반도체디스플레이기술학회지 제20권 제2호(2021년 6월)
Journal of the Semiconductor & Display Technology, Vol. 20, No. 2. June 2021.

실시간 EtherCAT 마스터 구현에 관한 연구

강성진[*†]
[*†]한국기술교육대학교 전기전자통신공학부

A Study on Implementation of Real-time EtherCAT Master

Sung Jin Kang[*†]
[*†]School of Electrical, Electronics & Communication Engineering,
Korea University of Technology and Education

ABSTRACT

EtherCAT is an Ethernet-based fieldbus system standardized in IEC 61158 and SEMI, and widely used in the fields of factory automation, semiconductor equipment and robotics. In this paper, a real-time EtherCAT master is implemented on Linux operating systems and its performances are evaluated. To enhance the real-time capability of mainline Linux kernel, Xenomai is applied as a real-time framework and an open source EtherCAT master stack, Simple Open EtherCAT Master (SOEM), is installed on it. Unlike other studies, the real-time performance of the EtherCAT master is evaluated at the output of the network interface card, so that the evaluation results include all possible effects from the EtherCAT master system. The implemented EtherCAT master can send and receive packets up to 20KHz control frequency with low jitter, even in stressed condition.

Key Words : EtherCAT, Linux, Real-time, Xenomai, SOEM

1. 서 론

산업용 컴퓨터 네트워크 프로토콜인 필드버스(fieldbus)는 공장 자동화에 반드시 필요한 통신 기술이며, 특히 4차 산업혁명 시대를 대표하는 스마트 팩토리는 공장 안의 모든 요소가 유기적으로 연결되고 지능적으로 운영되어야 하므로 고속의 필드버스에 대한 수요가 증가되고 있다.

EtherCAT은 Ethernet기반의 필드버스 시스템으로2007년에 국제 표준인 IEC 61158과 국제 반도체 장비 재료 협회인 SEMI의 표준으로 채택되었고, 원활하고 정밀한 실시간 모터 제어와 센서 데이터 수집이 가능한 유연한 네트워크 토폴로지를 가지면서 통신 속도가 빠르고 저비용으로 설치 및 유지가 가능하다는 장점을 가지고 있어, 최근 공장 자동화, 반도체 장비, 로봇 분야에서도 활발하게 적용되고 있다[1].

EtherCAT은 정주기와 실시간 제어를 위해서 RTOS (Real-Time Operating System) 상에서 동작해야 한다. 본 논문에서는 라이선스 비용이 발생하는 상용 RTOS대신 Linux 운영체제 상에서 실시간 EtherCAT 마스터 시스템을 구현한다. 일반적인 Linux 커널은 시스템의 전체 처리 능력을 최대화하도록 설계되어 있기 때문에, 실시간 처리가 요구되는 환경에 적합하지 않다. 따라서, Linux용 실시간 프레임 워크인 Xenomai를 설치하여 사용하거나, PREEMPT_RT 패치를 적용하여 사용해야한다[2]. PREEMPT_RT 패치를 사용하는 경우보다 Xenomai를 사용하는 경우가 실시간 반응성과 정주기 제어 성능이 우수하므로[2,3], 본 논문에서는 Linux 커널에 Xenomai를 적용하여 사용한다.

대표적인 공개 소스 EtherCAT 마스터 스택은 IgH EtherCAT Master for Linux (Etherlab)와 Simple Open EtherCAT Master (SOEM)가 있다. Etherlab과 SOEM은 모두 EtherCAT 마

†E-mail: sjkang@koreatech.ac.kr

스터에 필요한 대부분 기능을 제공하고 있지만, Etherlab은 실시간 Linux 커널 기반으로 동작하는 반면에 SOEM은 사용자 영역에서 동작하는 라이브러리이므로Linux 외의 다른 운영체제에서도 동작하는 특징이 있다[5,6]. 본 논문에서는 SOEM을 EtherCAT 마스터 스택으로 사용한다.

