공학석사 학위청구논문

E-Navigation을 위한 선박용 통신 프로토콜 구현에 관한 연구

A Study on the Implementation of Marine Network Standard for E-Navigation

지도교수 황 승 욱

2008년 8월

한국해양대학교 대학원

제어계측공학과

백 종 옥

本 論文을 白宗玉의 工學碩士 學位論文으로 認准함.

위 원 장　　김 기 문　(인)

위　　원　　이 서 정　(인)

위　　원　　황 승 욱　(인)

2008년 06월 25일

한국해양대학교 대학원

제어계측공학과

목 차

표 차 례

그 림 차 례

Abstract

Domestic electronic marine instruments, navigation equipment and communications equipment technique for a ship is very falling behind compare with Japan or European countries. Especially, there are no organized cooperation systems in terms of standardization activity. Because of this reason, the companies, which make electronic marine instruments, navigation equipment and communications equipment, have lots of problems with interfacing each device.

Presently, there is a trend that domestic products follow IEC61162-1/2. In this case, however, wiring is too complex and when a user wants to add the other device, due to lack of NMEA signal out port or no composition of Isolation, it needs additional device, such as NMEA Buffer.

If you develop the device which is supporting IEC61162-3 standard protocol, you will be able to solve these all problems. Besides, you are able to make a lot of expectative effects.

By materializing the protocol to support IEC61162-3 standard, I have learned about standard protocol perfectly and it will apply to the other company's product, as a crucial technique. Also, it is expected to making a profit both in a domestic market and in an oversea market.

In this paper, I would like to introduce NMEA standard, such as IEC61162-1 (NMEA0183), IEC61162-2 (NMEA0183-HS) and IEC61162-3 (NMEA2000), and the way of approaching to protocol design.

Also, I will introduce CAN and materializing stack to support

IEC61162-3. Later then, for MiTS system I am going to deal with TCP-IP protocol and materializing stack for E-Navigation.

제 1 장 서론

1.1 연구 배경

본 논문은 점차 복잡해져 가는 선박 IT 장비들을 통합하고, 앞으로 구현될 선박 내 e-Navigation 체계를 지원하기 위한 선박 통신 네트워크 구현에 대한 내용으로써, 선박과 관련된 선박 내외의 모든 정보를 통합할 수 있는 요소기술인 NMEA0183, TCP-IP, NMEA2000 통신 프로토콜 스택의 설계 및 구현에 관한 내용이다.

1980년대에 들어서서 미국의 NMEA(National Marine Electronic Association)에서 Autopilot(Position/Steering 데이터)를 위한 NMEA0180 및 NMEA0182 시리얼 인터페이스 표준을 제정하였다. NMEA0180/0182는 세계 최초의 선박 인터페이스 표준이며, 1,200bps의 저속 시리얼 인터페이스 방식이었다. 1983년 NMEA는 NMEA0183 표준 규격을 제정하였으며, 이는 4,800bps 속도의 시리얼 데이터 통신 규격 및 모든 선박 장비들을 위한 포괄적인 데이터 포맷을 포함하고 있다. [1]

1990년대에 들어 IEC(TC80/WG6)는 NMEA0183 규격을 그대로 수용하여 IEC61162-1 규격으로 제정하였으며, NMEA0183 High Speed를 IEC61162-2로 수용하고 NMEA2000을 IEC61162-3으로 2008년 최종 채택하였다. 또한 유럽에서 연구한 MiTS를 바탕으로 하여 IEC61162-4 규격을 제정하기에 이르렀다.

NMEA0183은 1:1 또는 1:n의 연결의 Point to Point 방식으로 접속이 1개의 Talker와 1 개 이상의 Listener로 이루어지기 때문에 Talker는 일방적으로 데이터를 내 보내며, Listener는 Talker에서 보내는 데이터를 받기만 한다. 또한 Talker의 출력 포트는 제한적이다. NMEA0183은 4,800bps 또는 38,400bps의 통

신 속도를 표준으로 하고 있으며, 전송 대역폭은 대략 1초에 10문장정도로 되어 있다. 예를 들어 4,800bps에 400bits의 문장을 전송한다고 하더라도 1초에 12문장밖에 전송할 수 없는 한계가 있다.

GYRO COMPASS 장비의 경우, 선박의 정보를 최소 초당 15문장의 메시지를 전송하여야 한다. 이러한 경우 NMEA0183 프로토콜에 의해서는 전송이 불가능한 상황이 발생하게 된다.

1990년대 중반 COMPASS, ECDIS(Electronic Chart Display and Information System), VDR(Voyage Data Recorders), AIS(Automatic Identification System)와 같은 선박용 항해통신장비들의 기능이 고도화됨에 따라, 연결 노드 수의 증가와 데이터 처리량의 증가로 기존에 사용하던 NMEA0183 및 NMEA0183 High Speed 방식을 개선하는 보다 간결하며 고속 데이터 처리가 가능하고 실시간 데이터 교환이 가능한 새로운 방식의 네트워크 표준이 필요하게 되었다.

NMEA2000은 1994년부터 미국의 NMEA의 주도로 미국의 Coast Guard 및 대학, 전 세계 여러 항해통신장비 업체, CAN 솔루션 업체들이 공동으로 개발을 시작하였으며, 2001년 9월 최초의 규격이 완성된, 선박 전자 장치들을 상호 연결하는 저가격, 적정 처리속도, 양방향, 다수의 송신기/수신기가 연결될 수 있는 실시간 인스트루먼트 네트워크 이다.

NMEA2000 프로토콜은 1:1 또는 1:n 접속 방식인 NMEA0183 보다 발전된, 최신 표준 규격 네트워크 방식의 실시간 프로토콜로써 항해 및 자동화 장비 등의 인터페이스뿐만 아니라, 다양한 선내외의 정보를 통합, 관리하고 육상의 광대역 Web-based 정보체제를 해상까지 연결시키는 요소기술이다. 250kbps 수준의, 충돌회피가 가능한 CAN 버스를 사용하는 실시간 인스트루먼트 네트워크로서 소형선박을 시작으로 점차 그 보급이 확대되고 있다. IEC에서는 IEC61162-3 규격으로 수용하였으나 아직 정식으로 발표는 하지 않은 상태로 있다.

본 논문은 NMEA2000 프로토콜 개발에 관한 내용으로서, 본 연구결과는 이 후 IEC61162-4 관련 규격을 포함하는 라우터 개발에 응용될 수 있으며, 또한 항해통신 장비의 표준화 부품으로 사용될 수 있어, 장비의 개발기간 단축과 제작기간 및 생산원가 절감효과도 기대된다. 또한 점차 확대되고 있는 ECDIS의 설치와 MiTS(Marine Information Technologies Standard) 구현 및 선박 내 e-Navigation 시스템 구축에 획기적인 기여를 할 수 있을 것이다.

1.2 연구의 목적 및 내용

국내 선박용 항해·통신 장비 업체들에서 제작된 장비들은 NMEA0183 표준 규격을 따르고 있다. 하지만 선박 내 메인 시스템 장비들은 대부분 수입제품들이며, 이 장비들과 연결이 되는 경우 호환성 문제를 일으키는 경우가 많다. 이 원인은 국내 업체들이 NMEA0183 표준 규격에 대한 배경 지식의 부족 및 표준 규격을 제대로 이해하지 못하고 제품을 설계하여 초래한 문제들로 보여 진다.

이러한 문제를 해결하기 위해서는 NMEA0183 표준 문서를 토대로 표준 규격에 대한 체계적인 연구가 이루어져야 하며, 이를 통하여 NMEA0183을 수신할 수 있는 하드웨어 설계 및 구현, 수신 데이터를 처리 할 수 있는 NMEA0183 Parsing 알고리즘을 설계 및 구현하여 국내 업체들에서 쉽게 이용할 수 있도록 보급되어야 한다.

본 논문은 선박용 항해·통신 장비들 간의 상호 연동을 위한 디지털 인터페이스 표준에 대한 내용으로서, NMEA0183, NMEA2000, MiTS(Maritime information Technology Standards) 프로토콜을 지원하는 선박용 통신 프로토콜 처리 장치의 개발에 관한 내용이다. 이를 위해 하드웨어를 구현하고, 소프트웨어 프로토콜 스택을 설계·구현하였다. 또한 차세대 선박용 항해·통신 장비들 간의 인터페이스를 지원하기 위하여, NMEA2000 기반 하에 추가적으로

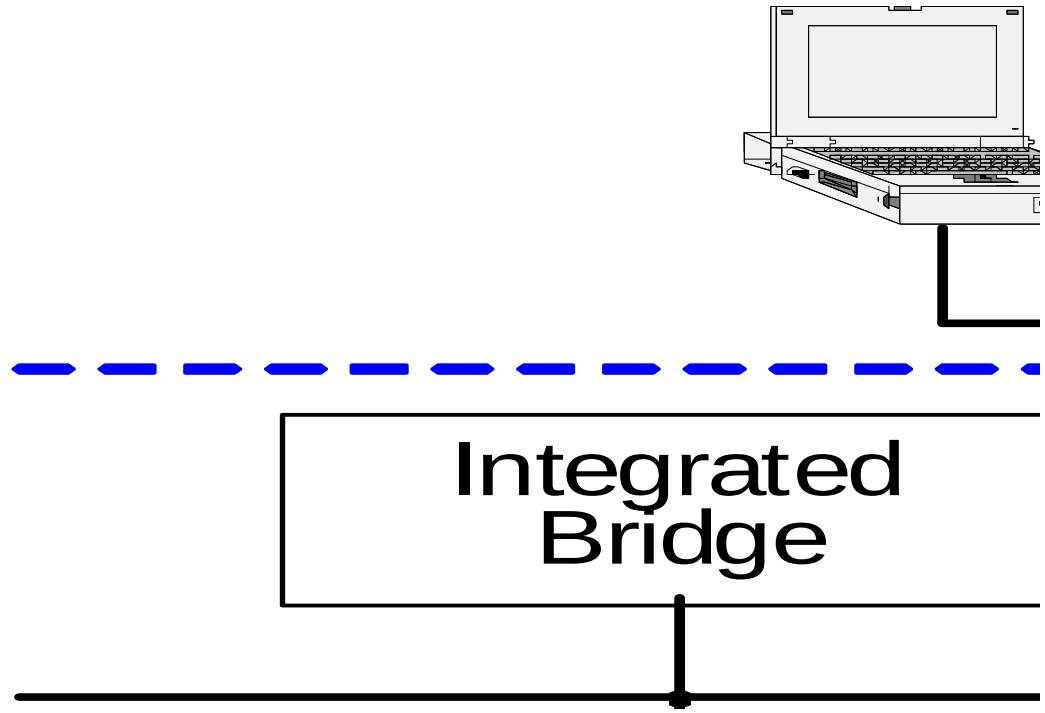
ECDIS, VDR등에서 사용하는 TCP-IP 인터페이스를 구현하여 좀 더 다양한 장비들 간에 연동이 가능하도록 구성하였다.

본 논문에서 설계한 하드웨어는 16 Channel의 UART(Universal Asynchronous Receiver Transmitter) 포트, CAN 인터페이스 포트, Ethernet 인터페이스 포트 및 소프트웨어에 의한 프로토콜 스택 구현 및 데이터 임시 저장 공간을 위한 외부 SRAM을 추가 구성하였다. 소프트웨어는 Micrium사에서 제공하는 uCOS-II 라는 RTOS를 STR912 메인 컨트롤러에 포팅 하여, Multi-Tasking 기법을 이용하여, NMEA0183 메시지 송·수신 TASK, NMEA2000 메시지 송·수신 TASK 및 TCP-IP 메시지 송·수신 TASK를 생성하였으며, 각 TASK의 동기화를 위하여 세마포어(Semaphore)를 이용하였으며, RTOS상에서 제공하는 메시지 큐 관리 기능을 사용하여 각 프로토콜별로 관리하여야 하는 메시지를 관리하였다. STR912 제작회사인 ST사에서 제공하는 최적화된 메모리 버퍼 복사 알고리즘을 이용하여 메시지 큐에서 메시지들을 이동하도록 하였으며, TCP-IP의 경우는 CPU에서 제공하는 DMA 기능을 이용하여 수신 메시지에 대해서 수신 버퍼에 적재되도록 하였다.

본 논문의 구성은 제1장 서론에 이어 다음과 같다. 제2장에서는 선박용 항해·통신 장비들 간의 디지털 인터페이스 표준에 대한 분석 및 정리하였고, 제3장에서는 선박용 항해·통신 장비들 간의 디지털 인터페이스 표준에 대한 프로토콜 스택을 설계 및 구현하였고, 하드웨어를 설계 및 구현하였다. 제4장에서는 설계·구현된 사항들에 대하여 실험 및 검토하였고, 마지막으로 제5장 결론 순으로 구성하였다.

제 2 장 선박용 통신 프로토콜

<그림 2-1>은 선박 내 네트워크층을 나타낸 그림으로, 제일 아래층은 “비디지털 인터페이스” 층이며, 2번째 층은 1:1 또는 1:n 구성의 NMEA0183 인터페이스이며, 3번째 층은 n:n 접속이 가능하며 실시간 처리가 필요한 NMEA2000 인스트루먼트 네트워크층을 나타낸다. 4번째 층은 선박 전체를 제어하고 모니터링하는 단계인 MiTS층을 나타내며, 최상위 층은 선박과 선박 간 또는 선박과 육상간의 통신을 위한 층을 나타내고 있다. [2]

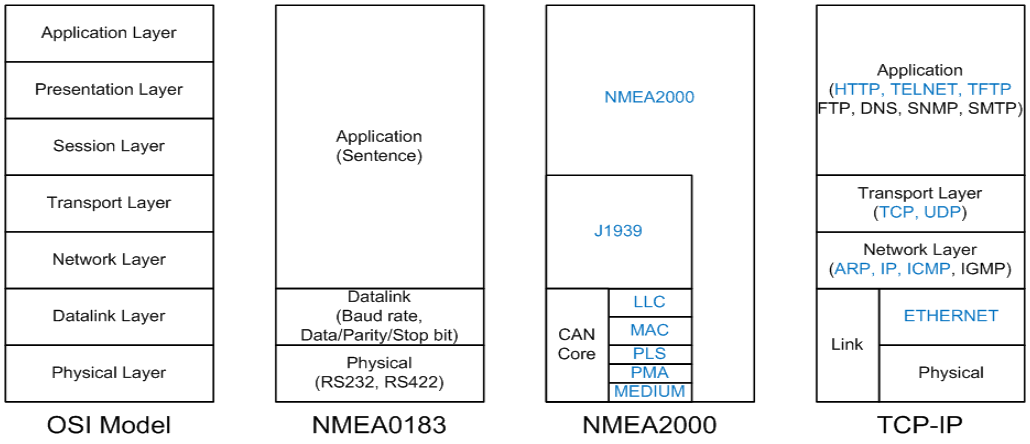


<그림 2-1> 선박 내 네트워크 계층도

<Fig 2-1> Shipboard Networks and Interfaces

본 논문에서는 <그림 2-1>에서 나열된 선박용 통신 프로토콜중 NMEA0183, NMEA2000 및 TCP-IP를 중점적으로 다룰 예정이며, <그림 2-2>에 OSI 7 Layer

모델과 각 계층을 비교하여 나타내었다. OSI 7 Layer의 각 계층에 특징 및 기능을 이해한다면, 각 통신 프로토콜별로 어떠한 기능들을 하는 부분인지 이해하는데 더 도움이 될 것이다.



<그림 2-2> 선박용 통신 프로토콜 비교

<Fig 2-2> The comparison of Marine Network Standards

NMEA0183은 3 계층 구조, TCP-IP의 경우는 5 계층 구조, NMEA2000은 5 계층 구조의 형태를 보이고 있다.

