPU-Timer는 시스템 클럭 기반으로 실행되는 카운트 다운 Timer이다. CPU-Timer를 시작할 때 설정된 시간 값으로 카운터 레지스터를 로드하고 이 후 측정하고자 순간의 경과 시간은 설정된 값과 현재 카운터 레지스터 값의 차이를 계산하면 얻을 수 있다.

두 번째 관리 PC로 전송할 데이터를 공유 메모리의 TxData 공간에 저장했을 때 CM 코어에서 인터럽트가 발생하도록 IPC 인터럽트를 설정하고 인터럽트 신호를 전송하는 동작이 필요하다. 또한 CM 코어가 RxData 공간에 데이터를 쓰면 C28x 제어 코어는 인터럽트 신호를 수신하여 인터럽트가 발생하는데 이 때 RxData 공간에서 데이터를 읽기 위한 ISR이 필요하고 이를 작성하여 ISR로 등록한다. 즉 ISR의 핵심 수행 내용은 RxData 공간에서 데이터를 읽고

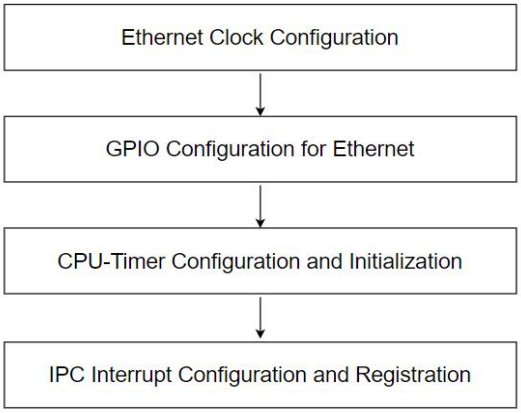

그림 2. 제어 코어 통신 프로그램 초기화 과정
Fig. 2. Control core communication program initialization process

CPU-Timer 이용하여 전송 속도 측정하고 상태변수를 TxData 공간에 저장하고 CM 코어에 인터럽트 트리거 신호를 전송하는 것이다.

이외에 C28x 제어 코어 구현을 위해 추가적 절차가 필요하나 일반적인 사항으로 자세히 기술하지는 않고 대신 그림 2에 전체 절차의 개요 및 순서를 나타낸다.

2. 이더넷 기반 통신 프로그램

CM 코어에서 TCP, UDP, Raw Socket 3개의 이더넷 기반 프로토콜을 사용하면서 실행되는 통신 프로그램을 구현한다. CM 코어 통신 프로그램은 전송 지연을 최소화하기 위해 TI 사의 SYS/BIOS 실시간 운영체제를 기반으로 작성한다. 이더넷을 통해 관리 PC와 C28x 제어 코어 간에 데이터를 교환하기 위하여 CM 코어에서 필요한 동작 과정은 다음과 같다.

첫 번째 이더넷 제어기에 대하여 SYS/BIOS 기반으로 초기화 및 설정이 필요하다. Raw Socket를 사용하는 경우 TI 사에서 제공하는 이더넷 모듈을 이용하여 프로그램을 작성한다. UDP를 사용하는 경우는 이더넷 모듈을 이용하여 데이터 연결 계층을 구축하고 lwIP 라이브러리를 이용하여 네트워크 계층과 전송 계층을 구축한다. 반면에 TCP를 사용하는 경우는 TI 사의 NDK (Network Developer's Kit) 기반으로 프로그램을 작성한다 [11].

두번째 이더넷 프레임을 수신 할 때 인터럽트가 발생하도록 하드웨어 초기화 파일에 있는 인터럽트 번호를 사용하여 인터럽트를 설정하고 인터럽트 발생 시 필요한 수행 동작을 ISR로 등록한다. ISR의 핵심 내용은 관리 PC에서 수신된 데이터를 IPC 인터럽트를 통해 C28x 제어 코어로 전송하는 것이다.

세 번째 C28x 제어 코어에서 관리 PC로 상태변수 데이터를 전송하기 위하여 IPC 인터럽트 설정해야 하고 인터럽트 발생 시 수행 동작을 ISR로 등록한다. ISR의 핵심 내용은 C28x 제어 코어에서 수신된 데이터를 TCP, UDP, Raw Socket 프로토콜을 이용해 관리 PC로 전송하는 것이다.