실시간 EtherCAT 마스터의 정주기 제어 성능은 사용하는 플랫폼과 실시간 Linux 커널의 성능에 직접적인 영향을 받지만 네트워크 드라이버와NIC (Network Interface Card)의 성능에도 큰 영향을 받는다[7]. 그러나, 지금까지 대부분의 연구는 응용 프로그램의 제어 루프에서 정주기 성능을 측정하거나, 실시간 Linux 커널의 실시간 반응 성능을 측정하고 있다[2,3,7]. 본 논문에서는 제어 루프의 주기성과 NIC에서 출력되는 패킷의 주기성을 측정함으로써, 구현된 실시간 EtherCAT 마스터 시스템에서 발생하는 모든 영향이 반영된 실제 정주기의 성능을 평가한다.

2. 리눅스 기반 실시간 EtherCAT 마스터

본 논문에서 사용된 플랫폼 사양은 Table 1과 같다. 인텔 H410 칩셋을 사용하는 메인보드에 동작 주파수가 2.9GHz인 인텔 코어 i5 CPU를 사용하였다. 네트워크 컨트롤러는 RTL8111H가 메인보드에 탑재지만, EtherCAT 통신용으로 WG82574L 컨트롤러를 사용하는 인텔 기가비트 NIC를 PCI-e 슬롯에 추가하였다.

실시간 EtherCAT 마스터 구현에 사용한 소프트웨어 툴 버전은 Table 2와 같다. 우분투 LTS 20.04.2 버전은 Linux 커널 5.8.0버전을 사용하지만, Xenomai IPIPE patch가 현재 Linux 커널 5.4까지 지원하므로[8], Linux 커널 5.4.102을 사용한다. Xenomai는 안정화된 최신 버전 3.1을 사용하고[4], SOEM은 안정화된 최신 버전 1.4.0 버전을 사용한다[6].

Table 1. Platform specifications

Main board	ASUS Prime H410I-PLUS
CPU	Intel Core i5-10400 @2.90GHz
Network controller (on board)	Realtek RTL8111H
Network controller (PCI-e Slot)	Intel GIGABIT adapter, WG82574L controller
Memory	DDR4 8GB
Form factor	mini-ITX

Table 2. Version of tools

Ubuntu	LTS 20.04.2
Linux Kernel	5.4.102
IPIPE patch	ipipe-core-5.4.102-x86-3.patch
Xenomai	3.1 (stable)
SOEM	1.4.0 (stable)

2.1 Xenomai 기반 실시간 Linux 커널

Xenomai 기반 실시간 Linux 커널을 구성하기 위해서는 먼저 ADEOS I-PIPE 패치를 Linux 커널에 적용해야 하며, 이 패치는 [8]에서 다운받을 수 있다. 본 논문에서는 Table 2와 같이 Linux 5.4.102 커널에 ipipe-core-5.4.102-x86-3.patch를 적용하였다.

Linux 커널이 실시간으로 동작하기 위해서는 Linux커널의 응답 속도를 저해하는 커널 옵션은 비활성화하고, Linux커널의 응답 속도를 빠르게 하는 커널 옵션은 활성