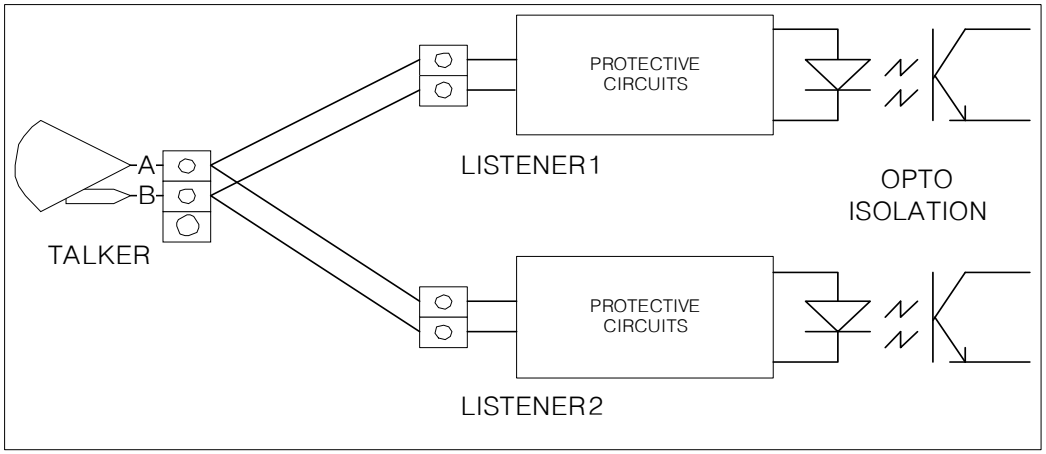
2.1 NMEA0183 프로토콜

2.1.1 NMEA0183 프로토콜 개요

NMEA(National Marine Electronics Association)는 선박 전자 업무에 대한 제조업자, 판매 대리점, 판매자, 교육기관 및 관심 있는 사람들에 의한 비영리 단체이다. 이중에서 NMEA0183 표준은 선박용 전자 장비들 간의 통신을 위한 전기적인 인터페이스와 데이터 프로토콜을 정의하고 있다.[3]

NMEA0183 표준은 Binary 포맷의 '1' 과 '0' 을 사용한 디지털 데이터 전송 방식이며, 4,800bps와 38,400bps를 표준으로 하는 시리얼 데이터 버스를 위한 전기적인 신호, 데이터 프로토콜 및 문장형식에 관하여 정의한다.[4]

GPS, Depth Sounder, Gyro Compass등 NMEA 데이터를 전송하는 장비를 “Talker"라고 하며, ECDIS, VDR, RADAR, NMEA DISPLAY등의 NMEA 데이터를 수신하는 장비들을 ”Listener"라고 한다. NMEA0183의 장비들 간의 연결은 <그림 2-3>와 같이 이루어지며, 1개의 “Talker"와 여러 개의 ”Listener"로 구성될 수 있다.



<그림 2-3> NMEA0183 기본적인 연결도
<Fig 2-3> Wiring in NMEA0183 Network

2.1.2 NMEA0183 프로토콜

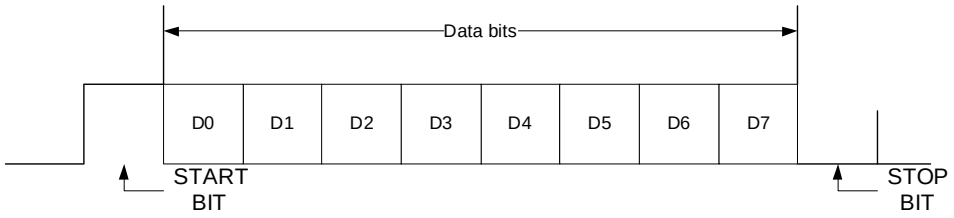
1) 하드웨어 사양

NMEA0183은 장치 사이에 차폐꼬임쌍선(STP) 케이블을 사용하도록 되어 있으며, 실드(Shield) 케이블을 구성하도록 되어 있는데, 모든 Listener의 Shield 케이블은 Talker와 Listener의 Chassis와 연결하는 것을 권장하고 있다. NMEA0183에서는 별도의 표준 커넥터 규격은 정의되어 있지 않으며, 신호는 EIA RS422방식으로 ‘A', 'B'라는 2개의 신호라인에 의한 차동(Differential) 방식을 권장하며, TTL Level과 같은 단일 종단(Single-Ended) 방식도 수용하도록 되어 있다. 그래서 'A'신호는 PC의 RS232 커넥터의 수신(RxD)단자에, 'B'신호

는 PC의 RS232 커넥터의 그라운드(GND)단자에 연결하여 사용이 가능하다. NMEA0183의 수신 회로는 보호회로와 함께 Opto-Isolator를 사용하는 것을 권장하며, 수신기의 그라운드(Ground)로부터 입력 신호를 절연하여야 한다.[5]

2) 데이터 전송

NMEA0183의 데이터 표현은 ANSI 표준인 7Bit ASCII에 의한 비동기 시리얼 통신 방식을 사용하도록 되어 있다. <그림 2-4>과 같은 구조로 데이터를 표현하도록 되어 있다.



<그림 2-4> NMEA0183 데이터 표현
<Fig 2-4> NMEA0183 Data Transmission

실제 데이터는 8Bit이며, NMEA0183은 4800bps, NMEA0183-HS는 38,400bps를 표준으로 규정하였다.

3) 데이터 형식 규정

NMEA0183은 <표 2-1>과 같이 예약되어서 사용되는 문자들이 있다. 문장의 시작을 나타내는 '$' 또는 '!', 문장의 끝을 나타내는 '<CR>'와 '<LF>', 문장의 각 영역을 구분하는 ',' 및 검사합(Check-Sum) 영역을 표시하는 '*'등을 대표적으로 사용하게 되어 있다. NMEA0183에서 사용하는 문자는 0x20 ~ 0x7E 사이의 모든 문자를 사용할 수 있으며, 표 2-1에 소개한 예약어를 제외한 출력

가능한 문자를 사용한다. [6]

<표 2-1> NMEA0183 예약어

	Hex	Dec	Description
<CR>	0D	13	Carriage return
<LF>	0A	10	Line feed
$	24	36	Start of Parametric sentence delimiter
*	2A	42	Checksum field delimiter
,	2C	44	Field delimiter
!	21	33	Start of Encapsulation sentence delimiter
\	5C	92	Reserved for future use
^	5E	94	Code delimiter for HEX representation of ISO8859-1 characters
~	7E	126	Reserved for future use
<del>	7F	127	Reserved for future use

정의되지 않은 문자를 표현하기 위해서는 <표 2-1>에 소개한 '^'예약어를 사용하여 그 뒤에 2자리의 ASCII 문자로 표현함으로써 표현할 수 있다.

NMEA0183 문장은 최대 82자로 구성되며, 시작 구분자와 종결 구분자를 제외하면 79자로 구성되며, 4,800bps는 초당 480자를 전송할 수 있는 속도이므로 82자로 구성된 문장을 전송할 경우 대략 1초에 6문장을 전송 가능하다.

2.1.3 문장

NMEA0183에서 정의된 문장은 Parametric, Encapsulation, Query, Proprietary 문장으로 구성되어 있으며, 대표적으로 가장 많이 사용되는 것은 Parametric 문장이다. Parametric 문장은 '$' 구분자를 시작으로 하는 문장으로 NMEA0183 표준에서 정의한 승인된 문장(Approved Sentence Formatter) 형

식들을 표현하기 위한 문장이다. <표 2-2>에 문장의 구조를 풀어서 소개하였다.

<표 2-2> NMEA0183 매개변수 문장

$aaccc,c--c*hh<CR><LF>		
Asc	Hex	Description
$	24	문장의 시작을 의미하는 구분자
aa		Talker의 식별자 부분
ccc		문장의 형식을 정의
,	2C	영역을 구분하기 위한 구분자
c--c		Data 영역으로 문장의 형식에 따라 정의된 영역
*	2A	Checksum 값의 시작을 구분하기 위한 구분자
hh		Checksum 영역
<CR> <LF>	OD OA	문장의 끝을 의미하는 구분자

Talker의 식별자의 경우 GPS는 'GP'로 표현되어 있다. 문장의 형식은 GPS의 위치 정보의 경우 'GGA', 선박의 헤딩정보의 경우 'HDT'로 표현되어 있다.

2.2 NMEA2000 프로토콜

2.2.1 NMEA2000 프로토콜 개요

NMEA2000은 선박용 항해·통신 장비들의 IT화에 따른 데이터 교환의 필요성에 의해, 연결 노드수의 증가와 데이터 처리량의 증가로 기존에 사용하던 NMEA0183방식으로는 불가능해져, 보다 간결하고 고속 데이터 처리가 가능하고, 실시간 데이터 교환이 가능한 새로운 방식의 네트워크 표준이다.

NMEA2000은 1994년부터 미국의 NMEA의 주도로 여러 업체들과 공동으로 개발이 시작되었으며, 2003년 최초의 규격이 완성된, 선박 전자 장치들을 상호 연결하는 저가격, 적정 처리속도, 양방향, 다수의 송신기/수신기가 연결될 수 있는 인스트루먼트 네트워크이다. [7]

NMEA2000은 CAN 기술을 하위 계층의 규격으로 하여 자동 산업을 위한 장비들을 연결하고, SAE J1939를 중간 계층의 규격으로 하는 상위 레벨 프로토콜이다. 시리얼 데이터 버스 표준인 NMEA0183의 차기 표준으로 구상되었으며, 250kbps의 높은 데이터 전송 속도와 간결한 Binary 메시지 형식이다. 또한 Multi-Talker, Multi-Listener로 구성되는 데이터 네트워크이며, 단일 채널 병렬 버스 방식이다.

표준 버스 프로토콜인 CAN을 이용하여 충돌을 감지하여 충돌을 회피하도록 되어 있으며, 메시지에 우선순위를 부여할 수 있어, 낮은 지연시간 및 실시간 메시지 전송을 가능하도록 한다. SAE J1939에서 지원하는 Single/Multi Packet 모드 외에 Fast Packet 모드를 추가적으로 제공한다.

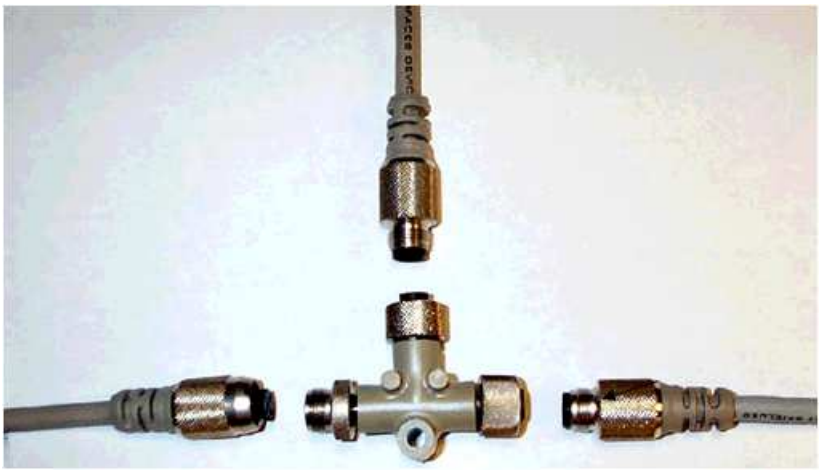
NMEA2000 표준은 데이터 포맷과 네트워크 프로토콜과 장비들을 인터페이스하기 위한 최소한의 물리적인 요구사항을 정의하며, 한 점에서의 장애로 인하여 전체 네트워크가 다운되지 않도록 하는 설계가 필요하며 이러한 내용들이 표준에서 제안되었다. 이 표준으로 설계된 장비는 하나의 신호 채널을 사용하

여 호환성이 있는 장비들 간에 명령과 상태정보 및 데이터를 공유할 수 있다. 데이터 메시지는 데이터 프레임의 직렬형태로 전송되며, 각각의 강력한 에러 검출 기능과 프레임 전송 확인, 실시간 전송이 보장된 데이터 전송을 보장한다. 데이터 프레임의 실 데이터는 전체 송신 데이터의 50%정도다.

NMEA2000 표준은 ISO/OSI 모델의 물리층부터 응용층까지 모든 계층을 정의하고 있다.

1) 물리 계층 (Physical Layer)

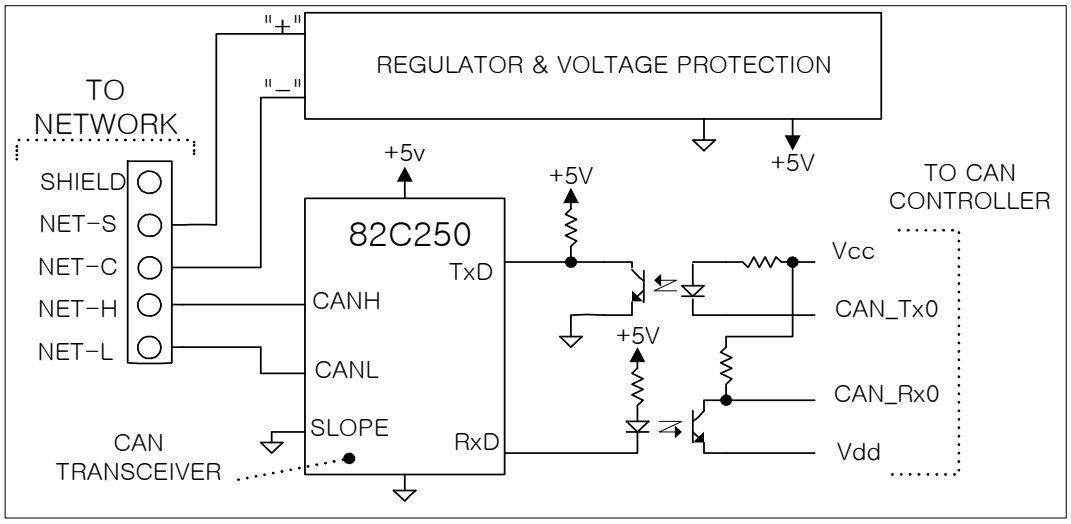
NMEA2000은 사용하는 케이블 및 커넥터에 대한 사양도 표준으로 규정되어 있다. 케이블은 굵기, 색상, 표식을 하게 되어 있으며, 전원 2라인, 신호 2라인, 실드(Shield) 1라인(Optional)으로 구성되며, 방수가 되어야 하며, Screw On 타입으로 되어서 쉽게 탈부착이 가능하도록 되어 있다. 중간에 장비를 추가 연결하려고 할 경우 T커넥터를 이용하여 장비를 연결하고 기존의 결선상태는 그대로 유지 시킬 수 있다. <그림 2-5>에 NMEA2000 표준 케이블 및 커넥터 사진을 나타내었다. [8]



<그림 2-5> NMEA2000 케이블 및 커넥터
<Fig 2-5> NMEA2000 Standard Cables and Connectors

NMEA2000 네트워크상에서 사용하는 자체 전원은 DC 9V ~ 16V이며, 단일 접점의 그라운드(Ground)를 처리하도록 되어 있다. 데이터 라인의 신호 레벨은 2개의 신호라인(CAN+:3.5V, CAN-:1.5V) 사이가 2V일 경우 Dominant('1')상태, 2개의 신호라인(CAN+:2.5V, CAN-:2.5V) 사이가 0V일 경우 Recessive('0')상태로 규정하고 있다. [9]

NMEA2000 네트워크는 장비와 네트워크가 서로 AC 및 DC적으로 절연이 되도록 규정하고 있다. <그림 2-6>에 NMEA2000 표준에서 권고하는 네트워크 인터페이스 방식을 소개하고 있다.