3. PC 통신 프로그램

관리 PC에서는 Qt 프레임워크를 사용하여 CM 코어와 데이터를 송수신하는 프로그램을 구현한다. TCP 프로토콜을 사용하는 경우에는 Qt 프레임워크에서 제공하는 QTcpSocket Class를 이용하여 프로그램을 작성하고 UDP 프로토콜을 사용하는 경우에는 QUdpSocket Class를 이용하여 프로그램을 작성한다. CM 코어로 부터 데이터를 수신할

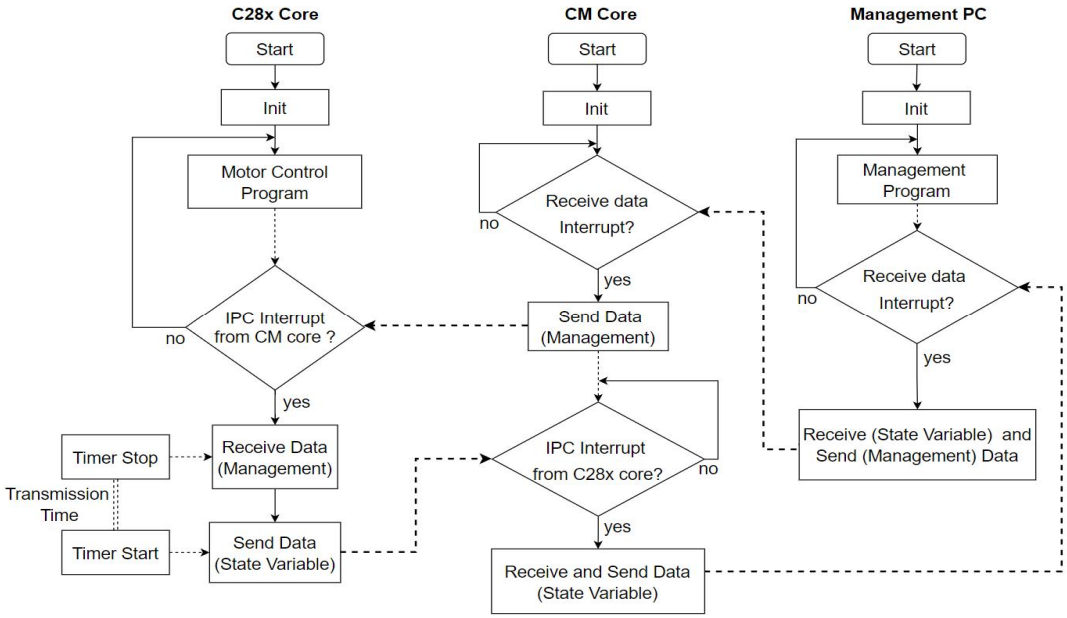

그림 3. 통신 방법 구조 및 데이터 흐름
Fig. 3. Communication method structure and data flow

때 즉시 CM 코어로 다시 전송할 수 있도록 데이터를 수신할 때 발생하는 인터럽트 설정하고 등록 한다. 특히, Raw Socket 프로토콜의 경우는 Qt 프레임워크에서 지원하지 않기 때문에 Npcap을 사용해서 구현한다. Npcap은 Nmap 프로젝트의 Windows 운영체제를 위한 패킷 캡처 및 전송 라이브러리로서 이를 통해 Windows 프로그램은 단순한 API를 사용하여 원시 네트워크 트래픽을 캡처하고 전송할 수 있다.

또한 비 실시간 운영체제인 Windows에서 데이터 전송 지연을 최소화하기 위해 Windows에서 이더넷 프레임 수신 시 인터럽트 발생 빈도를 최대로 유지하는 것이 필요하다. 이를 위해 Windows의 Interrupt Moderation 기능을 비활성화하는 것이 필요하고 이는 Windows 네트워크 드라이버에서 설정이 가능하다. Interrupt Moderation이 활성화 된 경우에는 이더넷 프레임을 수신해도 즉각적인 인터럽트가 발생되지 않으므로 한 주기의 데이터 전송 시간이 증가할 가능성이 있다.

C28x 제어 코어, CM 코어, 관리 PC로 구성되는 전체 통신 방법 구조와 전체 데이터 흐름 과정은 그림 3에 도시한다.