Table 3. Linux kernel configurations

General setup
• Timers subsystem: High Resolution Timer Support (*Enable)* • Preemption Model: *Preemption Kernel (Low-Latency Desktop)*
Processor type and features
• Linux guest support *(Disable)* • Processor family: *Core 2/newer Xeon* • Multi-core scheduler support *(Enable)*: CPU core priorities scheduler support *(Disable)* • Timer frequency: *1000 HZ*
Power management and ACPI options
• CPU Frequency scaling *(Disable)* • ACPI Support: Processor *(Disable)* • CPU Idle: *CPU idle PM support (Disable)*
Memory Management options
• Transparent Hugepage Support *(Disable)* • Allow for memory compaction *(Disable)* • Contiguous Memory Allocator *(Disable)* • Allow for memory compaction: Page Migration *(Disable)*
Xenomai/cobalt
• Sizes and static limits Number of registry slots *(set to 4096)* Size of system heap (Kb) *(set to 4096)* Size of private heap (Kb) *(set to 256)* Size of shared heap (Kb) (set to 256) Maximum number of POSIX timers per process *(set to 512)* • Drivers ➔ RTnet RTnet, TCP/IP socket interface *(Enable)* Protocol Stack ➔ TCP support *(Enable)* Drivers ➔ Intel(R) 82575 (Gigabit) *(Enable)*
Device Drivers
• Staging drivers Unisys SPAR driver support *(Disable)* • Microsoft Hyper-V guest support Microsoft Hyper-V client drivers *(Disable)*
Kernel hacking
• KGDB: kernel debugger *(Disable)*

화한 후에 커널을 빌드해서 사용해야 한다[9]. 본 논문에서는 Table 3과 같이 커널 옵션을 선택하여 커널을 빌드하고 설치했다.

2.2 Xenomai 사용자 라이브러리

Xenomai 기반의 실시간 Linux 커널 상에서 사용자 영역의 실시간 태스크(real-time tasks)를 개발하기 위해서는 Xenomai 사용자 영역 라이브러리를 설치해야 하며, 본 논문에서는 Table 2와 같이 안정적인 최신 버전 Xenomai 3.1을 설치한다. Xenomai 라이브러리 빌드를 위해 사용한 configure 옵션은 다음과 같다[4].

```
./configure --with-core=cobalt --enable-smp --disable-tls --
enable-dlopen-libs
```

Xenomai 라이브러리를 빌드한 후에 라이브러리 설치 및 환경 설정을 마치면 Xenomai 기반 실시간 Linux 커널 상에서 사용자 영역의 실시간 태스크를 개발할 수 있다.

2.3 EtherCAT 마스터 스택

공개 소스 EtherCAT 마스터 스택인SOEM은 사용자 영역에서 동작하는 응용 프로그램이며, 소스 코드는 [10]에서 다운로드 할 수 있다. SOEM은 CMake를 사용하여 빌드하기 때문에, 2.2절의 Xenomai 사용자 라이브러리를 포함하여 빌드하도록 CMakeLists.txt를 수정해야한다.

Xenomai 라이브러리를 사용할 수 있도록 cobalt와 modechk 라이브러리를 추가해 주어야 하고, include 디렉토리와 link 디렉토리를 설정해 주어야한다. 또한, 빌드와 링크를 위한 옵션도 추가로 설정해 주어야 한다. 다음은 CMakeLists.txt중에서 Linux와 관련된 부분을 수정한 내용이다.

```
set(OS "linux")
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall -
   Wextra -Werror")
set(OS_LIBS cobalt modechk pthread rt)
execute_process(COMMAND xeno-config --posix --cflags
   OUTPUT_VARIABLE XENO_CFLAGS
   OUTPUT_STRIP_TRAILING_WHITESPACE)
execute_process(COMMAND xeno-config --posix --ldflags
     OUTPUT_VARIABLE XENO_LDFLAGS
     OUTPUT_STRIP_TRAILING_WHITESPACE)
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS}
     ${XENO_CFLAGS} ${XENO_LDFLAGS}")
set(CMAKE_C_COMPILER gcc)
include_directories(
/usr/xenomai/include
/usr/xenomai/include/cobalt
)
link_directories( /usr/xenomai/lib )
```

수정된 CMakeLists.txt를 이용하여 SOEM을 빌드하면 라이브러리, 헤더, 실행 파일 등이 생기고, 적절한 디렉토리로 옮기고 환경 설정을 해주면 EtherCAT 마스터 스택을 사용할 수 있다.