<그림 2-6> NMEA2000 절연 네트워크 인터페이스
<Fig 2-6> Isolated Network Interface

2) 데이터 링크 계층 (Data Link Layer)

데이터 링크 계층은 ISO 11783-3 국제 표준으로 제정된 SAE J1939 프로토콜

을 기본 구성으로 하여 설계되어 있다. 데이터 링크 계층을 크게 MAC부분과 LLC부분으로 나눌 수 있는데, MAC 부분은 CAN Ver.2.0B를 따르며, LLC 부분은 NMEA2000에서 규정한 부분을 따르게 되어 있다. [10]

MAC(Media Access Control) 부분은 네트워크상의 버스의 접근, Frame 전송 기능 및 전송 에러 검출기능을 수행한다. LLC(Logical Link Control) 부분은 CAN 필드 처리, 주소 관리 기능, 표준 데이터 타입과 메시지 정의, 메시지 분할기능을 지원하는 흐름제어, 네트워크 에러 보고 기능을 수행한다. LLC 메시지는 네트워크 관리, 데이터, 상태, 명령에 대한 메시지, 요구/응답 메시지, 확인(Acknowledge) 메시지 등으로 제공된다.

3) 네트워크 계층 (Network Layer)

네트워크 계층은 라우터 및 게이트웨이를 위한 계층으로 분리되어 있다. 이 부분은 현재 명확하게 규정은 되어 있지 않으며, 추후에 규정될 것으로 보인다. 이 계층에서는 네트워크 세그먼트 결합, 네트워크 주소 공간 확대, 네트워크 길이 확장 등의 기능을 수행하게 될 것이다.

4) 응용 계층 (Application Layer)

응용 계층에서는 장치의 이름을 명명하는 기능, 장치의 설정 데이터 제공 기능, 제조사별 개별 데이터 허용, 데이터 그룹의 정의, 표준 데이터의 표현 등을 규정한다. 장치의 이름을 정의하는 기능에서는 기능 코드, 인스턴스 번호, 제조사, 장치 모델 등을 정의하도록 되어 있으며, 장치의 설정 데이터 제공하는 부분에서는 인스턴스 번호, 위치, 필드 노트 등을 정의하도록 되어 있다.

데이터 그룹의 정의 부분에서는 데이터 그룹 번호(DGN : Data Group Number), 데이터 정의(위치, 시간, 이름,...), 파라미터(Latitude, Longitude,

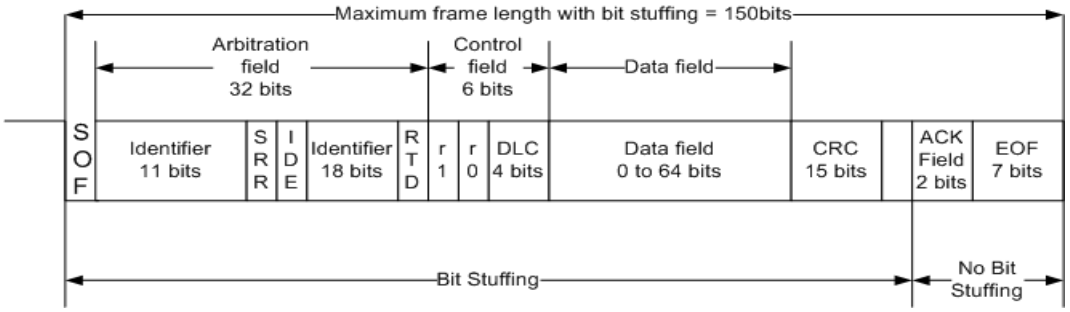
...), 우선순위, 확인 요청, 데이터 경신 주기 등을 정의하도록 되어 있다.

표준 데이터의 표현부부에서는 Character형은 8/16 bit로 규정하였으며, Integer형은 8/16/32 bit의 Signed 및 Unsigned형으로 규정하였으며, Floating Point형은 32/64/80 bit로 정의되어 있다.

2.2.2 NMEA2000 프로토콜 문장

1) NMEA2000 문장 구조

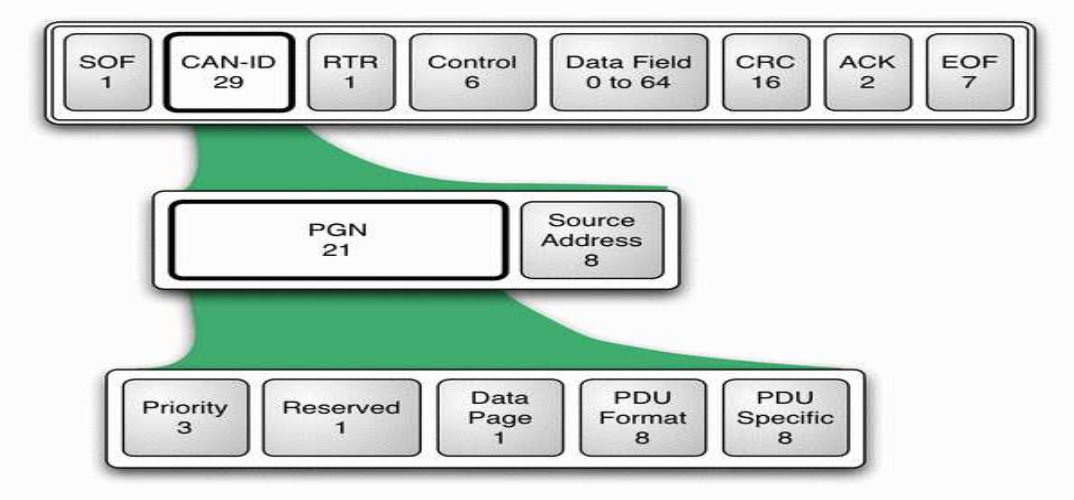
NMEA2000은 CAN Ver.2.0B Extended Frame 구조를 이용하여 <그림 2-7>와 같이 구성이 된다. Identifier 29 bit를 이용하여 각 장치들을 식별하도록 되어 있다. [11]



<그림 2-7> CAN Ver.2.0B 확장 프레임 구조
<Fig 2-7> CAN Ver.2.0B Extended Frame Structure

CAN ID부분의 29 bit에 대해서 자세히 나누어 보면 <그림 2-8>과 같이 이루어지며, PGN(Parameter Group Number) 21 bit와 SA(Source Address) 8bit로 구성이 된다. PGN은 다시 우선순위(Priority) 3 bit, DP(Data Page) 1 bit, PF(PDU Format) 8 bit, PS(PDU Specific) 8 bit로 구성되게 되어 있다.

문장에서 데이터가 8 Byte 이상인 경우, 분할 기능을 이용하여 분할 전송되게 되어 있으며, 첫 번째 Byte는 분할 번호를 사용하게 되어 있으며 최대 255개까지 하여 1,785 Bytes의 데이터 전송 가능한 다중 패킷 모드를 사용하게 되어 있다. [12]



<그림 2-8> CAN ID 영역
<Fig 2-8> CAN Identification Field

우선순위는 0이 가장 높은 우선순위를 의미하며, 네트워크의 중재(Arbitration)하는 동안에 이용되게 된다. DP(Data Page)는 향후에 PG(Parameter Group)가 늘어날 경우 PF(PDU Format)의 확장을 위해 존재한다.

PF(PDU Format)는 2개의 형식을 제공하며, PDU 1 Format은 PF값이 0 ~ 239까지이며, PDU 2 Format은 PF값이 240 ~ 255까지 이다. PDU 1의 경우는 노드 우선 전송으로 PS의 값을 상대의 주소로 표시하며 PS의 값이 255일 경우 Broadcast를 의미하며, PDU 2의 경우는 메시지 우선 전송으로 모든 노드에 전달하게 되며, 송신자는 수신자가 누구인지 모르고 전송을 하게 된다. PS(PDU Specific)는 PF(PDU Format)에 의해서 결정지어지는 부분으로 DA(Destination

Address) 또는 GE(Group Extension)의 역할을 하게 된다.

SA(Source Address)는 메시지를 보내는 장치 노드의 주소이며, 모든 노드는 네트워크상에서 유일한 주소를 갖는다.

2) NMEA2000 문장 전송 방법

NMEA2000에서는 3가지의 데이터 전송 방법(Single Frame, ISO Multi-Packet, NMEA Fast-Packet)이 있으며, 각 방법들은 서로 다른 사양과 성능을 갖고 있다. <표 2-3>에 전송 방법들을 비교하여 나타내었다.

<표 2-3> NMEA2000 전송 방법 비교

Single Frame	Multi Packet	Fast Packet
8 Byte 데이터	최대 1785 Byte 데이터	최대 223 Byte 데이터
PGN에 의해 정의된 목적지에 전달	브로드캐스트 PGN을 특정 장치에 전송	PGN에 의해 정의된 목적지에 전달
Hand-Shaking 없음	Hand-Shaking에 의한 전송 방법	Hand-Shaking 없음
전송 지연 없음	긴 시간이 필요	전송 지연 없음
모든 산업에 사용 중	모든 산업에 사용 중	NMEA2000에만 사용

SAE J1939에서 제공하는 전송 방법에서 추가적으로 Fast Packet 전송 방법이 NMEA2000 규격에서 추가되었다. 이는 최대 223 Byte 데이터까지 전송 지연 없이 목적지에 전달하기 위하여 고안된 것으로 실시간 데이터 전송을 보장하기 위한 방법이다.

2.3 TCP-IP 프로토콜

2.3.1 TCP-IP 프로토콜 개요

TCP-IP (Transmission Control Protocol - Internetworking Protocol) 프로토콜은 OSI계층 모델보다 먼저 개발되어 OSI 모델의 계층구조와 정확하게 일치하지는 않으며, 5개의 계층(물리층, 데이터링크층, 네트워크층, 전송층, 응용층)으로 구성되며, 처음 4개의 계층은 OSI 모델의 4개의 계층과 일치하는데, 각각 물리적인 표준, 네트워크 인터페이스, 네트워크 간 상호연결, 그리고 전송기능을 제공한다. 그러나 OSI 모델의 상위 3개의 계층은 TCP-IP에서는 응용계층이라는 하나의 계층으로 표현된다. [13],[14]

TCP-IP는 특정 기능을 제공하는 각 모듈이 대화식으로 되어 있는 계층구조를 갖는 프로토콜이지만, 모듈들이 반드시 상호 의존적이지는 않다. OSI 모델이 각 계층에 속해 있는 기능을 나타내고 있는 반면, TCP-IP 프로토콜의 계층은 시스템의 요구에 따라 혼합되고 대응되는 상대적으로 독립적인 프로토콜이 포함되어 있다. “계층(Hierarchical)구조”라는 용어는 각 상위 프로토콜이 하나 또는 그 이상의 하위 프로토콜에 의해 지원됨을 의미한다.

1) 물리 계층(Physical Layer)과 데이터링크 계층(Data Link Layer)

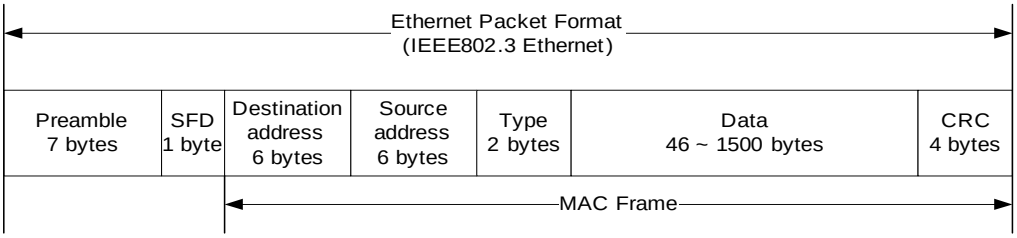
TCP-IP는 물리 계층과 데이터링크 계층에 대해서는 어떤 특정 프로토콜을 규정하지 않고, 모든 표준과 기술 프로토콜을 지원하고 있다. 물리 계층에서는 실제 장치들을 연결하기 위해 필요한 전기적, 물리적 세부 사항들을 정의한다. 예를 들어, 핀들의 배치나 전압, 전선의 명세 등이 이 계층에 포함되며, 허브나 리피터가 물리계층의 장치이다.

데이터 링크 계층에서는 Point to Point간 신뢰성 있는 전송을 보장하기 위한 계층으로 CRC기반의 오류제어와 흐름제어가 필요하다. 네트워크 위의 개체

들 간 데이터를 전달하고, 물리 계층에서 발생할 수 있는 오류를 찾아내고, 수정하는데 필요한 기능적, 절차적 수단을 제공한다. 대표적으로 이더넷(Ethernet) 프로토콜을 들 수 있다.

• ETHERNET (IEEE 802.3)

이더넷 프로토콜은 CSMA/CD (Carrier Sense Multiple Access/Collision Detection) 방법을 이용하여 전송하면서 충돌을 감지하는 방식을 사용하며, MAC 주소라는 고유의 48 bit 주소 체계를 이용하여 각 장치를 구분한다.



<그림 2-9> 이더넷 패킷
<Fig 2-9> Ethernet Packet

<그림 2-9>에 이더넷 패킷을 표현하였으며, 실제 Data 처리를 위해서 사용하는 부분은 MAC Frame부분만을 취급하게 된다.

2) 네트워크 계층 (Network Layer) : IP, ARP, ICMP,

여러 개의 노드를 거칠 때마다 경로를 찾아주는 역할을 하는 계층으로 다양한 길이의 데이터를 네트워크들을 통해 전달하고, 그 과정에서 전송 계층이 요구하는 서비스 품질(QoS)를 제공하기 위한 기능적, 절차적 수단을 제공한다.

네크워크 계층에서는 논리 주소를 지정하여 데이터의 흐름을 결정하는 경로배정 기능을 수행하며, 에러 제어 및 흐름 제어를 제공한다.

네트워크 계층에 해당하는 인터넷 프로토콜(IP)은 4개의 지원 프로토콜(ARP, RARP, ICMP, IGMP)을 포함하고 있다. 이 중에서 IP, ARP, ICMP에 대해서 설계 구현을 하였다.

• IP (Internet Protocol, RFC791, RFC919, RFC922)

TCP-IP 프로토콜에서 사용하는 전송 메커니즘으로 신뢰성을 제공하지 않는 비연결형 데이터그램 프로토콜이다. IP는 데이터그램이라는 패킷 안에 데이터를 전송한다. 각 데이터 그램은 개별적으로 전송되며 데이터 그램은 서로 다른 경로로 보내질 수 있으므로 순서대로 도착하지 않거나 중복되어 도착할 수도 있다.

IP 패킷에 대한 처리 과정에서 Protocol에 대하여 확인을 한 후에 상위 계층의 프로토콜이 ICMP, UDP, TCP등 어떤 것인지 판단할 수 있다.

<표 2-4> IP 프로토콜의 문제점

신뢰성이 없고 비연결형 데이터그램 전달 제공
최선의 노력 전달 서비스 - 보장은 없음
오류 제어와 지원 메커니즘이 없음
오류 보고 및 오류 수정 기능이 없음
호스트와 관리 질의를 위한 메커니즘이 없음

<표 2-4>에 IP 프로토콜의 문제점을 나열하였으며, 이 문제점들을 해결하기 위해서 추가적으로 이용되는 프로토콜이 ICMP 프로토콜이다. 아래에서 ICMP에 대해서 간략히 소개할 것이다.

• ARP (Address Resolution Protocol, RFC826)

인터넷상에서 사용하는 주소는 논리 주소(IP Address)와 물리 주소(MAC Address)로 구분할 수 있는데, 이 주소 중에 하나만 알고 있는 상태에서 프로토콜을 이용하여 다른 하나의 주소를 알아낼 수 있다. 이중에서 IP주소를 이용하여 MAC주소를 알아내는 방법을 제공하는 프로토콜이 ARP이다.

ARP 패킷의 구조에서 Target Hardware Address의 내용을 NULL로 하여 ARP 요청을 하면, ARP 응답하는 호스트 측에서 호스트의 MAC 주소를 추가하여 Sender 필드에 채워 넣어서 응답하게 된다. 이렇게 하여 요청하였던 호스트는 수신한 패킷에서 Sender 필드의 내용을 확인하여 상대의 물리 주소를 확인 할 수 있다.

• ICMP (Internet Control Message Protocol, RFC792, RFC950)

<표 2-4>에 나열한 IP 프로토콜의 문제점을 보완하기 위해 설계된 프로토콜이다. 메시지 유형은 크게 오류 보고 메시지와 질의 메시지로 구분이 된다.