Ⅳ. 실 험

1. 실험 환경 구축 및 방법

제안된 통신 방법의 안정성 및 성능을 검증하기 위해 PC와 멀티 코어 DSP 보드로 구성되는 실험 환경을 구축하고 데이터 전송 실험을 수행하며 실험 결과를 분석한다. 실험에서 멀티 코어 DSP 보드는 TI 사의 F28388D 제어 카드를 채택하고 PC에서는 Windows 기반 Qt 프로그램을 작성하여 사용한다. 그림 4는 F28388D 제어 카드의 실제 사진을 나타내고 TMS320F28388D MCU와 이더넷 포트가 장착된 것을 확인할 수 있다.

실험에서 1개의 C28x DSP 코어는 제어 코어로 사용되고 CM 코어는 통신 코어로 사용된다. F28388D 제어 카드는 PC와 이더넷 케이블로 연결되며 이더넷 기반으로 TCP, UDP, Raw Socket의 프로토콜을 사용하여 실험을 수행 한다.

전송 속도 성능 측정 방법은 그림 3의 데이터 흐름 구조를 활용하여 다음과 같이 설명할 수 있다. 처음에 PC는 CM 코어로 시작 신호를 보내고 CM 코어는 시작 신호를 수신한 다음 IPC 인터럽트를 통해 C28x 제어 코어로 전송한다. C28x 제어 코어는 ISR를 통해 시작 신호를 수신하고 바로 IPC를 통해 상태변수 데이터를 CM 코어로 전송하면서 상태변수 데이터 전송의 처음 주기가 시작된다. CM 코어는 IPC 인터럽트를 통해 데이터를 수신한 후 다시 PC로 전송하고 PC는 데이터를 수신하고 즉시 CM 코어로 다시 제어 관리 데이터를 전송한다. CM 코어가 데이터를 수신하고 IPC 인터럽트를 통해 C28x 제어 코어로 제어 관리 데이터를 전송한다. C28x 제어 코어는 ISR를 통해 제어 관리 데이터를 수신하고 한 주기가 완료된다.

매 주기 소요 시간을 정밀하게 측정하기 위해 C28x 제어 코어에서 CPU-Timer를 사용하며 C28x 제어 코어에서 상

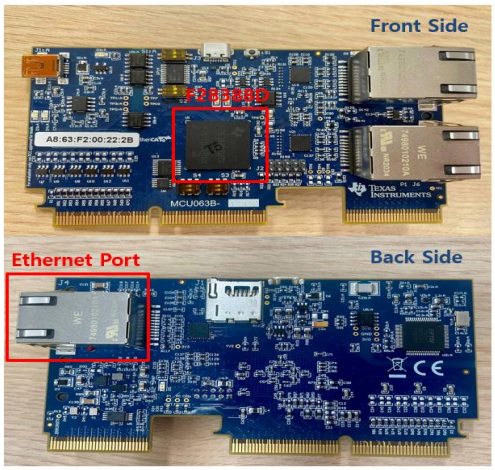

그림 4. TI 사의 TMS320F28388D 제어 카드
Fig. 4. TI's TMS320D28388D control card

태변수 데이터를 송신하기 직전 CPU-Timer를 시작하고 설정된 시간 값으로 카운터 레지스터를 로드한다. C28x 제어 코어에서 상태변수 데이터 송신 후 데이터는 CM 코어를 거쳐 PC에 도착하고, 이후 PC는 제어 관리 데이터를 CM 코어를 통해 C28x 제어 코어로 전송하고 C28x 제어 코어에서는 제어 관리 데이터 수신 후 ISR에 의해 CPU-Timer의 현재 카운터 레지스터 값을 읽어 저장한다. 상태변수 데이터 전송 직전 설정된 시간 값과 제어 관리 데이터 수신 시 저장한 카운터 레지스터 값의 차이를 계산하면 한 주기 동안 소요된 전송 시간을 정확하게 계산할 수 있다. 이러한 전송 과정을 연속적으로 1000번 반복하고 매 주기 시간을 측정 및 저장하고 평균값을 계산한다.

또한 전송되는 데이터 패킷의 크기에 따라 전송 시간이 달라지는데 그 양상을 측정하기 위해 1 Byte, 100 Bytes, 500 Bytes, 1000 Bytes의 4가지 데이터 패킷 크기로 실험을 수행한다.