2.4 RTnet 설정

Linux 커널의 네트워크 드라이버는 실시간 동작을 보장하지 않기 때문에, Xenomai 기반 실시간 Linux 커널 상에서 실시간 태스크를 생성해서 패킷을 전송해도 NIC에서 송신되는 패킷은 실시간성과 정주기성이 보장되지 않는다. 또한, Ethernet 프로토콜은 일정한 주기의 통신이 허락되지 않는다. RTnet은 이더넷 계층과 애플리케이션 계층(또는 IP 계층) 사이에서 동작하는 프로토콜 스택으로, TDMA방식의 시간 슬롯을 사용하여 일정한 주기의 통신(deterministic communication)이 가능하도록 하며 패킷 충돌을 감지하는 CSMA/CD를 비활성화하고 패킷을 버퍼링하지 않도록 한다[11].

Xenomai 3.1은 Linux 커널 옵션에서 RTnet을 선택할 수 있게 지원하고 있고, Linux 커널 빌드를 마치면 실시간 네트워크 드라이버와 관리 툴이 설치된다. RTnet은 다양한 네트워크 컨트롤러를 지원하지는 않는다. 본 논문에서 사용하는 플랫폼의 메인보드에 탑재된 RTL8111H는 RTnet 네트워크 드라이버가 지원되지 않으므로, 82574L 컨트롤러를 사용하는 인텔 기가비트 NIC를 PCI-e 슬롯에 추가하여 실시간 EtherCAT 통신용으로 사용한다.

RTnet을 사용하기 위해서는 Table 3과 같이 Linux 커널 옵션에서 RTnet을 선택하고 빌드한 후에, 설정 파일 /usr/xenomai/etc/rtnet.conf을 아래와 같이 수정해야 한다.

```
RT_DRIVER="rt_e1000e"
RT_DRIVER_OPTIONS=""
REBIND_RT_NICS="0000:01:00.0"
IPADDR="192.168.0.21"
NETMASK="255.255.255.0"
RT_LOOPBACK="yes"
RT_PROTOCOLS="udp packet tcp"
RTCAP="no"
TDMA_MODE="master"
```

여기에서 rt_e1000e는 실시간 네트워크 드라이버이고, e1000e는 일반적인 네트워크 드라이버이다. REBIND_RT_NICS="0000:01:00.0"는 NIC의 PCI 버스 정보를 정확히 입력해 주어야 하며, lshw 명령어로 확인할 수 있다[12].

3. 실험 및 성능 평가

3.1 EtherCAT 마스터의 제어 주파수

MTU (Maximum Transmission Unit)가 1,500byte인경우, Ethernet 프레임의 길이는 Ethernet header 14byte, FCS (Frame Check Sequence) 4byte를 포함하여 최대 1,518Byte이고, 프리엠블 8byte와 최소 IPG (Inter-packeet Gap) 12byte를 포함하는 Ethernet 패킷의 길이는 최대 1,538Byte이다. EtherCAT 데이터는 EtherCAT 헤더 2byte, Datagram 헤더 10byte, WKC(working counter) 2byte를 가지므로 EtherCAT 데이터는 최대 1,486byte가 될 수 있다. EtherCAT slave가 FMMU (Fieldbus Memory Management Unit)를 사용한다고 가정하면, EtherCAT 마스터의 이론적인 최대 제어 주파수(EtherCAT 패킷 전송 주파수)는 다음과 같다.

$$f_{max} = f_{Eth} / (52 + \mathrm{L}) \quad (1)$$

여기에서 L은 EtherCAT 데이터의 길이(byte)이고, f_{Eth}는 Ethernet의 속도이다. 만약 FMMU를 사용하는 EtherCAT slave가 50개이고, 각 slave마다 제어 데이터가 25 bytes이면, 100Base-T Ethernet NIC를 사용하는 EtherCAT 마스터의 이론적인 최대 제어 주파수는 $12.5 \times 10^6/(52 + 50 \times 25) \approx 9{,}600$ Hz이다. 실제 실시간 EtherCAT 마스터에서 전송할 수 있는 최대 제어 주파수는 이론적 최대 주파수 f_{max}의 10%~20%정도로 구현 가능하다.

