MOST 인터페이스를 위한 I2C 통신 드라이버의 구현에 관한 연구

성현용* · 장시웅*

*동의대학교 대학원 IT융합학과

A Study on I2C Communication Driver Implementation for MOST Interface

Hyun-yong Sung* · Si-Woong Jang*

Dongeui University

E-mail : hyunyong@deu.ac.kr, swjang@deu.ac.kr

요 약

차량용 멀티미디어 네트워크 시스템의 증가로 인해 MOST 인터페이스 모듈의 수요가 증가하고 있다. MOST 장치는 네트워크 컨트롤러인 INIC 부분과 마이크로 컨트롤러의 EHC 부분으로 구성된다. MOST 장치를 개발함에 있어서 EHC에서 INIC을 통하여 MOST 네트워크로의 효율적인 데이터 송수신을 하기 위해서는 적절한 장치 드라이버의 구현이 요구된다. 본 논문에서는 MOST 네트워크 컨트롤러가 지원하는 I2C, MediaLB, I2S 통신방식 중 MOST 네트워크상의 각 노드간 상태 및 제어 메시지를 전달하는데 이용하는 I2C 통신 드라이버 구현 방안을 제시한다. INIC을 통한 MOST 네트워크와의 효과적인 통신을 위해서 NetService API와 연계하여 I2C 통신 드라이버를 구현한다.

본 연구에서는 I2C 통신의 low level driver의 구현을 위해 MOST 오디오 인터페이스 장치에 통신 드라이버 소스를 포팅함으로써 테스트 하였으며, 향후 이에 대한 연구를 확장하여 OS 기반의 MOST 장치에 대한 다양한 드라이버를 개발할 예정이다.

ABSTRACT

The demand of MOST interface module is increasing with car-multimedia network system. MOST devices consist of INIC part which controls MOST network and EHC part which is used by user. The efficient data communication between EHC and INIC demands implementation of a proper device driver. This paper presents a design method for I2C communication driver which is used for transmitting control messages between nodes of MOST network. For effetive I2C communication, we design driver with NetService API.

For testing the experiment, we use the MOST audio interface deivce for porting driver sources and will develop various driver on MOST device based OS.

키워드

MOST, interface, I2C, driver

Ⅰ. 서 론

현대 사회에서의 자동차는 단순한 이동수단을 떠나서 복합적인 기능을 갖춘 IT 융합기술의 집합체라고 할 수 있다. 과거에는 자동차의 구성이 대부분 기계적 장치로 이루어졌으나, 오늘날에 이르러서는 빠른 속도로 전장장치가 차지하는 비중이 늘어가고 있다는 것을 그 예로 볼 수 있다.

전기, 전자적인 모듈을 기반으로 하는 전장 장치의 수요가 늘어나면서 각 장치간의 제어 및 통신을 위한 수단, 방법의 필요성이 대두되었으며, 이는 종래에 CAN, FlexRay, LIN, MOST 등의 자동차 네트워크 시스템을 위한 프로토콜이 출현하는 배경이 되었다.

자동차 버스 시스템은 이동 차량에서의 멀티미디어 응용 서비스에 대한 요구가 급증함에 따라 높은 대역폭을 가진 차량용 통신 프로토콜을 요구하게 되었는데 이를 만족하기 위한 것이 MOST(Media Oriented System Transport) 네트워크이다[1].

인포테인먼트 시스템의 발달로 차량에서의 멀티미디어 네트워크인 MOST의 이용률이 증가하

는 추세이고, 이에 따라 효율적인 MOST 장치의 개발도 더불어 요구되는 실정이다.

MOST 장치는 크게 운전자 측면에서의 EHC(External Host Controller)부분과 차량 내 통신 측면인 INIC(Intelligent Network Interface Controller)부분으로 나뉘며, 광섬유 재질의 Ring 형 구조로 이루어져 있다. 운전자의 조작을 통해 EHC에서 INIC을 거쳐 MOST 네트워크로의 데이터 송수신이 효율적으로 이루어지기 위해서는 EHC와 INIC간의 데이터 전달을 위한 적절한 장치 드라이버의 구현이 요구되는데 MOST 프로토콜에서는 I2C, MediaLB, I2S의 통신방식을 지원하고 있다[2].

본 연구에서는 각 MOST 장치간의 상태 및 제어 메시지를 전달하는데 사용하는 I2C 통신 드라이버를 구현함에 있어 NetService API를 참조하여 최적화된 방안을 제안하고자 한다.

본 논문 2장에서는 I2C 통신 드라이버의 관련 연구에 대해 설명하고, 3장에서는 I2C 통신 드라이버 모듈 구성과 MOST 오디오 인터페이스 장치를 기반으로 구현된 드라이버를 제시하고, 4장에서는 결론을 서술한다.