ICMP 메시지 유형에 따라서 Type 영역의 값이 다르게 설정이 된다. 에코 요청의 경우 0x8h 값이 설정되며, 에코 응답의 경우 0x0h 값이 설정된다.

① 오류 보고 메시지

IP 패킷 처리 도중 발견된 문제를 보고하는 기능으로 목적지 도달 불가능, 발신지 전송 억제, 시간 경과, 매개 변수 문제, 경로 재지정 기능 등을 수행

② 질의 메시지

다른 호스트로부터 특정 정보를 획득하기 위해 사용하는 기능으로 에코요청과 응답, 타임스탬프 요청과 응답, 왕복 시간 계산, 주소 마스크 요청과 응답, 라우터 요청과 응답 기능 등을 수행하며, 대표적인 예가 PING 테스트이다.

이 중에서 주로 사용이 되는 PING 테스트 기능인 에코요청과 응답에 대한 기능만 우선 설계하여 지원 하였다.

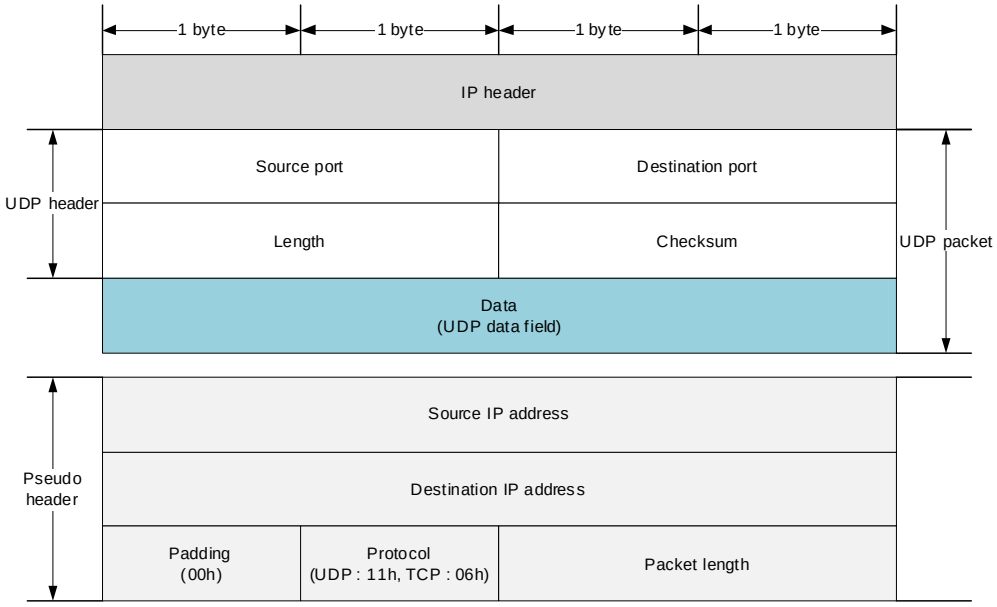
3) 전송 계층 (Transport Layer) : UDP, TCP [15]

양 끝단(End to End)의 사용자들이 신뢰성 있는 데이터를 주고받을 수 있도록 해 주어, 상위 계층들이 데이터 전달의 유효성이나 효율성을 생각하지 않도록 해준다. 시퀀스 넘버 기반의 오류 제어 방식을 사용하며, 특정 연결의 유효성을 제어하고, 일부 프로토콜은 상태 개념이 있고, 연결 기반이다.

TCP-IP에서 전송 계층은 TCP와 UDP의 두 가지 프로토콜로 대표된다. IP는 호스트-호스트 연결 프로토콜로서 물리적으로 한 장치에서 다른 장치로 패킷을 전송한다. UDP와 TCP는 한 프로세서로부터 다른 프로세서로 메시지를 전달할 것을 책임지는 전송 프로토콜이다. 프로세서 고유의 주소를 지정하는 서비스 지점 주소(Port Number)를 제공하며, 패킷의 분할 및 재조립 기능을 담당한다.

• UDP (User Datagram Protocol, RFC768)

프로세서 대 프로세서 통신 방식으로 포트번호를 이용하여 통신이 이루어지며, 비연결형으로 최소한의 오류 제어 메커니즘인 검사합 오류 제어 기능만을 수행하는 신뢰성이 없는 전송 프로토콜이다. 데이터 순서를 유지하지 않으며(흐름 제어 없음), 중복/분실의 가능성이 있는 프로토콜이다.



<그림 2-10> UDP 패킷 구조

<Fig 2-10> UDP Packet

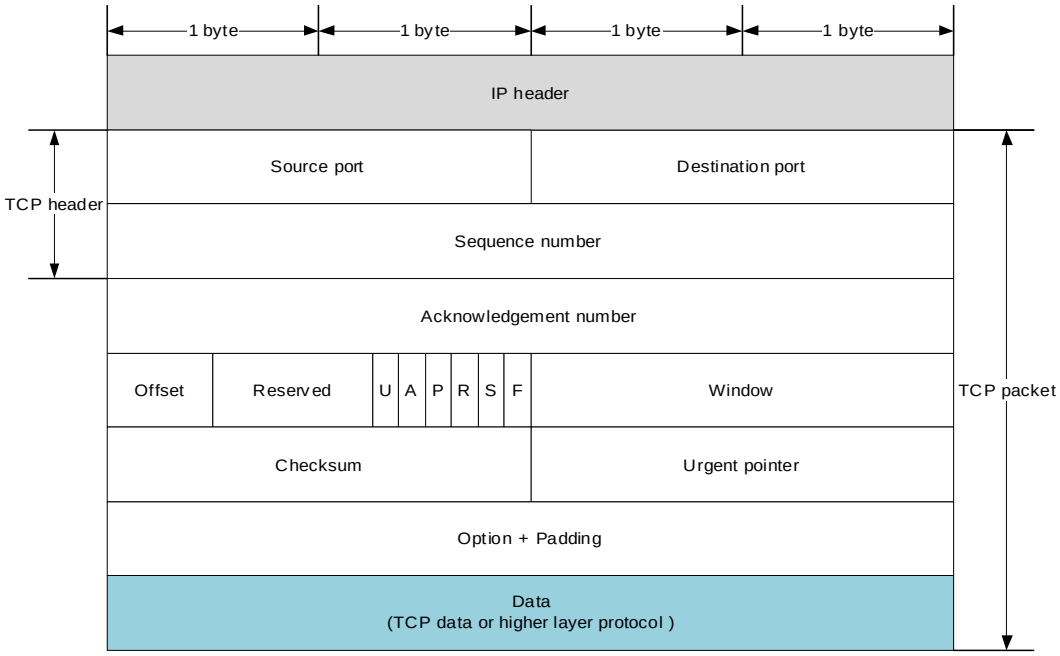
<그림 2-10>에 UDP 패킷의 구조를 나타내었으며, 윗부분이 실제 UDP 패킷이다. 그리고 아래의 Pseudo Header 부분은 검사합(Check-Sum) 기능을 위하여 임시 헤더를 추가로 사용하게 된다.

• TCP (Transmission Control Protocol, RFC675, RFC793)

신뢰성이 있는 스트림 전송 포트-포트 프로토콜이다. 스트림이라는 용어는 연결지향을 의미하며, 데이터를 전송하기 전에 종단 간에는 연결이 설정되어야 한다는 것이다.

각 송신종단에서 TCP는 메시지를 작은 데이터 단위로 나누어 세그먼트라는 프레임에 넣어서 각 세그먼트는 확인응답 수신한 후의 재 정렬을 위한 순서번호를 포함한다. 수신종단에서 TCP는 데이터그램이 들어오는 대로 모아 순서번

호에 따라 메시지를 재 정렬한다.

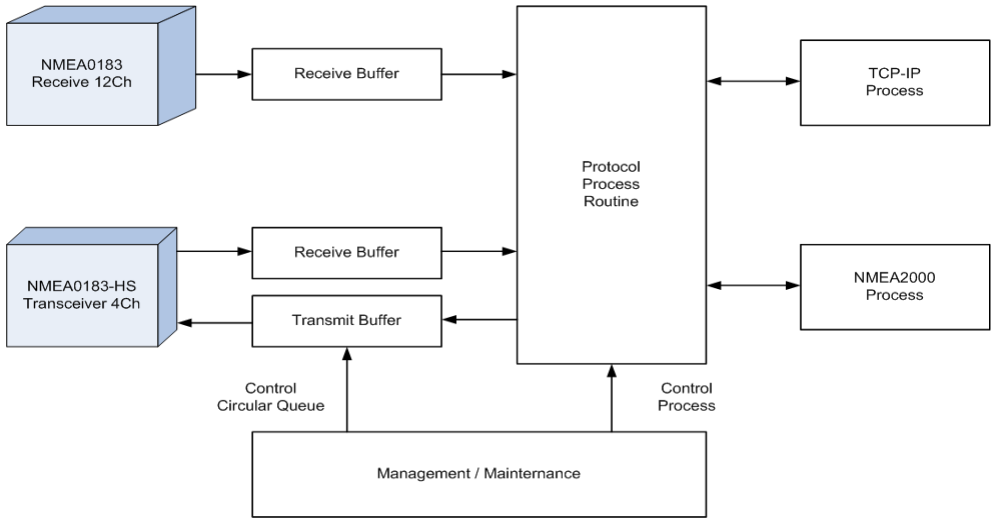


<그림 2-11> TCP 패킷 구조
<Fig 2-11> TCP Packet

<그림 2-11>에 TCP 패킷의 구조를 나타내었으며, Sequence Number와 Acknowledgement Number를 이용하여 데이터 전송에 있어서 흐름제어를 하게 된다. TCP-IP에서 일반적으로 사용하는 방식으로는 Sliding-Window 방식을 이용하는데, 이 경우 많은 메모리 공간을 필요로 하게 되는데, 특히 중간에 패킷이 분실되거나 하여 재전송을 하여야 하는 경우, 이전 전달하였던 패킷들에 대하여 모두 저장을 하고 있어야 하기 때문이다.

제 3 장 선박용 통신 프로토콜 스택 설계 및 구현

선박용 통신 프로토콜 스택을 구현한 시스템의 전체적인 데이터 흐름은 <그림 3-1>과 같이 구성하였다.



<그림 3-1> 시스템 데이터 흐름도

<Fig 3-1> Data Flow Diagram in System

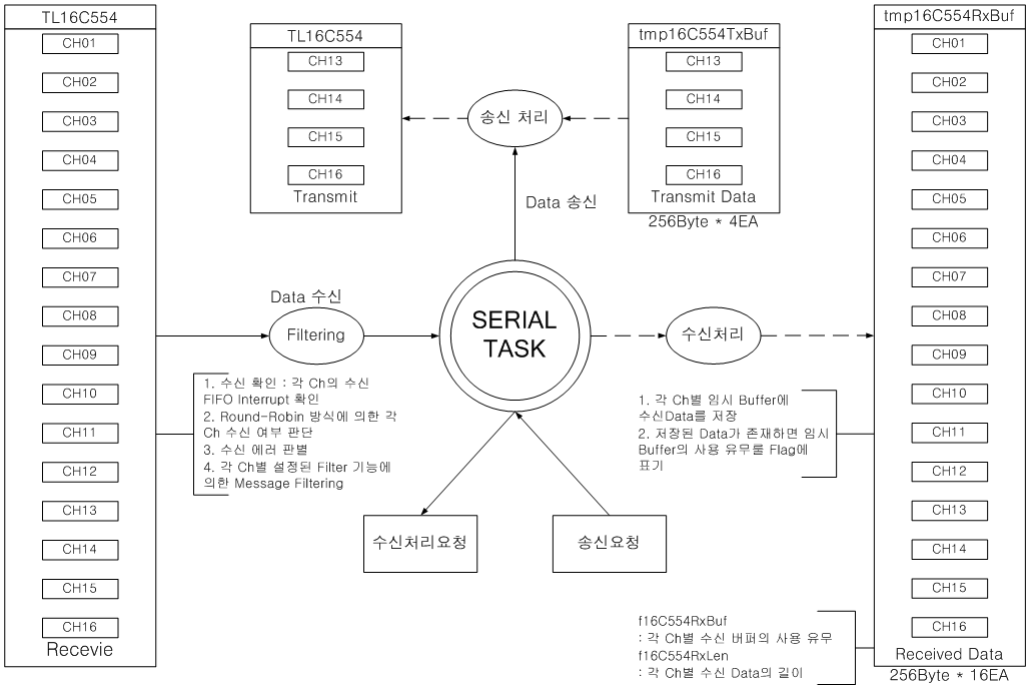
전체 시스템은 uCOS-II라는 RTOS를 이용하여 TASK를 관리하도록 설계하였으며, NMEA0183 메시지에 대한 송·수신 TASK, TCP-IP 메시지에 대한 송·수신 TASK, CAN(NMEA2000) 메시지에 대한 송·수신 TASK들을 생성하여 세마포어를 이용하여 관리하도록 하였다. 그리고 각 송·수신 TASK에서 메시지의 송·수신이 발생하여 패킷의 완성 또는 타임아웃에 의한 송·수신 종료가 발생하면, 버퍼 관리 TASK에게 Event를 생성하여 데이터를 처리하도록 되어 있다.

추가적으로 시스템의 펌웨어(Firmware)에 대한 업그레이드를 용이하게 하기 위하여 부트로더(Boot-Loader)를 제작하였으며, RS232와 TCP-IP를 이용하여 펌

웨어 업그레이드가 가능하다. 이때는 YMODEM을 지원하도록 되어 있으며, PC의 Windows환경에서 기본적으로 제공하는 하이퍼터미널과 같은 프로그램을 이용하면 된다.

3.1 NMEA0183 프로토콜 스택 설계 및 구현

NMEA0183 프로토콜 스택은 <그림 3-2>와 같이 각 채널별로 수신 버퍼를 두었으며, 한 패킷의 수신이 완료되면, 세마포어를 이용하여 버퍼 관리 TASK에게 신호를 전달하여 데이터를 버퍼에 옮겨서 저장하고 TCP-IP 및 NMEA2000 TASK에서 데이터 전송을 수행 할 수 있도록 소프트웨어를 구성하였다.



<그림 3-2> NMEA0183 데이터 흐름도

<Fig 3-2> Data Flow Diagram in NMEA0183 Stack

NMEA0183 프로토콜 스택에서는 NMEA0183 한 패킷 단위로 수신 처리하여 검사합을 계산하여 패킷의 유효성을 판단하며, 이를 문장 형식(Sentence Formatter)을 비교하여 수신 여부를 결정하는 필터를 제공한다. 이후 NMEA0183 Parsing 알고리즘을 이용하여 문장의 유효 데이터를 추출하여 사람이 쉽게 확인 할 수 있도록 데이터를 출력해주는 기능을 제공한다.

3.1.1 NMEA0183 메시지 구조체

NMEA0183 수신을 위한 구조체를 <표 3-1>과 같이 선언하였으며, UART 수신 인터럽트에서 패킷의 상태를 확인하면서, 1 바이트 단위로 NMEA0183 수신 구조체내의 버퍼에 저장을 한다.