3.2 실시간 EtherCAT 마스터 성능 평가

본 논문에서 구현한 실시간 EtherCAT 마스터의 정주기 및 실시간 성능을 평가하기 위해 Fig. 1과 같이 장치를 구성하였고, EtherCAT 마스터에서 소프트웨어적으로 주기를 측정하면서 동시에 NIC의 출력 신호를 netAnalyzer[13]에서 실시간으로 주기를 측정할 수 있다.

EtherCAT Slave는 Infineon XMC4800 EtherCAT Kit를 사용하였으며, Output size는 8bits, Input size는 2bits, 4개의 Sync manager, 2개의 FMMU를 가지고 있다[14]. XMC4800 EtherCAT Kit의 Output 8bits는 8개의 LED에 각각 연결되어 있어서, 본 논문에서는 EtherCAT 마스터가 패킷을 전송할 때 마다 LED를 토글(toggle)하도록 응용 프로그램을 작성하였다. 따라서, LED on/off 제어 신호의 주파수는 패킷 전송 주파수의 1/2이다. 예를 들어, EtherCAT 패킷을 8KHz로 전송하면, LED on/off 제어 신호는 4KHz의 주파수를 갖는 사각 펄스(rectangular pulse) 신호가 된다.

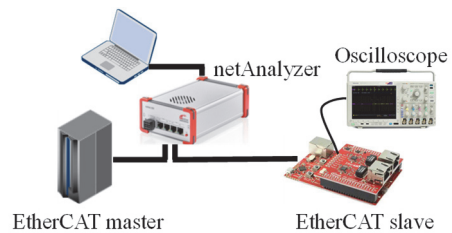

Fig. 1. Experimental setup.

3.2.1 EtherCAT 마스터 응용 프로그램

EtherCAT 마스터 응용 프로그램은 Xenomai의 POSIX 스킨을 사용하지만, 정주기 루프 제어는 Xenomai의Alchemy 라이브러리를 이용하기 때문에 다음과 같이 Makefile을 작성하여 빌드한다.

```
XENO_SKIN = posix
SOEM_INC_DIR = /usr/local/include/soem
SOEM_LIB_DIR = /usr/local/lib
SOEM_LIBS = -lsoem
ALCHEMY_INC_DIR = /usr/xenomai/include/alchemy
ALCHEMY_LIBS = -lalchemy -lcopperplate
CFLAGS = -I$(SOEM_INC_DIR) -I$(ALCHEMY_INC_DIR)
$(shell xeno-config --skin=$(XENO_SKIN) --cflags)
LDFLAGS = -L$(SOEM_LIB_DIR) $(SOEM_LIBS)
$(ALCHEMY_LIBS) $(shell xeno-config --
skin=$(XENO_SKIN) --ldflags)
CC = $(shell xeno-config --cc)
SRC_TARGET = $(TARGET:=.c)
OBJ_TARGET = $(SRC_TARGET:.c=.o)
OBJS = $(OBJ_TARGET)
all : $(TARGET)
$(TARGET) : $(OBJS)
    $(CC) -o $@ $< $(CFLAGS) $(LDFLAGS)
```

응용 프로그램의 실시간 동작을 위해 main() 함수에서 mlockall() 함수를 호출하여 메모리 스와핑을 방지하게 하고, 함수 rt_task_set_periodic()를 호출하여 제어 주기를 설정한 다음, 한 패킷을 전송한 후에 다음 패킷을 전송할 때까지 함수 rt_task_wait_period()를 호출하여 대기한다.

3.2.2 성능 평가

성능 평가는 EtherCAT 마스터 플랫폼의 소프트웨어 부

하가 있는 경우와 없는 경우를 비교하기 위해 'stress' 패키지를 설치하여 사용하였고[2], 부하를 주는 경우는 '-v -c 12 -i 12 -d 12' 옵션을 사용하여 stress를 주었다.