2. 실험 결과

TCP, UDP, Raw Socket 프로토콜을 사용하여 연속적인 1000번의 전송을 수행했을 때 전송 주기 시간의 평균 값은 표 1과 같이 나타난다. Raw Socket 프로토콜을 사용할 때 1,

표 1. 프로토콜과 패킷 크기에 따른 전송 시간 결과
Table 1. Transmission time results according to protocol and packet size

	Protocol	Packet Size (Bytes)			
		1	100	500	1000
Transmission Time Average (μs)	Raw Socket	106	117	196	298
	UDP	306	376	583	825
	TCP	688	720	920	1127

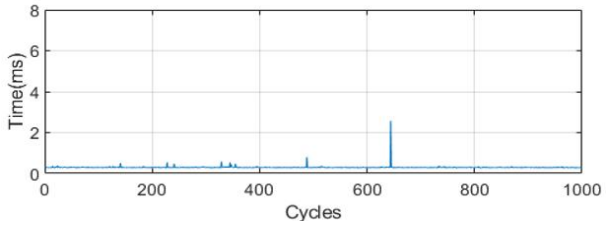

그림 5. 샘플 별 전송 시간 (Raw Socket, 1000 Bytes)
Fig. 5. Transmission time per sample (Raw Socket, 1000 Bytes)

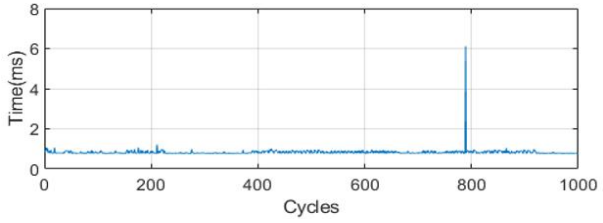

그림 6. 샘플 별 전송 시간 (UDP, 1000 Bytes)
Fig. 6. Transmission time per sample (UDP, 1000 Bytes)

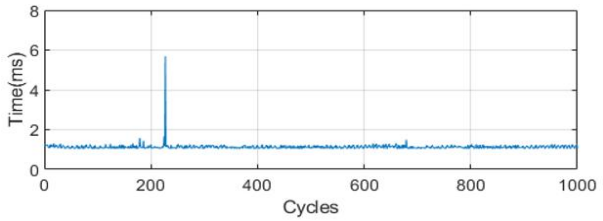

그림 7. 샘플 별 전송 시간 (TCP, 1000 Bytes)
Fig. 7. Transmission time per sample (TCP, 1000 Bytes)

100, 500, 1000 Bytes의 경우 각각 106 μs, 117 μs, 196 μs, 298 μs의 평균 데이터 패킷 전송 속도가 측정되고, UDP의 경우 각각 306 μs, 376 μs, 583 μs, 825 μs로 측정되며 TCP의 경우 각각 688 μs, 720 μs, 920 μs, 1127 μs로 측정된다.

이러한 결과로부터 구현된 통신 방법의 정상적인 동작을 검증할 수 있고 성능의 측면에서 Raw Socket, UDP, TCP 프로토콜의 순서로 우수한 것을 확인할 수 있다. 이러한 성능 차이에 대한 원인을 분석하면 다음과 같다. TCP/IP 프로토콜 계층 구조에서 TCP, UDP는 전송 계층 프로토콜이고 Raw Socket은 데이터링크 계층 프로토콜이므로 TCP, UDP는 네트워크 및 전송 계층 처리 과정이 추가되므로 Raw Socket보다 전송 시간이 더 소요된 것이다. 또한 TCP는 연결형 프로토콜이고 UDP는 비연결형 프로토콜이므로 TCP는 UDP에 비해 복잡한 처리 과정을 수행해야 하므로 TCP가 UDP보다 더 긴 전송시간을 나타낸다.

한편 데이터 패킷을 매번 전송할 때 전송 시간의 변화를 분석하기 위해 1000 Bytes 전송 실험 결과에 대해 1000번의 전송에 대한 전송 시간 결과를 그림 5, 6, 7에 나타낸다. 그림에서 알 수 있듯이 1000개의 샘플 중에 1개의 샘플이 비정상적으로 큰 값으로 측정되고 있는데 이 결과의 원인은 PC 프로그램이 실시간 운영체제가 아닌 Windows 기반으로 동작하고 있기 때문인 것으로 예상되고 이 문제를 해결하기 위해 후속 연구를 진행할 계획이다.