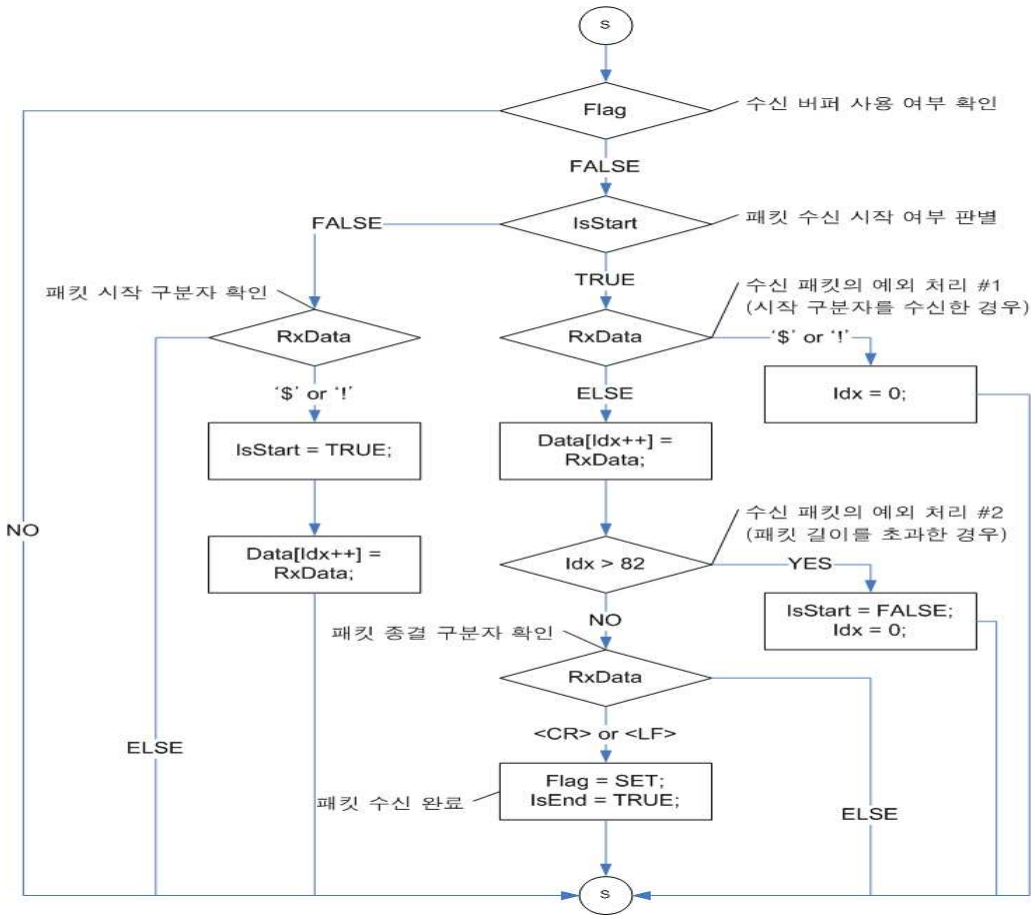
<표 3-1> NMEA0183 패킷을 위한 구조체

```
#define NMEA_SENTENCE_LENGTH          83
#define NMEA_START1                   '$'
#define NMEA_START2                   '!'
#define NMEA_CR                       ' ₩r'
#define NMEA_LF                       ' ₩n'
#define NMEA_ASTERISK                 '*'

typedef struct _NMEA0183
{
  char       Data[NMEA_SENTENCE_LENGTH];
  char       IsStart
  char       IsEnd
  char       Flag;
  char       Idx
} rxNMEA0183;
```

NMEA0183 예약어에 대해서 모두 #define하였으며, _NMEA0183 구조체의 내용을 보면, 데이터를 저장하기 위한 저장 공간으로 Data[] 버퍼를 사용하며, IsStart 변수는 패킷의 시작 문자('$' 또는 '!')의 수신 여부를 나타내며, 시작 문자를 수신함과 동시에 Data[] 버퍼에 데이터를 저장하기 시작한다. IsEnd 변수는 패킷의 종단 문자('<CR><LF>')의 수신 여부를 나타내는 것으로 main()에서 NMEA0183 패킷의 수신 여부를 확인하기 위해 사용된다. Flag 변수는 현재 Data[] 버퍼에 유효한 정보의 유무를 판단하기 위하여 사용하는 것으로 수신 인터럽트 루틴에서 IsStart 변수를 설정할 때 함께 설정하며, main()에서 패킷을 정상적으로 가져간 경우에 설정을 해제하게 되어 있다. Idx 변수는 Data[] 버퍼에 저장된 패킷의 길이를 나타낸다.

3.1.2 NMEA0183 메시지 수신 알고리즘

NMEA0183 메시지를 수신하기 위하여 UART 수신 인터럽트 서비스 루틴에서 <그림 3-3>과 같은 방법으로 수신 버퍼에 메시지를 수신하여 저장한다. 수신 여부에 대해서는 Main Routine에서 IsEnd 플래그를 확인하여 확인 할 수 있으며, 이를 이용하여 다음에서 소개할 NMEA0183 Parsing 처리를 수행하게 된다.



<그림 3-3> NMEA0183 수신 알고리즘

<Fig 3-3> Receive Process for NMEA0183

3.1.3 NMEA0183 메시지 Parsing 알고리즘

NMEA0183 메시지 수신이 완료되면, 수신 버퍼의 내용을 임시 처리 버퍼로 복사를 하고, 다음 메시지 수신을 위하여 버퍼를 비워둔다. 그리고 수신 메시지의 유효성을 검사하기 위하여 검사합(Check-sum)을 확인하여, 유효하지 않은 경우는 메시지를 비워버리도록 조치하였다. <표 3-2>에 NMEA0183의 일반적인 검사합 방법에 대해서 소개하였다.

<표 3-2> NMEA0183 검사합 계산

```
char buf[] = "$GPGLL,5057.970,N,00146.110,E,142451,A*27<CR><LF>"
unsigned char count, checksum;
count = 1;
checksum = 0;
while (buf[count] !=  '*' )
{
  checksum ^= buf[count++];
}
```

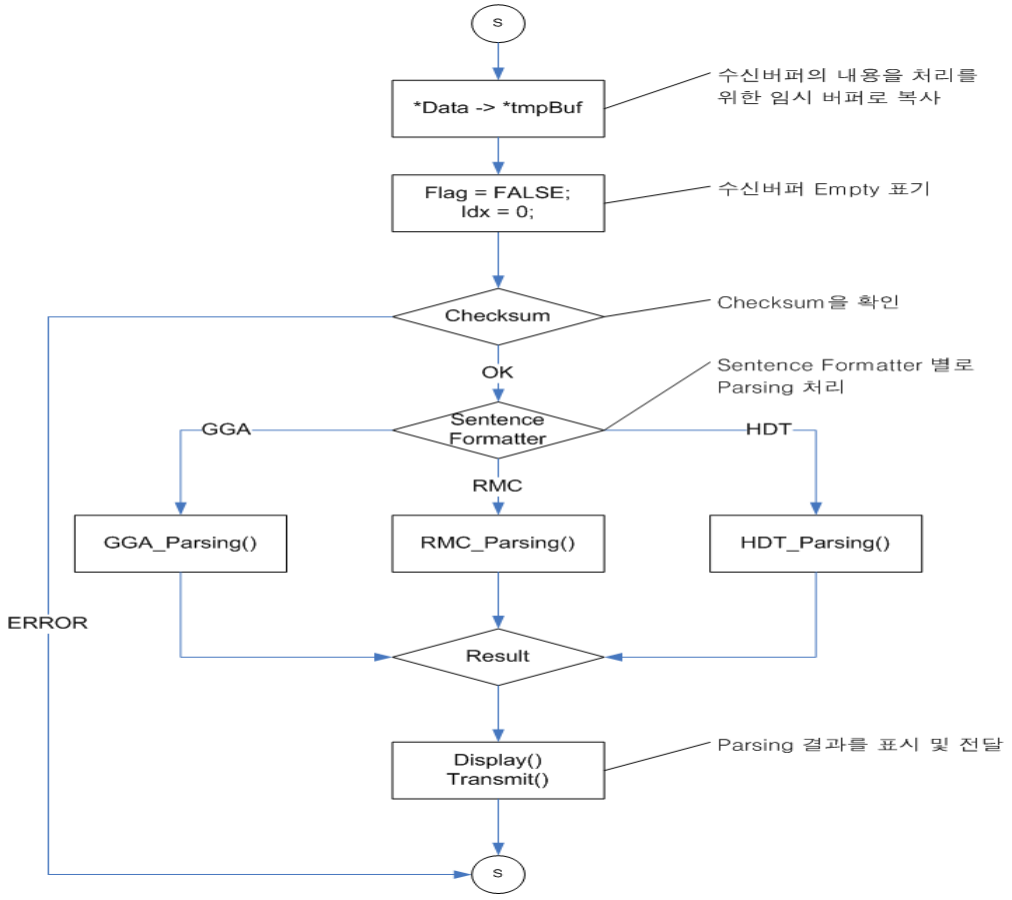
NMEA0183 메시지 수신 여부를 결정하기 위하여 문장 형식 테이블을 비교하여 각 채널별로 설정된 수신 가능 문장 형식과 일치하는 문장에 대해서는 수신을 허용하여 NMEA0183 Parsing 알고리즘을 적용하도록 되어 있다.

그림 은 NMEA0183 Parsing 처리 함수를 호출하기 위한 전단계의 플로 차트다. 메시지 구조체의 Flag 변수들을 확인하여 각 상태를 확인하여 다음을 진행하도록 되어 있다.

NMEA0183 메시지의 구조상 문장 형식에 따라서 각 필드이 내용이 다르게 구성이 되어 있다. 이에 따라 문장 형식을 확인한 후에 각 문장형식별로 별도의

Parsing 알고리즘을 적용하도록 하였으며, 그 각 처리 함수에서 각 필드의 유효한 정보들을 추출하도록 되어 있다.

이 후 메시지 저장 버퍼에 저장이 되며, TCP-IP를 통해 전송하기 위해서 TCP-IP 정송 처리 TASK에 세마포어를 전달한다. NMEA2000 전송 TASK에서는 메시지 전달을 위해서는 NMEA0183에서 추출한 유효한 정보를 이용하여 NMEA2000 패킷을 만들 수 있도록 NMEA2000 메시지 생성 함수에게 정보를 전달하도록 구성하였다.



<그림 3-4> NMEA0183 분석 알고리즘

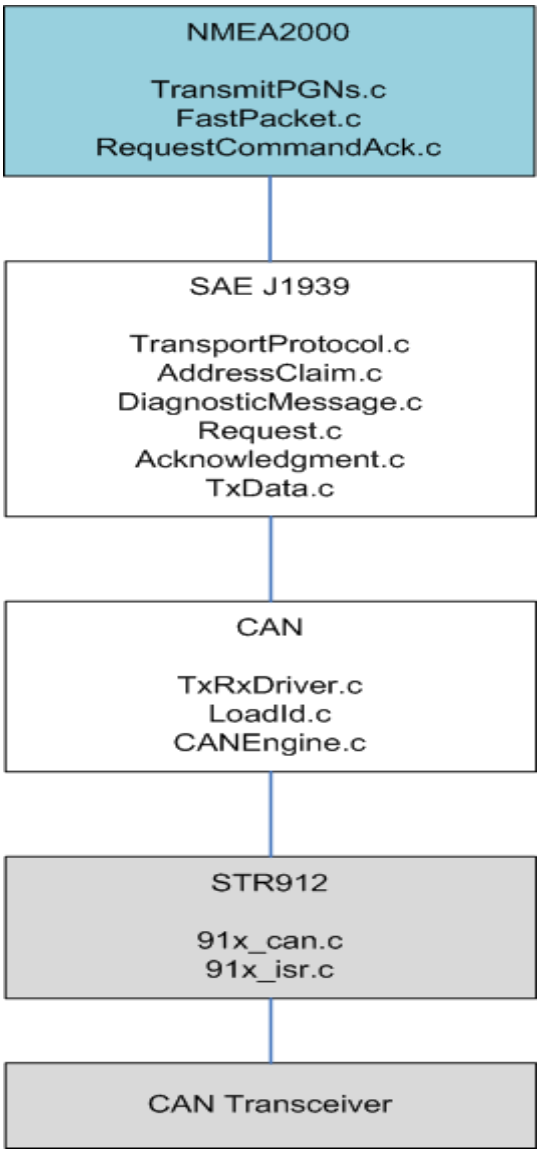
<Fig 3-4> NMEA0183 Parsing Algorithm

<그림 3-4>는 일반적인 NMEA0183의 Parsing 알고리즘으로 필드 구분자('.')와 검사합 구분자('*')를 토큰으로 사용하여 패킷에서 각 필드들을 추출하도록 되어 있다. <표 3-3>에 일반적인 NMEA0183 Parsing 알고리즘을 소개하였다.

<표 3-3> NMEA0183 분석 알고리즘

```
int count = 0;
int SizeOfToken = 0, i=0;
char seps[] = ",* ₩r ₩n";
char *token;

token = strtok( (char *)tmpBuf.Data, seps );

while( token != NULL )
{
  strcpy( RMC_Pointer[count], token );
  count++;
  i=1;

  SizeOfToken = SizeOfToken + strlen(token);

  while( tmpBuf.Data[SizeOfToken+i] == ',' )
  {
    i++;
    count++;
  }
  SizeOfToken += i
  token = strtok( NULL, seps );
}
```

3.2 NMEA2000 프로토콜 스택 설계 및 구현



<그림 3-5> NMEA2000 스택의 구조

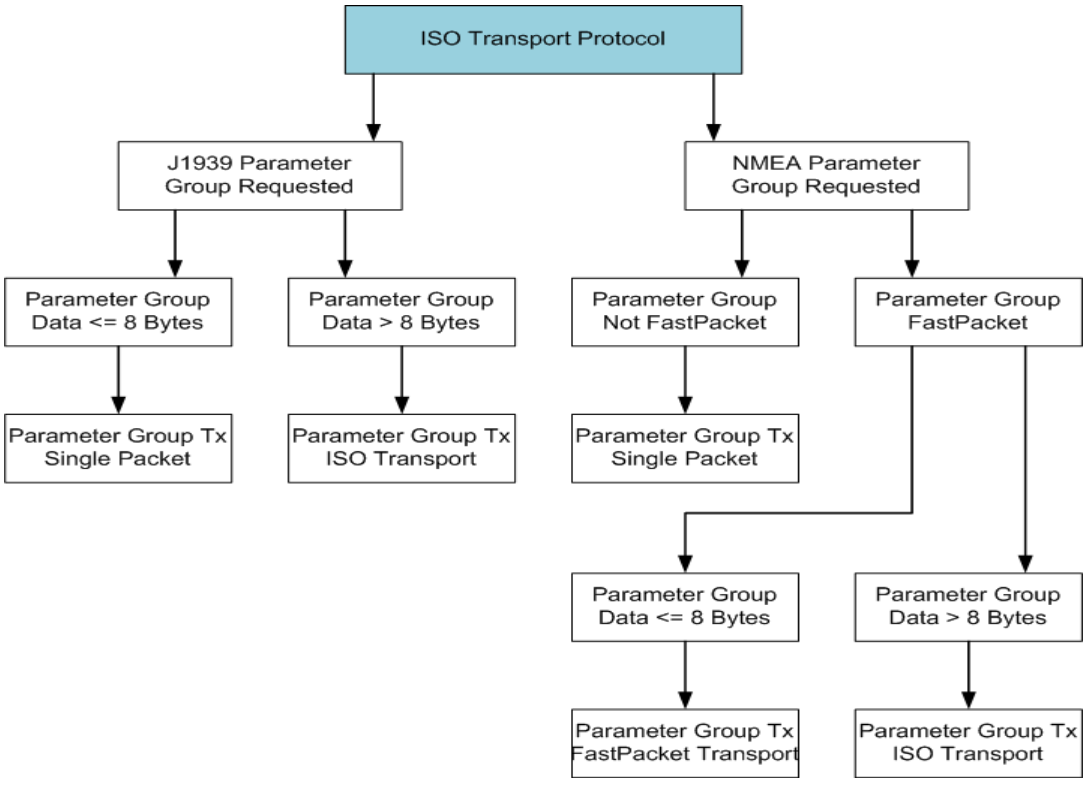
<Fig 3-5> NMEA2000 Stack

NMEA2000 Stack은 <그림 3-5>와 같은 구조로 설계하였으며, 하위 계층은 ST 사에서 제공하는 CPU 관련 API함수를 사용하여 쉽게 제어하였으며, Interrupt Service 루틴을 사용하여 송수신을 처리하였다.