Fig. 2는 Xenomai test suite에서 제공하는 latency 툴을 이용하여 구현된 실시간 Linux커널의 latency를 측정한 결과이다. stress가 없는 경우는 대부분 0.5usec 이내의 latency를 가지고, stress가 있는 경우에는 1.5usec 이내의 latency를 가지므로 실시간 반응이 우수함을 확인할 수 있다.

Fig. 3은 EtherCAT 마스터의 제어 주파수가 1KHz일 때, Fig. 1과 같이 netAnalyzer로 패킷의 전송 주기를 측정한 결과를 히스토그램으로 나타낸 것이다. Linux의 네트워크 드라이버인 e1000e를 사용한 경우(검정색)보다 RTnet 드라이버 rt_e1000e를 사용한 경우(빨간색)이 정주기에 집중되어 나타나는 것을 볼 수 있고, 부하(stress)가 있는 경우에 제어 주기에 더 큰 jitter가 발생하는 것을 확인할 수 있다.

Fig. 4는 EtherCAT 마스터의 제어 주파수가 8KHz(제어 주기는 125usec)일 때, RTnet 드라이버 rt_e1000e를 사용하고 부하(stress)를 주지 않은 경우에 netAnalyzer의 타이밍 분석 결과이다. 총 204,781패킷을 분석한 결과로서, 주기의 평균은 124.82usec, 최소 주기는 124.82usec, 최대 주기는 128.01usec, 주기의 표준 편차는 196nsec로 정확한 주기를 가짐을 확인할 수 있다.

Fig. 5는 Fig. 4와 동일한 조건에서 EtherCAT slave의 LED 제어 신호를 오실로스코프로 측정한 결과이다. 제어 주파수가 8KHz이므로 LED를 on/off 토글하는 제어 신호의 주파수는 4KHz가 되어야한다. 실제 측정 결과가 주파수는 약 4.007KHz이고, 주기의 표준 편차는 234.4nsec이므로 정주기 제어가 잘 이루어지고 있음을 알 수 있다. 주기의 평균이 124.8usec로 Fig. 4의 결과와 일치함을 볼 수 있다. Fig. 5는 EtherCAT slave에서 EthrCAT 패킷을 수신한 후에

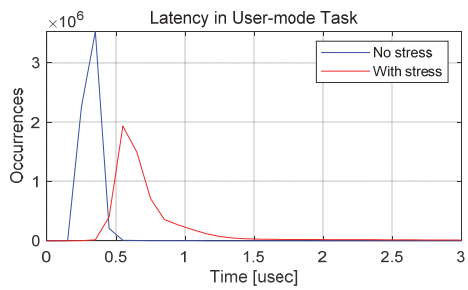

Fig. 2. Latency on Xenomai 3.1.

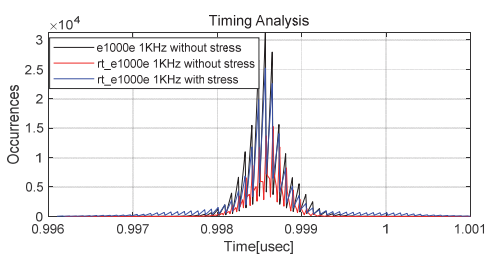

Fig. 3. Histogram of actual period at 1KHz control frequency.

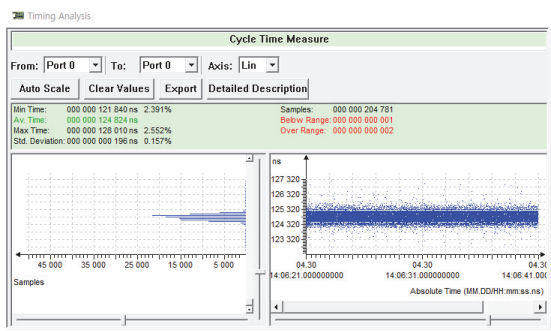

Fig. 4. Timing Analysis with netAnalyzer at 8KHz control frequency with rt_e1000e and no stress.