II. 관련 연구

1. I2C 제어

I2C는 Philips에서 제안한 IC 간의 통신방식으로 클럭(SCL)과 데이터(SDA) 라인을 사용하는 동기 양방향 2선식 제어방식으로써 TWI(Two Wire Interface)라고도 한다. 버스에 연결된 각 디바이스는 고유의 어드레스를 가지며, 필요에 따라 송신기와 수신기로도 동작한다. 전송속도는 일반모드에서 100kbps까지 지원되며, 고속모드에서는 400kbps까지 지원가능한 통신방식으로 동작원리는 다음과 같다[3].

(1) Transmitter : 버스에 데이터를 전송하는 장치

(2) Receiver : 버스로부터 데이터를 수신하는 장치

(3) Master : 전송을 개시하는 장치로 클럭 펄스를 생성하며, 전송을 종료함.

(4) Slave : Master가 어드레싱한 장치

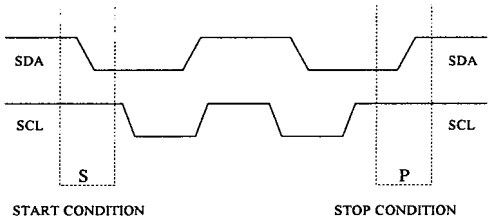

그림 1. I2C통신 타이밍 구조

Master가 초기화되면 버스에 연결된 장치들은 데이터를 수신하기 위해 대기한다. Master가 Slave 어드레스를 전송하면 각 칩은 자신의 고유 어드레스와 비교한다. 어드레스가 일치하는 칩은 이어지는 확인신호 구간에서 데이터를 Low로 유지해 응답을 보낸다. 그러면 Master는 데이터를 읽거나 쓰기를 할 수 있다. 모든 작업이 끝나면 Master는 종료상태를 발생하고 버스를 Release한다[4].

클럭이 Low일 때만 데이터 상태를 바꿀 수 있기 때문에 데이터는 클럭이 High인 구간에만 유효하다. 따라서 Master는 데이터 라인을 적당한 데이터로 바꾸고, 클럭 라인을 일정시간 놓아둔 다음 SDA 라인의 상태를 바꾸기 전에 클럭을 Low로 만든다. 데이터는 8bit이고 MSB가 먼저 전송된다. 초기상태와 종료상태 사이에 전송되는 데이터의 수는 제한이 없다.

2. NetService API

MOST NetServices는 애플리케이션과 MOST 네트워크 사이의 전송 메카니즘을 제공해주는 API로써, MOST 장치의 개발시간 감소, 개발비용 절감, 호환성의 증대 등의 장점을 가진다. API의 구조는 아래 그림 2와 같다.

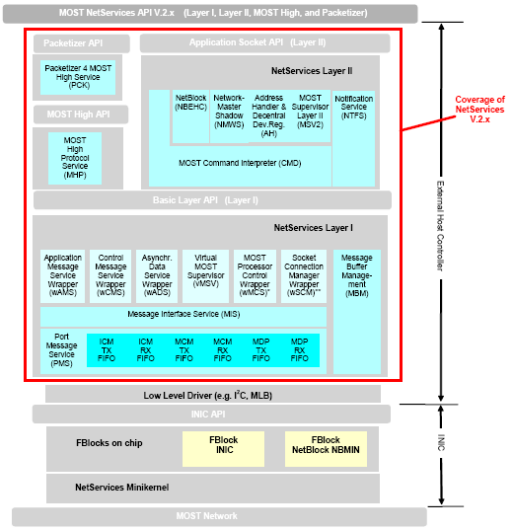

그림 2. NetService API 구조

NetServices API는 크게 Layer II, Layer I으로 나뉘며, 그밖에도 LLD(Low Level Driver), INIC API, Minikernel 등으로 이루어져 있다[5].

I2C 통신 드라이버가 포함되는 LLD 부분은 EHC와 INIC간의 통신을 효율적으로 해주기 위해서 존재하는데 이를 구현하기 위해서는 Layer I의 각 모듈에 관해서 상세히 알 필요가 있다.

그림 3은 NetServices API를 이루는 각 헤더 파일의 연관성을 나타내고 있다.