그리고 Stack의 전체구조는 CAN부분, J1939부분, NMEA2000부분으로 구분하여 쉽게 접근할 수 있도록 되어 있다. 실제 사용에 있어서 NMEA2000 하위 부분의 접근없이 NMEA2000부분의 함수들만을 이용하여 NMEA2000관련 응용을 할 수 있는 구조이다.

3.2.1 NMEA2000 스택의 구조

NME2000 스택에서는 크게 2개의 구조로 나누어지는데 표준 J1939에서 제공하는 ISO Transport구조와 NMEA2000 고유의 Fast-Packet Transport구조로 나타낼 수 있다.

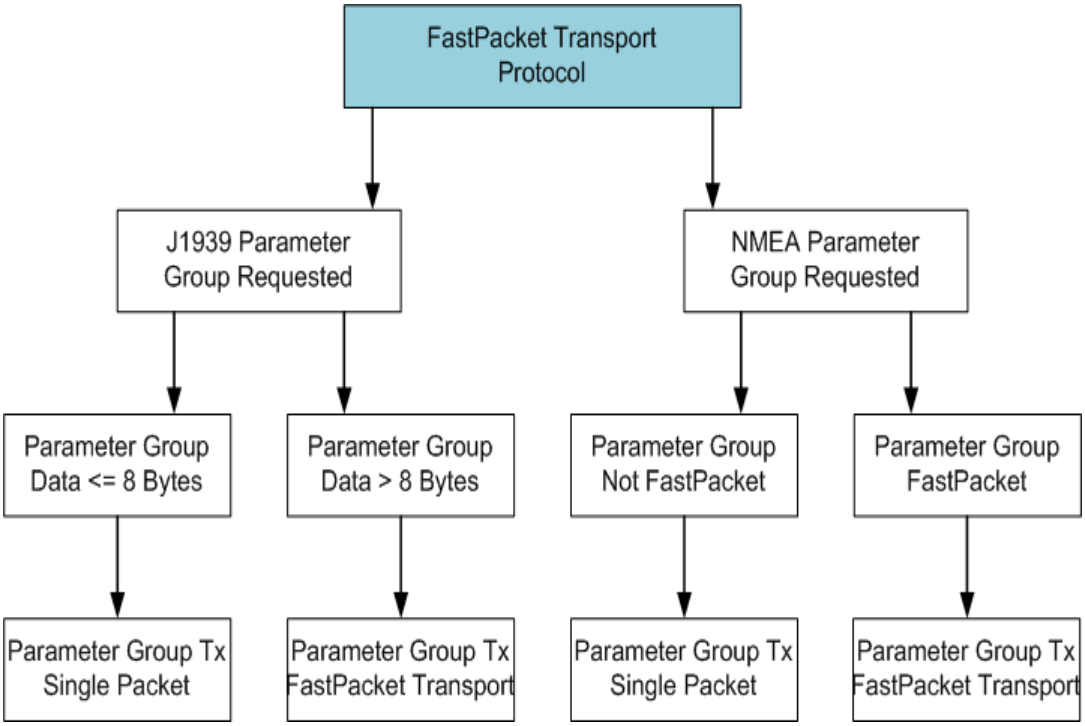


<그림 3-6> ISO 전송 프로토콜 구조

<Fig 3-6> ISO Transport Protocol

<그림 3-6>에 ISO Transport 프로토콜의 데이터 흐름을 나타내었으며, 수신된 패킷을 확인하여 메시지 유형을 판단하여서 각 메시지 유형에 따라서 처리 과정을 두어서 진행한다.

<그림 3-7>에는 NMEA2000 고유의 메시지 유형인 Fast-Packet Transport 프로토콜에 대한 처리 과정을 나타내었다.



<그림 3-7> Fast-Packet 전송 프로토콜 구조

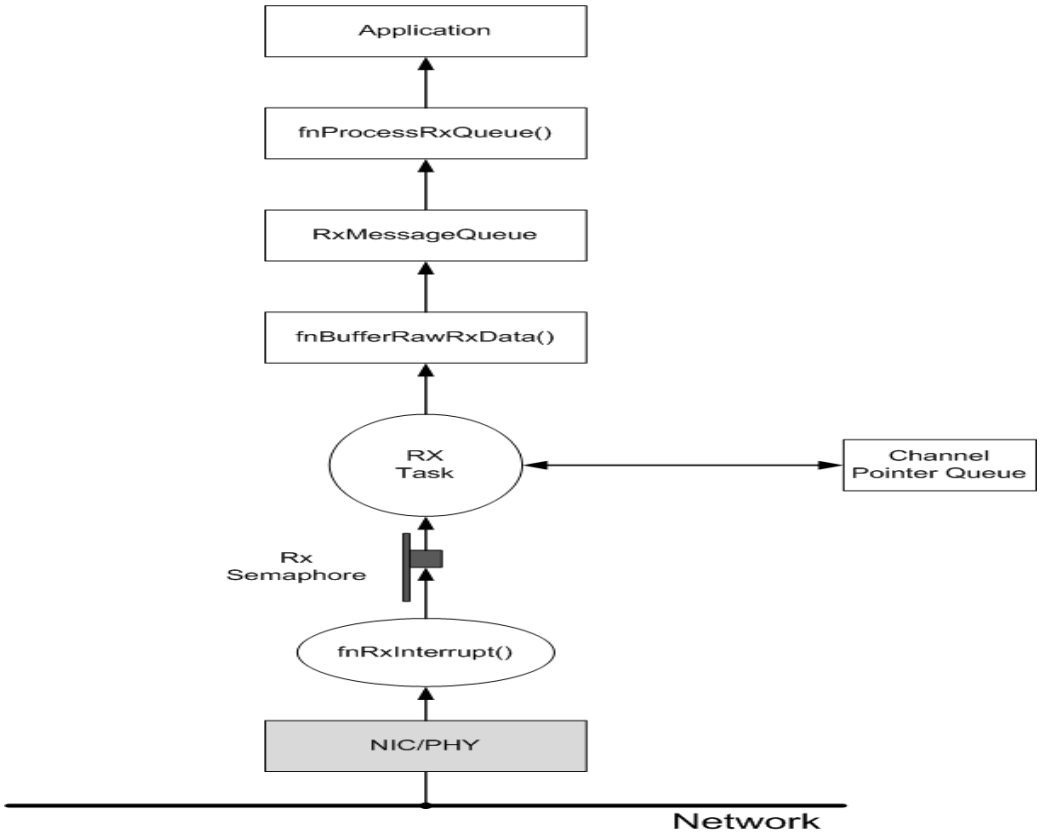
<Fig 3-7> Fast-Packet Transport Protocol

NMEA2000 스택의 2가지 구조 모두 내부적으로는 중복되는 부분이 많이 있다. 하지만 패킷의 송•수신 처리를 위해서는 수신된 패킷을 분석하여서 송신을 진행하여야 하기 때문에 이러한 구조를 가질 수밖에 없다.

3.3.2 NMEA2000 스택 구현 내용

1) NMEA2000 메시지 수신 흐름도

<그림 3-8>는 NMEA2000 패킷의 수신 과정에 대한 전체적인 흐름을 나타낸 것이다. 메인 컨트롤러에서 CAN 수신 완료 인터럽트에 의해서 fnBufferRawRxData()함수가 호출이 된다. 임시 버퍼에 수신된 CAN 데이터를 RxMessageQueue에 옮기는 작업을 수행한다. RxMessageQueue에 저장된 데이터들은 fnProcessRxQueue() 함수에서 각각 처리하도록 되어 있다. 이곳에서 각 필드의 내용을 추출하여 응용 프로그램에서 사용할 수 있게 한다.



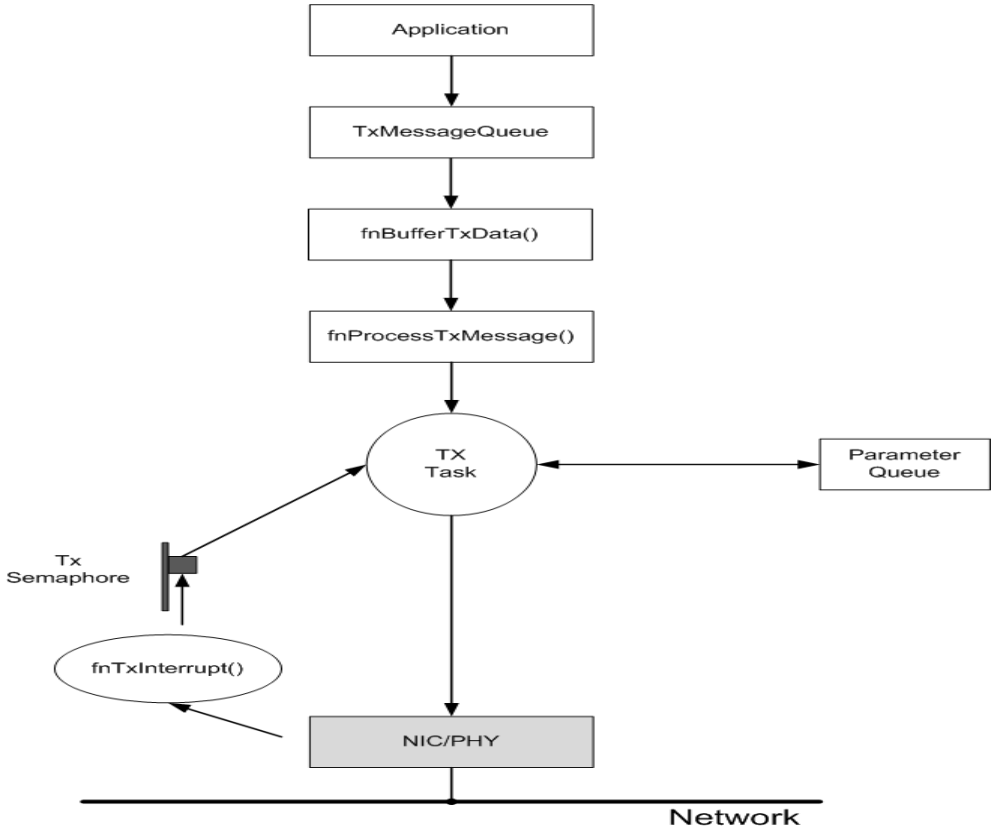
<그림 3-8> NMEA2000 수신 처리

<Fig 3-8> NMEA2000 Receiving Process

그리고 수신된 데이터를 처리 할 때 메시지 유형에 따라서 최종적으로 저장되는 구조체가 구분이 되어 있다. Single Frame의 경우 tTP_CM_SINGLE_PACKET 구조체에 저장이 되며, Multi-Packet의 경우 tTP_CM_RX 구조체에 저장이 되며, Fast-Packet의 경우 tFASTPACKET_RX 구조체에 저장이 된다. 응용 프로그램에서는 패킷의 타입을 확인하여 각 구조체에서 데이터를 가져가게 된다.

2) NMEA2000 메시지 송신 흐름도

<그림 3-9>은 NMEA2000 패킷의 송신 과정에 대한 전체적인 흐름을 나타낸 것이다.



<그림 3-9> NMEA2000 송신 처리

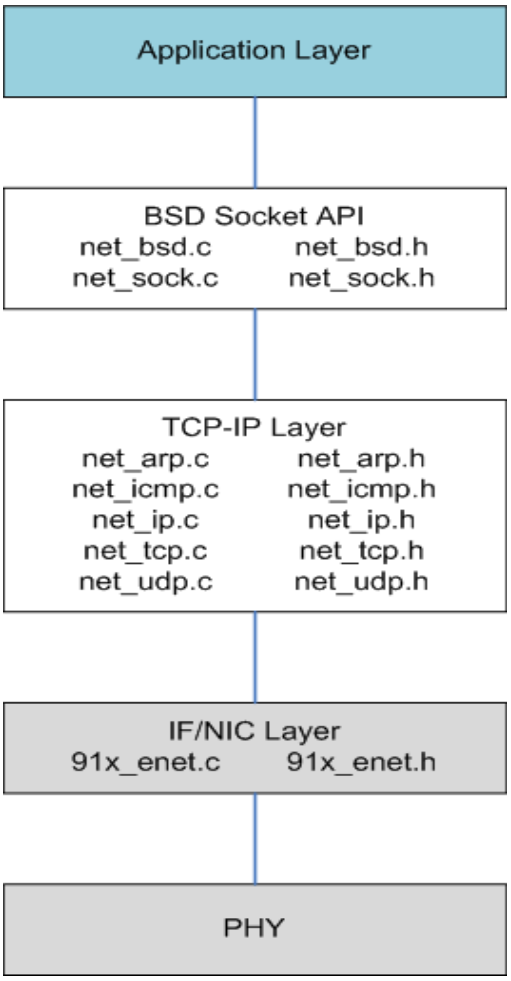
<Fig 3-9> NMEA2000 Sending Process

응용 프로그램에서 fnBufferTxData()를 이용하여 TxMessageQueue에 전달할 Data를 적재하게 된다. 그리고 fnProcessTxMessage()함수에 의해서 TxMessageQueue에 있는 메시지 유형을 판단하여 각 유형에 맞는 구조체에 데이터를 적재하게 된다. 이 후 CAN Core에 송신 요청을 하면, CAN Core에서는 송신 가능상태가 되기를 기다린 후에 CAN 버스의 상태를 확인하여 전송이 가능할 경우 메시지를 전송하게 되어 있다. 물론 Fast-Packet의 경우는 우선순위 비교에서 CAN 버스를 선점하게 됨으로 빠른 전송이 가능하게 된다.

3) 주소 요구 메시지 (Address Claim Message)

주소 요구 메시지는 CAN 버스 상에 다른 장치사이에서 주소의 충돌을 해결하는 것을 의미한다. SAE J1939 표준에 따르면 장치는 전원이 인가되면 브로드캐스트방식으로 주소 요구 메시지를 전송하게 되어 있다. 이때 주소 충돌을 해결하기 위하여 NAME 필드를 사용하게 된다. 2개의 장치가 주소 충돌이 발생한 경우 NAME 필드의 우선순위가 낮은 장치가 주소를 변경하도록 되어 있다. 이 기능에 대해서는 fnAddressClaim()함수에 구현이 되어 있다.

3.3 TCP-IP 프로토콜 스택 설계 및 구현



<그림 3-10> TCP-IP 스택의 구조

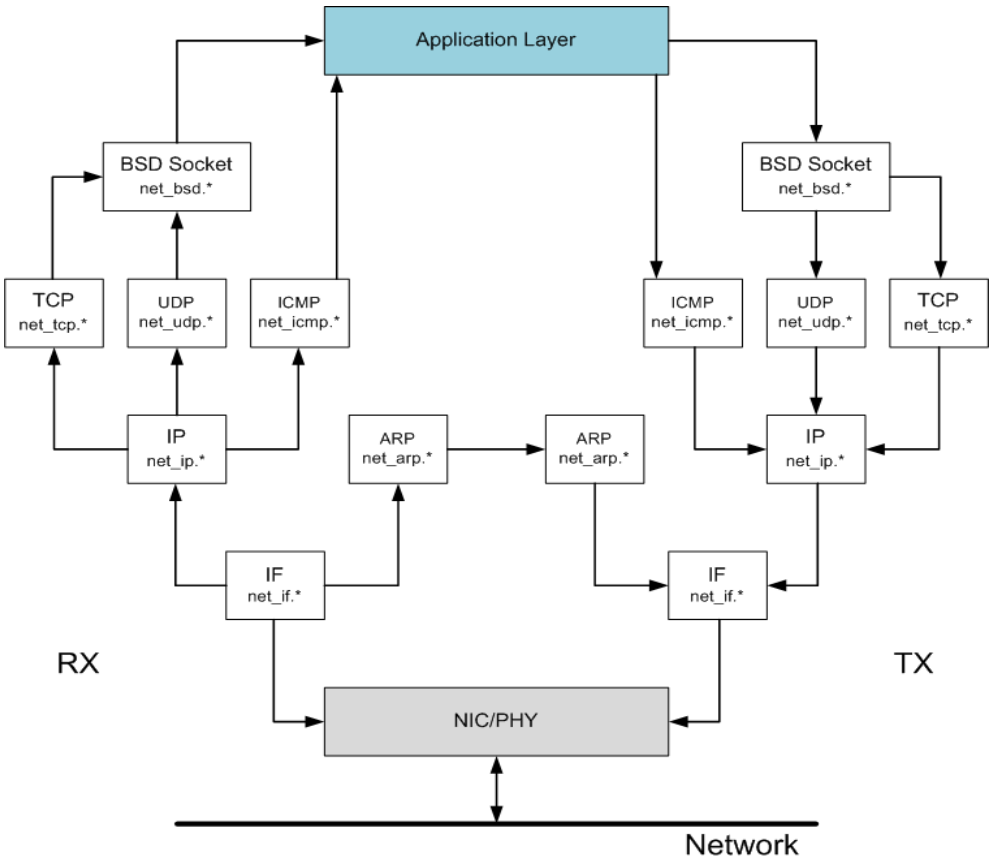
<Fig 3-10> TCP-IP Stack

TCP-IP 프로토콜 Stack은 <그림 3-10>와 같은 구조로 설계하였으며, OSI 모델의 각 계층에서 사용되는 Protocol별로 각각 파일로 생성하여 구분을 용이하도록 하였다. 하위 계층은 ST사에서 제공하는 CPU관련 API함수를 사용하여 쉽게 제어하였으며, 최적화된 메모리 버퍼 복사 알고리즘 및 DMA기능을 이용하여 최적의 속도를 제공하도록 설계하였다. 그리고 사용자의 입장에서 쉽게 접근할 수 있도록 BSD Socket API를 제공하여 이용을 용이하게 처리해 두었다. <그

림 3-10>에서 TCP-IP Layer의 내용을 모르더라도 PC환경의 Socket 프로그램을 하는 것처럼 BSD Socket API를 이용하면 되는 구조이다.