Table 4. Period measurement results in micro-second.

Control Frequency			1KHz		2KHz		4KHz		8KHz		10KHz		20KHz	
			B500G	Oscope	B500G	Oscope	B500G	Oscope	B500G	Oscope	B500G	Oscope	B500G	Oscope
Generic Driver (e1000e)	No stress	min	254.09	996.00	16.72	493.00	10.72	252.80	248.48	252.80				
		avg	998.57	998.40	499.24	499.40	254.70	254.60	254.90	254.70				
		max	1,485.14	1,000.00	1,697.62	505.00	1,712.26	259.60	1,741.14	261.20				
		Std. Dev.	3.90	0.81	6.06	0.78	3.92	0.63	4.54	1.20				
RTnet (rt_e1000e)	No stress	min	994.70	997.80	496.61	498.00	246.25	248.80	121.84	124.00	97.33	99.16	48.49	49.32
		avg	998.59	998.40	499.30	499.40	249.65	249.60	124.82	124.80	99.86	99.91	49.93	49.98
		max	1,003.14	1,002.00	503.61	500.10	253.37	250.40	128.01	125.60	103.52	101.00	52.40	50.40
		Std. Dev.	0.23	0.82	0.22	0.51	0.19	0.29	0.20	0.23	0.19	0.24	0.18	0.21
	With stress	min	992.41	994.00	490.17	494.00	240.16	244.00	115.28	119.20	90.96	96.66	39.92	45.99
		avg	998.59	998.90	499.30	499.40	249.65	249.80	124.82	124.80	99.86	100.00	49.93	50.01
		max	1,006.57	1,002.00	508.25	504.10	257.61	254.40	134.48	130.80	108.00	102.80	60.00	57.02
		Std. Dev.	0.59	1.08	1.16	1.08	1.12	0.93	1.15	0.64	1.07	1.05	1.09	1.22

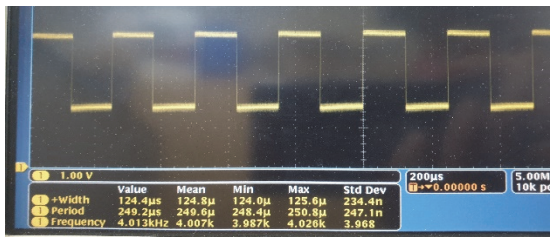

Fig. 5. Timing Analysis with oscilloscope at 8KHz control frequency with rt_e1000e driver and no stress.

LED 제어 신호가 출력된 것이므로 slave의 반응 시간도 포함된 결과임을 주목할 필요가 있다.

Table 4는 제어 주파수가 각각 1KHz, 2KHz, 4KHz, 8KHz, 10KHz, 20KHz 일 때 EtherCAT 마스터가 송신한 패킷의 주기를 측정한 결과이다. Table 4의 측정 값은 e1000e 드라이버를 사용하는 경우와 rt_e1000e 드라이버를 사용하는 경우에 대해 netAnalyzer와 오실로스코프로 주기를 측정한 결과이며, 부하(stress)가 있는 경우도 포함하고 있다. Table 4에서 B500G는 netAnalyzer로 EtherCAT 패킷의 주기를 측정한 결과를 의미하고, Oscope는 오실로스코프로 EtherCAT slave의 LED 제어 신호를 측정한 결과를 의미한다.