V. 결 론

본 논문에서는 모터 드라이브 시스템에서 모터 제어 루프의 성능 저하 없이 시스템의 상태변수를 측정하고 측정 데이터를 고속으로 전송하기 위해 멀티 코어 DSP 구조를 활용하는 이더넷 기반 데이터 통신 방법을 제안하였다. 이더넷 통신 프로그램과 모터 제어 프로그램은 개별적으로 할당된 CM 코어 및 DSP 코어에서 실행되며 두 코어 간에 데이터 교환은 IPC 및 인터럽트를 사용하여 이루어 진다. DSP 코어 프로그램은 펌웨어 형태로, CM 코어 프로그램은 TI 사의 실시간 운영체제 SYS/BIOS 기반으로, PC 프로그램은 Windows 기반 Qt 프레임워크로 작성하였다. TCP, UDP, Raw Socket의 3가지 프로토콜에 대하여 이더넷 기반 전송 방법을 구현하였으며 다양한 패킷 크기에 대하여 전송 실험을 수행하였고, 실험 결과로부터 제안된 통신 방법의 정상적 동작과 성능을 검증하였다.

한편 전송 시간이 급격히 증가하는 경우가 매우 드물게 발생하는 데 이 문제의 원인 분석과 해결 방법 개발을 향후 후속 연구 내용으로 수행할 계획이다.

References

[1] Z. Q. Zhang, Y. L. Zhang, “Realization of Communication Between DSP and PC Based on Modbus Protocol,” Proceeding of 2009 International Conference on Multimedia Information Networking and Security, Vol. 2, pp. 258-261, 2009.

[2] Z. Fang, Y. Fu, “A Networked Embedded Real-time Controller for Complex Control Systems,” Proceeding of 2011 Chinese Control and Decision Conference (CCDC), pp. 3210-3215, 2011.

[3] S. Bejerke, “Digital Signal Processing Solutions for Motor Control Using the TMS320F240 DSP-Controller,” Proceeding of the First European DSP Education and Research Conference, 1996.

[4] T. Matsumaru, S. Kawabata, T. Kotoku, N. Matsuhira, K. Komoriya, K. Tanie, K. Takase, “Task-based Data Exchange for Remote Operation System Through a Communication Network,” Proceeding of 1999 IEEE International Conference on Robotics and Automation, Vol. 1, pp. 557-564, 1999.

[5] S. M. Park, H. W. Kim, H. M. Cho, J. Y. Choi, “Development of EtherCAT Slave Based on Multi-Core DSP,” Proceeding of 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 157-161, 2018.

[6] Y. K. Song, Q. S. Qian, D. L. Zhang, “Design and

Implementation of Dual-port Network on Chip Based on Multi-core System," Proceeding of 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), pp. 1401-1403, 2016.

[7] S. M. Chen, J. H. Wan, J. Z. Lu, Z. Liu, H. Y. Sun, Y. J. Sun, H. Z. Liu, X. Y. Liu, Z. T. Li, Y. Xu, X. W. Chen, "YHFT-QDSP: High-performance Heterogeneous Multi-core DSP," J. Comput. Sci. Technol, Vol. 25, No. 2, pp. 214-224, 2010.

[8] "TMS320F2838x Real-Time Microcontrollers With Connectivity Manager," Technical Reference Manual, Texas Instruments, 2021.

[9] https://www.prnewswire.com/news-releases/the-worldwide-digital-signal-processors-industry-is-expected-to-reach-19-billion-by-2026-301415240.html

[10] L. Karam, I. Alkamal, A. Gatherer, G. A. Frantz, D. V. Anderson, B. L. Evans, "Trends in Multicore DSP Platforms," IEEE Signal Processing Magazine, Vol. 26, No. 6, pp. 38-49, 2009.

[11] "TI Network Developer's Kit (NDK)," User's Guide, Texas Instruments, 2017.

Dung Huy Nguyen (응우옌 후 동)

2017 Automation and Control Engineering from Hanoi University of Science and Technology, Hanoi, Viet Nam (B.S.)

2020~Electronics Engineering from Pusan National University (M.S. candidate)

Field of Interesst: Embedded system and control system

Email: leegaram2019@pusan.ac.kr

Joon-Young Choi (최 준 영)

1994 Electronics and Electric Engineering from Pohang University of Science and Technology (B.S.)

1996 Electronics and Electric Engineering from Pohang University of Science and Technology (M.S.)

2002 Electronics and Electric Engineering from Pohang University of Science and Technology (Ph.D.)

2005~Department of Electronics Engineering at Pusan National University, Busan, Korea (Prof.)

Field of Interesst: Embedded system and control system

Email: jyc@pusan.ac.kr