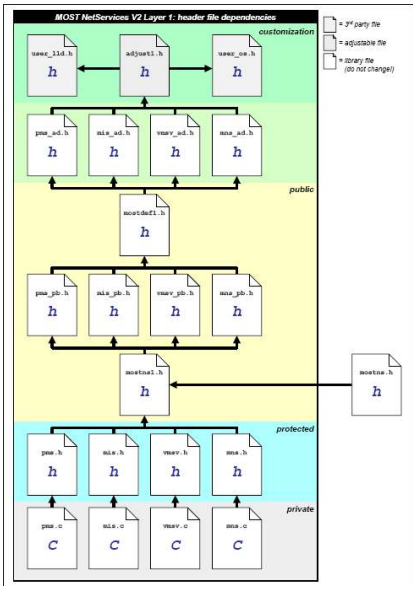

그림 3. NetService API Layer I 헤더파일의 연관성

3. INIC API

INIC API는 EHC와 INIC 사이의 통신을 위해 제어 인터페이스로써 제공된다. INIC API는 소위 Port 메시지 형태로 INIC의 메시지 인터페이스를 통해 송수신되며, 크게 두 개의 Function Block인 NetBlock NBMIN과 INIC FBlock을 제공한다. NetBlock NBMIN은 MOST 장치에 영향을 주는 모든 함수를 포함하고 있으며, FBlock INIC은 서로 다른 API의 그룹으로 이루어져 있는데 내부의 EHC와 네트워크 상에 있는 EHC에 의해 접근이 가능하다. 아래의 그림 4는 INIC의 소프트웨어 stack 구조를 나타내고 있다[6].

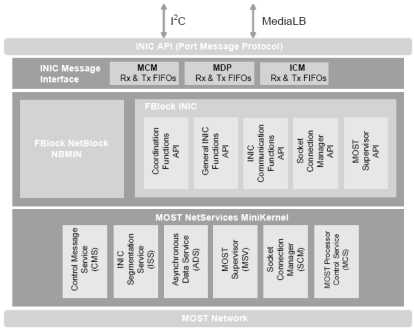

그림 4. INIC Software Stack

III. I2C 통신 드라이버의 구현

본 연구는 MOST INIC 인터페이스와 EHC 모듈간의 효율적인 데이터 변환, 제어, 송수신을 위해 최적화된 I2C 통신 드라이버 구현 방안을 제시하는 것이다.

I2C 통신은 EHC와 INIC 부분을 각각 마스터 또는 슬레이브로 인식하여 상호간에 데이터를 주고 받으며, Master(EHC)의 프로세서 종류에 따라 데이터 발생시 제어하는 레지스터의 형태, 종류, 방법이 달라진다. 즉, Master와 Slave 이외에도 I2C 통신 드라이버를 최적화하기 위해 고려되어야 할 사항이 있다. I2C 통신은 프로세서 Chip의 핀을 통한 인터럽트(이벤트 발생) 형태의 데이터 송수신 방식이다. 활용 과정은 그림 5와 같다.

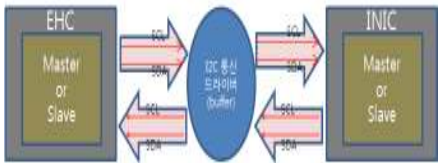

그림 5. I2C 통신 드라이버 역할

1. I2C 통신 드라이버 설계

I2C 통신 드라이버 설계를 위해 NetServices와 INIC간의 통신 관련 함수를 포함하는 i2c_lld 모듈과 i2c 인터페이스 관련 함수를 포함하는 i2c_driver 모듈로 나눠서 설계하였다.

구조는 크게 NetServices에서 INIC단으로 데이터를 전송할 때와 INIC에서 NetServices로 데이터를 수신할 때로 나누었으며 전송시의 구조는 그림 6과 같다.

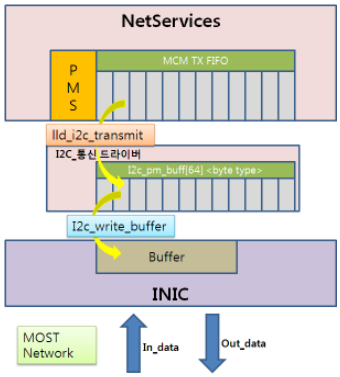

그림 6. I2C 통신 드라이버를 활용한 데이터 전송 구조

I2C 통신 드라이버를 통한 제어 메시지의 전송은 TWI 하드웨어 핀 초기화 및 I2C 인터페이스의 에러 점검을 담당하는 i2c_master_init() 함수 호출을 시작으로 알고리즘이 구현되며 전송 메시지가 있을 때마다 lld_i2c_transmit()을 호출하여 PMS(Port Message Service) 상의 FIFO 버퍼에 있

는 데이터를 임시 I2C 드라이버 버퍼인 i2c_pm_buffer에 옮긴 후에 i2c_write_buffer()를 통해 INIC 상의 제어 포트인 I2C 주소(0x40)에 최종적으로 데이터를 전송할 수 있게 구현하였다.