Linux의 네트워크 드라이버인 e1000e를 사용하는 경우의 평균 주기 값은 rt_e1000e을 사용하는 경우의 평균 주기 값과 큰 차이는 없지만, 최소 주기와 최대 주기에 큰 오차가 있음을 볼 수 있다. 또한 제어 주파수가 8KHz일 때는 패킷이 원하는 주기로 전송되고 있지 않음을 볼 수 있다. 반면에 rt_e1000e를 사용하는 경우에는 모든 제어 주파수에 대해 수usec이내의 오차로 주기가 제어되고 있음을 확인할 수 있고, 부하(stress)가 있는 경우에도 jitter가 증가하지만 여전히 수usec이내의 오차로 주기가 제어되고 있음을 볼 수 있다. 제어 주파수가 20KHz인 경우에도 안정적인 정주기 제어가 이루어지고 있고, 제어 주파수가 증가함에 따라 절대적인 jitter양은 증가하지 않지만, 제어 주기에 대한 상대적인 jitter 양은 증가한다고 볼 수 있다.

4. 결 론

본 논문에서는 Xenomai 기반의 실시간 Linux 커널과 공개 소스 EtherCAT 마스터 스택인 SOEM을 이용하여 실시간 EtherCAT 마스터를 구현하고, 정주기 및 실시간성에 대한 성능을 평가했다. 기존의 연구와 다르게, 제어 루프의 정주기성과 NIC에서 출력되는 전송 패킷의 정주기성을 측정함으로써 EtherCAT 마스터 시스템에서 발생하는 모든 영향이 반영된 실제 정주기 성능을 평가했다.

구현된 실시간 EtherCAT 마스터는 20KHz 제어 주파수에서도 안정적인 정주기 제어가 가능하고, 다양한 제어 주파수에 대해 수usec 이내의 오차로 정주기 제어가 가능하다.

감사의 글

이 논문은 2020학년도 한국기술교육대학교 연구연간제 연구비 지원에 의하여 연구되었음.

참고문헌

1. EtherCAT Technology Group, http://www.ethercat.org [accessed June 21, 2021]
2. C. Huang, C. Lin, C. Wu, “Performance Evaluation of Xenomai 3”, Available at http://wiki.csie.ncku.edu.tw/embedded/xenomai/rtlws_paper.pdf [accessed June 21, 2021]
3. S. Kim, E. Shin, “A Performance Evaluation of Open Source-based EtherCAT Master Systems”, in Proc. of the 4th ICCDR, pp. 128-1-128-4, 2017.
4. Xenimai, https://source.denx.de/Xenomai/xenomai/-/wikis/home [accessed June 21, 2021]
5. Igh EtherCAT master, https://www.etherlab.org/en/index.php [accessed June 21, 2021]
6. Open EtherCAT Society, Simple Open EtherCAT Master (SOEM), https://openethercatsociety.github.io/ [accessed June 21, 2021]
7. S. Park, J. Choi, “Cycle Time Improvement of EtherCAT Networks Using Linux Kernel Space Application Module”, in the Transactions of the KIEE, Vol. 69, No. 1, pp. 184-189, 2020.
8. https://xenomai.org/downloads/ipipe/v5.x/x86/ [accessed June 21, 2021]
9. R. Delgado, B. Choi, “New Insights into the Real-Time Performance of a Multicore Processor”, in IEEE Access, Vol. 8, pp. 186199-186211, 2020.
10. https://github.com/OpenEtherCATsociety/SOEM [accessed June 21, 2021]
11. RTnet, https://source.denx.de/Xenomai/xenomai/-/wikis/RTnet [accessed June 21, 2021]
12. https://rtt-lwr.readthedocs.io/en/latest/rtpc/rtnet.html [accessed June 21, 2021]
13. https://www.hilscher.com/products/product-groups/analysis-and-data-acquisition/ethernet-analysis/nanl-b500g-re/ [accessed June 21, 2021]
14. https://www.infineon.com/cms/en/product/evaluation-boards/kit_xmc48_relax_ecat_v1/ [accessed June 21, 2021]

접수일: 2021년 6월 16일, 심사일: 2021년 6월 21일, 게재확정일: 2021년 6월 21일