3.3.1 TCP-IP 프로토콜 전체적인 구조

TCP-IP의 전체적인 데이터 흐름은 <그림 3-11>와 같은 구조로 설계하였으며, 송신/수신의 Data 흐름은 분리하여 구성하였다.



<그림 3-11> TCP-IP 데이터 흐름도

<Fig 3-11> TCP-IP Data Flow Diagram

IF부분은 Ethernet Packet에서 MAC Frame부분을 추출하거나, IP Datagram을 MAC Frame으로 생성하여 Ethernet Core에 전달하여 Ethernet Packet을 생성하

는 부분이다.

BSD Socket 부분은 socket(), close(), shutdown(), bind(), connect(), listen(), accept(), recvform(), recv(), sendto(), send()등의 표준 BSD 4.x와 호환되는 기능을 제공하는 함수들로 구성하였다. Windows 환경에서는 Winsock에서 제공하는 함수들을 이용하여 TCP-IP관련 어플리케이션을 개발하곤 하는데, 이때 사용되는 함수들 또한 BSD Socket 호환으로 제공을 하는 부분이다.

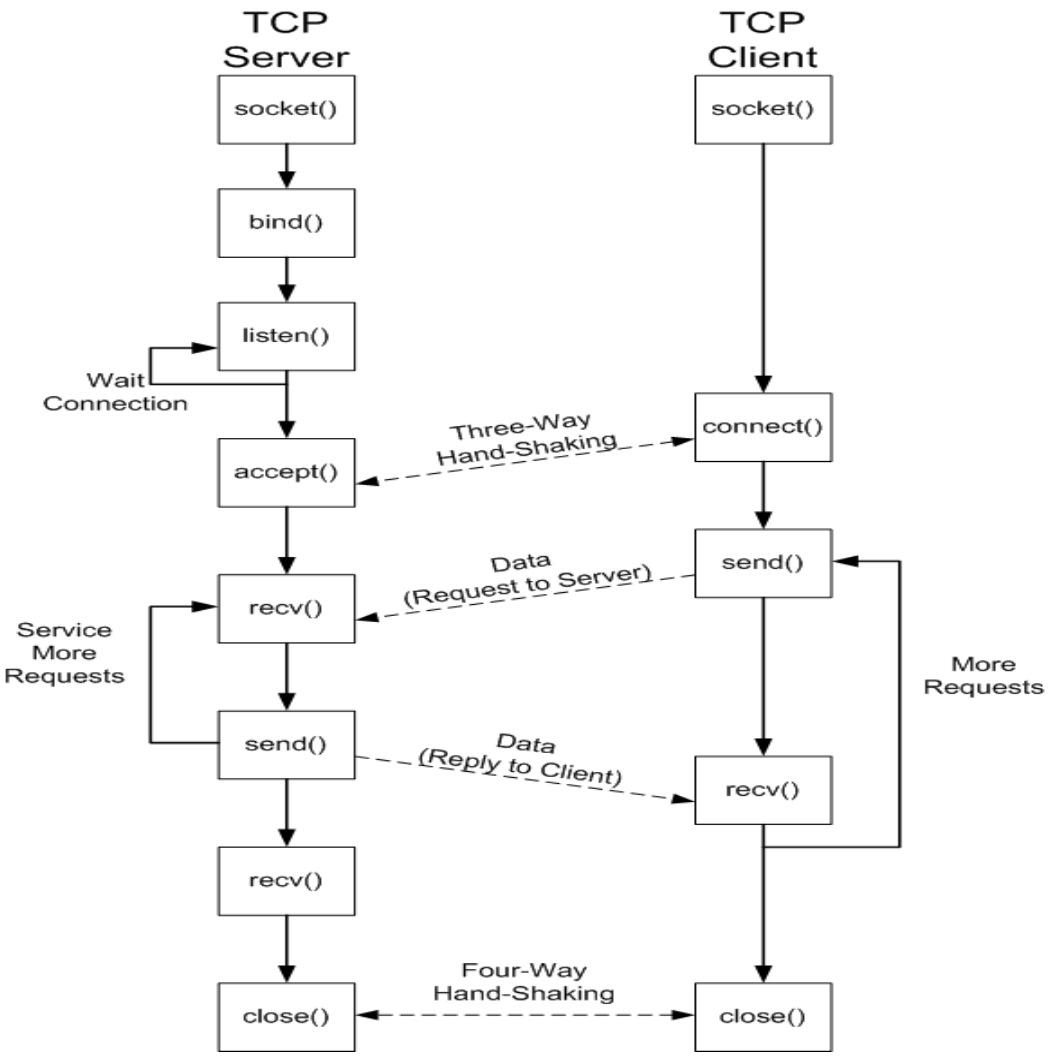
3.3.2 TCP-IP 스택 구현 내용

1) TCP 소켓의 흐름

<그림 3-12>에서는 일반적인 TCP Client-Server 응용 및 BSD 함수의 사용을 보여주는 것으로 본 논문에서도 그림과 같은 구조로 설계 및 구현하였다.

TCP-IP 프로토콜에 대한 응용을 할 경우, 물리 계층에서부터 상위 응용 계층까지의 모든 관련 함수들에 대해서 알아야 하겠지만, PC환경에서 SOCKET 프로그램을 개발하는 것처럼, 사용자는 쉽게 BSD Socket API함수들만 이용하여서 TCP-IP응용 프로그램을 개발할 수 있다. 여기서 사용된 BSD Socket API함수들은 모두 표준 규격의 것과 일치하도록 설계하였기 때문에 이용에는 별다른 어려움이 없을 것으로 판단된다.

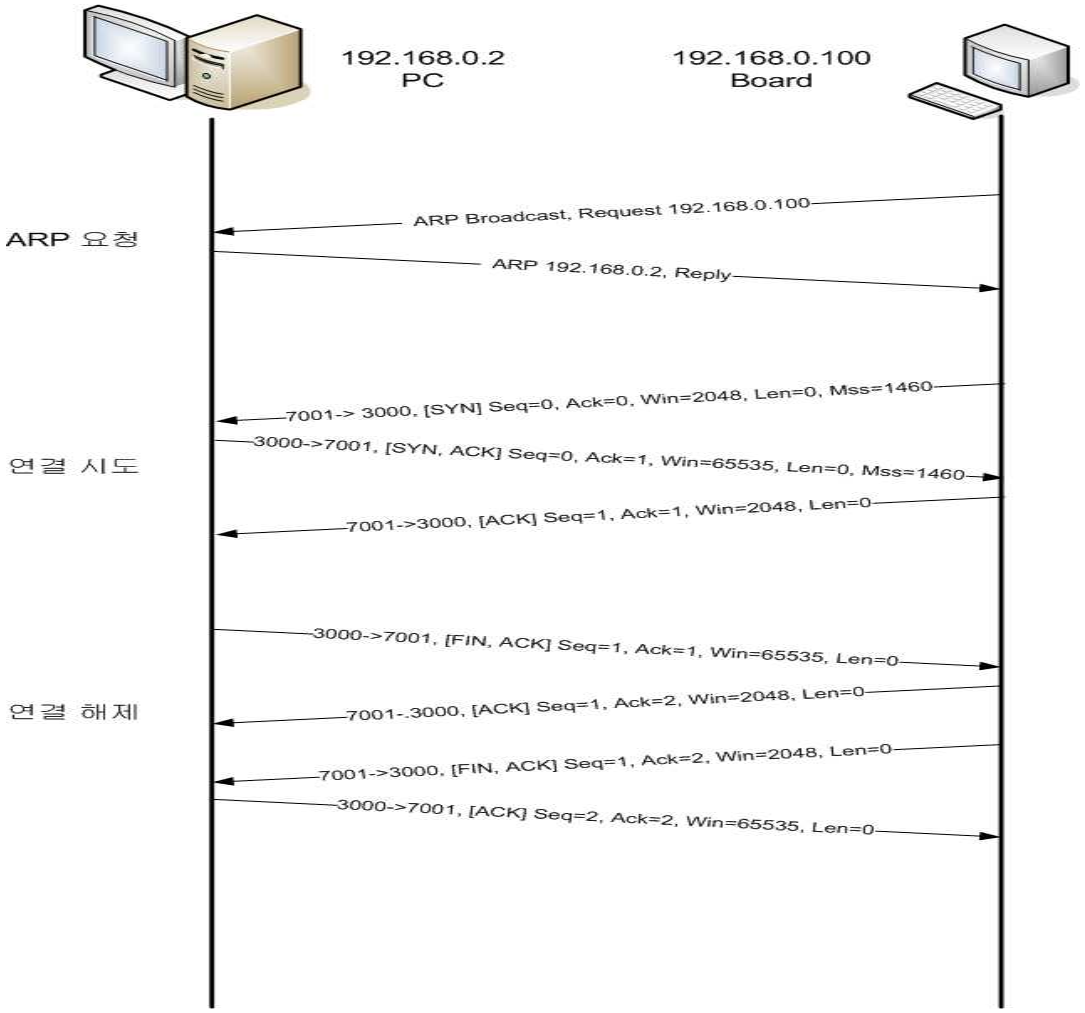
초기 Server 쪽에서는 소켓을 생성하고 포트에 할당한 후에 연결을 기다리는 대기상태에 들어가며, Client쪽에서는 소켓을 생성하고 Server에 연결 요청을 하게 된다. 이를 Server에서 연결 요청에 대한 승인을 하여 접속을 하게 되며, 서로 자료 요청과 응답을 지속하게 된다. 연결을 종료할 경우에도 연결 종료를 요하는 측에서 연결 종료 요청을 하면, 상대에서 승인을 하여 연결을 종료하게 된다.



<그림 3-12> TCP 응용

<Fig 3-12> TCP Application

초기 연결과 종료에 있어서 기본적으로 TCP에서 이루어지는 방법을 그대로 유지하였다. <그림 3-13>은 실제 구현된 보드와 PC와의 TCP를 이용한 연결·종료의 과정을 캡처한 내용을 나타낸 것이다.



<그림 3-13> TCP 연결 설정 및 종료

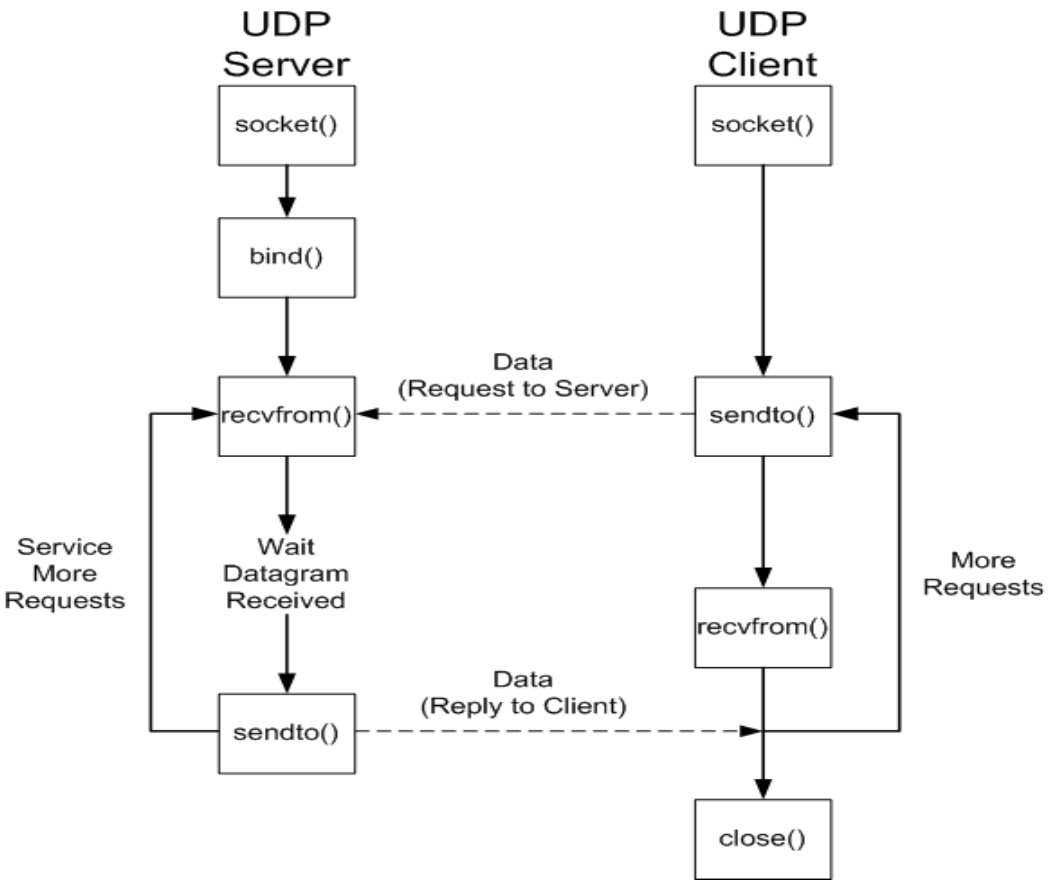
<Fig 3-13> TCP Hand-Shaking

PC 환경이 아닌 임베디드 환경에서는 하드웨어의 제약 조건 때문에 Sliding Window와 같은 흐름제어 기법을 이용하지 못하는 상황이다. 그래서 한 패킷을 전송하고 ACK를 받은 후에 다음 패킷을 전송하는 구조로 설계하였다. 만약 상대로부터 ACK를 받지 못하는 경우는 타임아웃에 의하여 연결 종료를 하게 된다.

그리고 재전송 기능에 대해서도 하드웨어의 제약 조건 때문에 바로 전에 전송한 패킷에 대해서만 재전송이 가능한 구조이다.

2) UDP 소켓의 구조 와 흐름

<그림 3-14>에서는 일반적인 UDP Client-Server 응용 및 BSD 소켓의 사용을 보여주는 것으로 본 논문에서도 그림과 같은 구조로 설계 및 구현하였다.