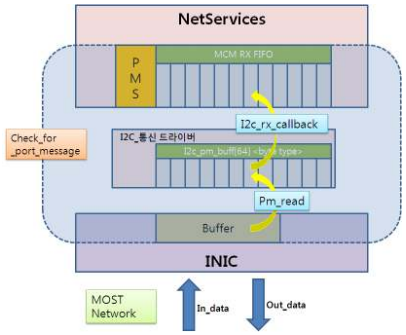

그림 7. I2C 통신 드라이버를 활용한 데이터 수신 구조

INIC상에서 NetService로의 데이터 수신은 main 모듈에서 Check_for_port_message() 함수가 while(1)문을 통해 네트워크로부터 수신되는 데이터를 지속적으로 감지하게 하였다. 감지가 되면 i2c_driver 모듈의 Pm_read() 함수를 통해 I2C의 임시 버퍼로 데이터를 저장하고 그 데이터 크기를 전달하여 PMS에서 미리 받을 공간을 확보할 수 있게 하며, i2c_rx_callback() 함수에 의해 임시 버퍼에서 NetServices 상의 FIFO로 데이터를 수신하도록 설계하였다.

그 외에도 에러 발생에 대한 lld_reset() 함수와 버퍼 공간을 비우는 lld_on_buf_freed()를 구현하였으며, driver 모듈내에서도 단일, word 타입의 데이터를 쓰고 읽는 함수 등을 구현하여 좀 더 유동적인 통신이 될 수 있게 하였다.

2. MOST 오디오 플랫폼 기반의 테스트

I2C 통신의 Low Level Driver 구현을 위해 MOST 오디오 인터페이스 장치를 사용하였다. MOST 오디오 플랫폼의 아키텍쳐는 아래의 그림 8과 같다[7].

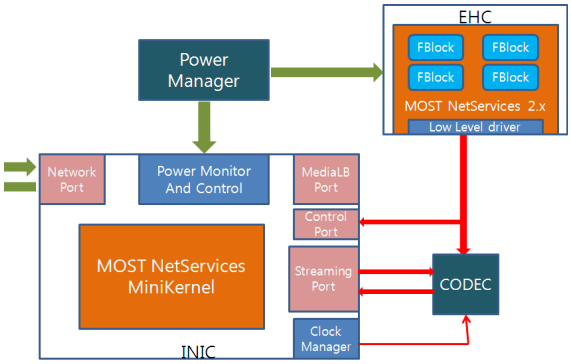

그림 8. MOST 오디오 인터페이스 블록도

위 MOST 오디오 장치에 모듈 hex 파일을 포팅한 후에 네트워크상에 노드로써 연결하여 동작을 실험하였으며, 정상적인 동작을 관찰할 수 있었다. 테스트 환경은 아래 표 1과 같다.

MPU	컴파일러	포팅 인터페이스
ATmega128	IAR	RS232

표 1. I2C 통신 드라이버 테스트 환경

IV. 결론 및 향후계획

본 논문에서는 EHC와 INIC간의 통신을 효율적으로 하기 위한 I2C 통신 드라이버 구현 방안을 제시하였다. 그러나 위 결과가 모든 MOST 장치에서 적용할 수 있는 I2C 통신 드라이버라 할 순 없다. INIC단에서 사용되는 API 및 하드웨어적인 스펙과 EHC 단의 NetService API, MCU 하드웨어 스펙을 어떤 식으로 분석하고 적용하는가에 따라서 그 상황에 맞는 최적의 I2C 통신 드라이버를 설계할 수 있음을 알 수 있었다.

향후 MOST 장치 기반의 통신 드라이버 개발을 진행하면서 적용하고자 하는 분야는 OS가 탑재된 임베디드 보드를 EHC단으로 하여 MOST 인터페이스와의 장착시에 자동 인식하고 원활한 통신이 가능하도록 하며, I2C뿐만 아니라 MediaLB, I2S 통신에서도 최적화된 드라이버를 설계하고 구현하는 것이다.

참고문헌

[1] The MOST System, Franzis Verlag, Andreas Grzemba, 2008

[2] 이건용, 자동차의 통신 대동맥 버스 규격 총집합, 임베디드 월드, automotive Bus System

[3] 이성우, 김정익, 황재문, 박진수 “I2C 제어 기법을 이용한 디지털 영상처리 시스템의 설계”

[4] 장원영, 문상필, 서대화, "DTV에 내장된 리눅스의 I2C Device Driver 제작“

[5] MOST NetServices Layer 1 V2.1.x User Manual/Specification, SMSC

[6] MOST INIC API User's Manual, SMSC

[7] MOST INIC Audio Interface V1.0x User Manual, SMSC