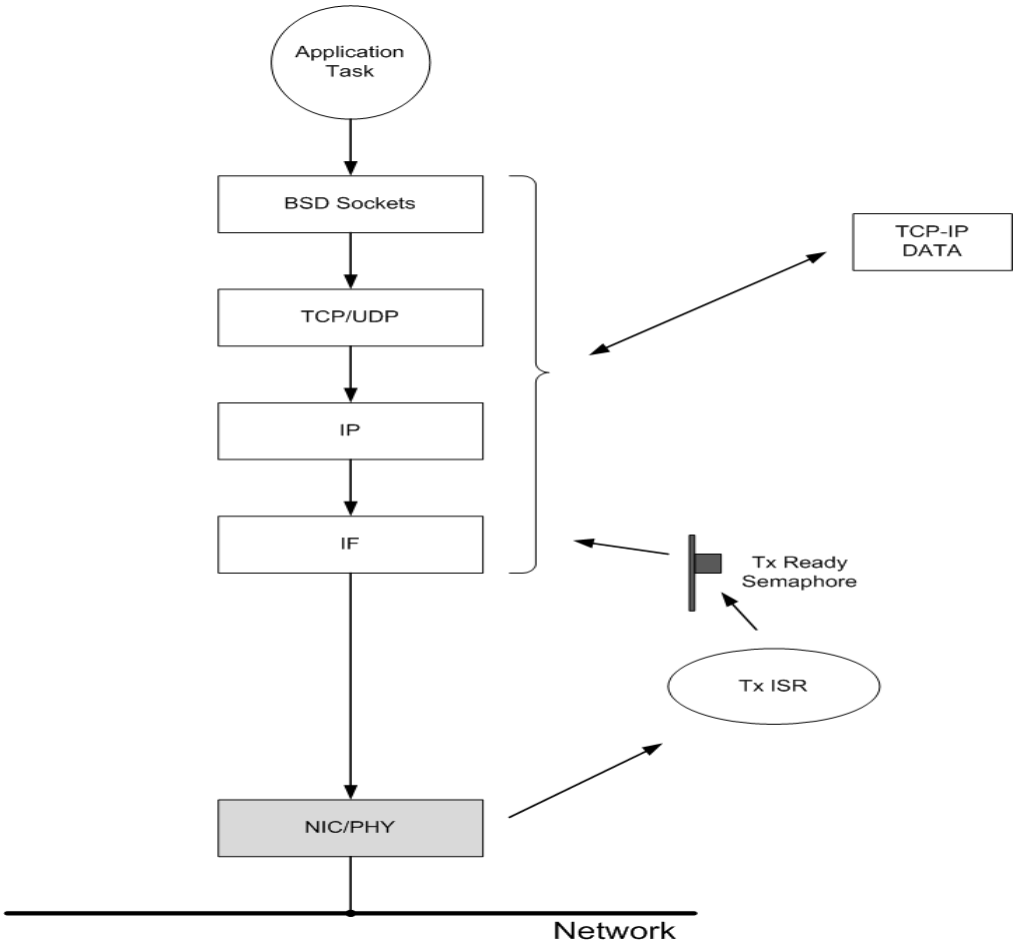
<그림 3-14> UDP 응용

<Fig 3-14> UDP Application

TCP의 응용과 다르게 UDP는 비 연결형 프로토콜이므로 연결·종료의 흐름이 불필요하다. UDP의 응용에서도 BSD Socket만 사용함으로써 쉽게 응용에 적용할 수 있다.

3) TCP-IP 패킷 송신 과정

<그림 3-15>은 구현된 TCP-IP 프로토콜 스택에서 데이터 송신 과정에 대하여 소개한 것이다.



<그림 3-15> 패킷 송신

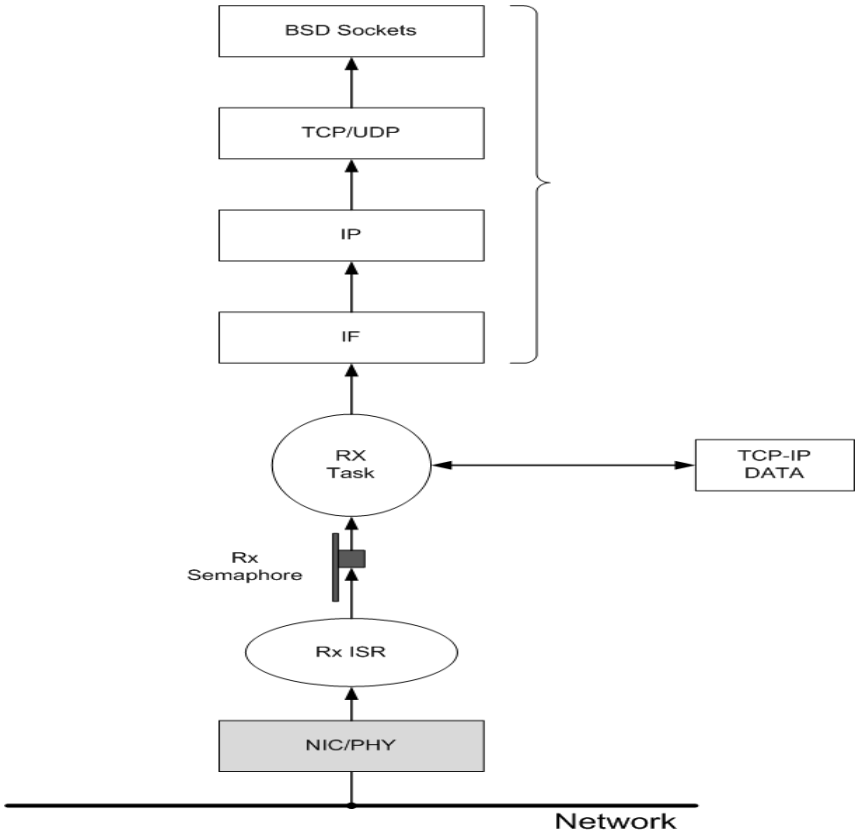
<Fig 3-15> Sending Packet

응용 TASK에서 Data 전송하기 위해서 Data를 BSD Socket을 이용해서 호출하면, BSD Socket을 통해 TCP 또는 UDP 자료 구조에 Data가 전달되어 패킷이 완성이 된다. 그 후 IP Layer에 내려가서 IP 패킷이 완성되고, 최종 전송을 위한 준비를 마치게 된다. 이때 IF Layer에서는 네트워크의 상태가 전송 가능한 상

태가 될 때까지 세마포어 대기 상태에 들어간다. 그래서 NIC가 이전 패킷의 전송이 완료되어 전송 가능한 상태가 되면, 전송 완료 인터럽트에 의해서 세마포어를 IF Layer에 전달함으로써 IF Layer에게 전송할 패킷을 요청하게 된다.

4) TCP-IP 패킷 수신 과정

<그림 3-16>은 구현된 TCP-IP 프로토콜 스택에서 데이터의 수신 과정에 대하여 소개한 것이다.



<그림 3-16> 패킷 수신

<Fig 3-16> Receiving Packet

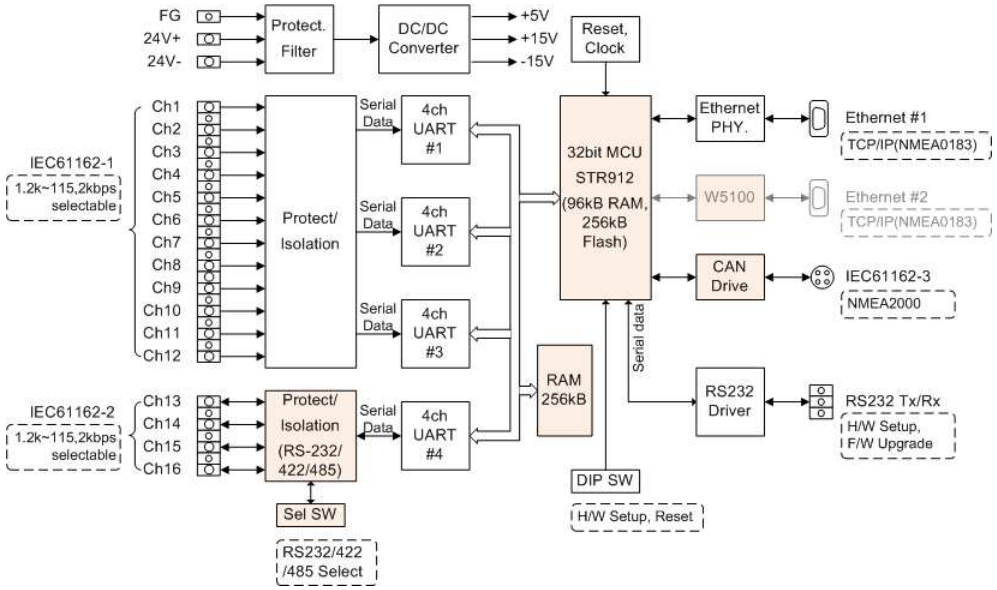
Data의 수신은 CPU 내부의 Ethernet 수신 인터럽트에 의해서 이루어지게 된다. Ethernet Core에서는 설정된 MAC 주소의 일치여부를 판단하여 Data를 수

신하면 수신 인터럽트를 발생하여 IF Layer에 수신 Data를 가져가도 세마포어를 발생하는데 이때는 바이너리 세마포어가 아닌 카운팅 세마포어를 사용한다. 이를 이용하여 IF Layer에서는 수신된 패킷의 개수를 확인할 수 있다. IF Layer에서는 Ethernet 패킷을 분석하여 상위 Layer에 전달한다.

3.4 하드웨어 설계 및 구현

3.4.1 하드웨어 구성

본 논문에서 설계한 하드웨어의 구성은 <그림 3-17>와 같이 구성하였다. 기본적으로 필요한 Ethernet Core, CAN Core 부분은 MCU 내부에 내장되어 있으며, 외부에는 물리층을 위한 Driver IC만 추가해주면 쉽게 이용이 가능하도록 되어 있다

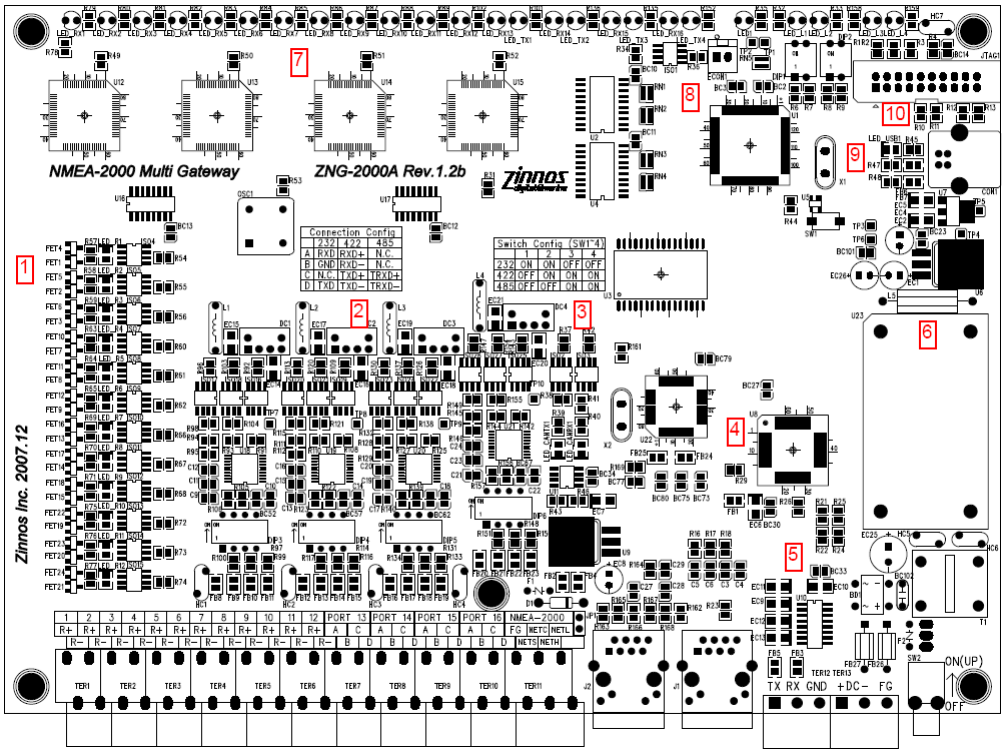


<그림 3-17> 하드웨어 구조

<Fig 3-17> Hardware Diagram

3.4.2 하드웨어 내용

실제 제작한 보드의 구성은 <그림 3-18>과 같으며, 각 부분에 대하여 설명하도록 하겠다. 외부 인터페이스 커넥터 부분에는 모두 정전기 보호회로를 부가하였다.

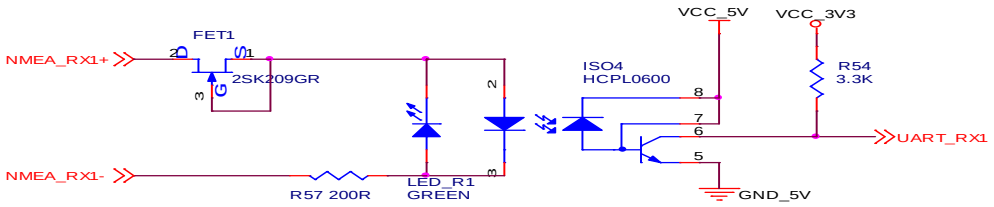


<그림 3-18> 하드웨어 외관

<Fig 3-18> Hardware Outline

1) NMEA0183 수신 포트

NMEA0183 규격에 따라 입력 신호와는 절연처리 하여 규격에 맞게 입력 전압이 최소 2V에서도 동작되도록 설계하였으며, 12 채널을 지원한다. <그림 3-19>는 NMEA0183 표준 문서에서 권고하는 수신회로의 예를 나타낸 것이다.



<그림 3-19> NMEA0183 수신 회로

<Fig 3-19> NMEA0183 Receive Circuit

2) NMEA0183 양방향 포트

이 부분은 4채널로 구성되어 있으며, 각 채널별로 DC-DC 모듈을 이용하여서 전원을 분리하였으며, RS232/RS422/RS485 통신 모드를 지원하도록 설계되어 있다. DIP 스위치의 조작으로 통신 모드를 설정하도록 되어 있어서, 사용 용도에 맞게 설정하면 된다.

3) NMEA2000 포트

NMEA2000 규격에 따라 네트워크와 장치를 절연시켰으며, 네트워크에서 공급되는 전원(DC 9V~16V)을 사용하여 구동되도록 설계되었다. CAN Transceiver 칩을 사용하여 하드웨어 레벨의 인터페이스를 구성하였으며, 포토커플러를 통해서 메인 컨트롤러와 Data를 주고받도록 설계 되었다. NMEA2000 인터페이스 회로에서 외부 인터페이스 포트에는 정전기 방지 기능을 위하여 TVS(Transient Voltage Suppression) 다이오드를 추가하였다.

4) ETHERNET 포트

TCP-IP 인터페이스를 구현하기 위하여 ETHERNET 하드웨어 레벨 인터페이스를 위한 PHY 칩을 사용하였으며, 외부와의 인터페이스는 트랜스포머가 내장된 RJ-45 잭을 사용하였다.

5) DEBUG 포트

Firmware 개발을 하는 동안 디버깅을 위한 용도로 사용이 되며, 추후 제품으로 출시된 경우 Firmware를 업그레이드 할 수 있는 용도로 RS232 포트를 구성하였다. PC와 연결하여 쉽게 이용을 할 수 있으며, 115,200bps를 사용하도록 설정되어 있다.

6) 전원 회로

DC 24V에서 동작되도록 기본 설계되어 있으며, 입력 전압은 AC/DC 18V ~ 36V 사이에서 공급이 가능하도록 되어 있으며, DC 입력의 경우 역방향 결선에 대한

대책으로 브리지 다이오드가 추가되었으며, 이를 통해서 AC의 전원도 입력이 가능하다. 입력 전원과의 절연을 위해서 DC-DC 모듈을 이용하였으며, DC-DC에서 5V 전원이 출력되며, 이를 이용하여 3.3V, 1.8V 레귤레이터를 통해서 보드 내부에 필요한 전원을 만들도록 되어 있다.

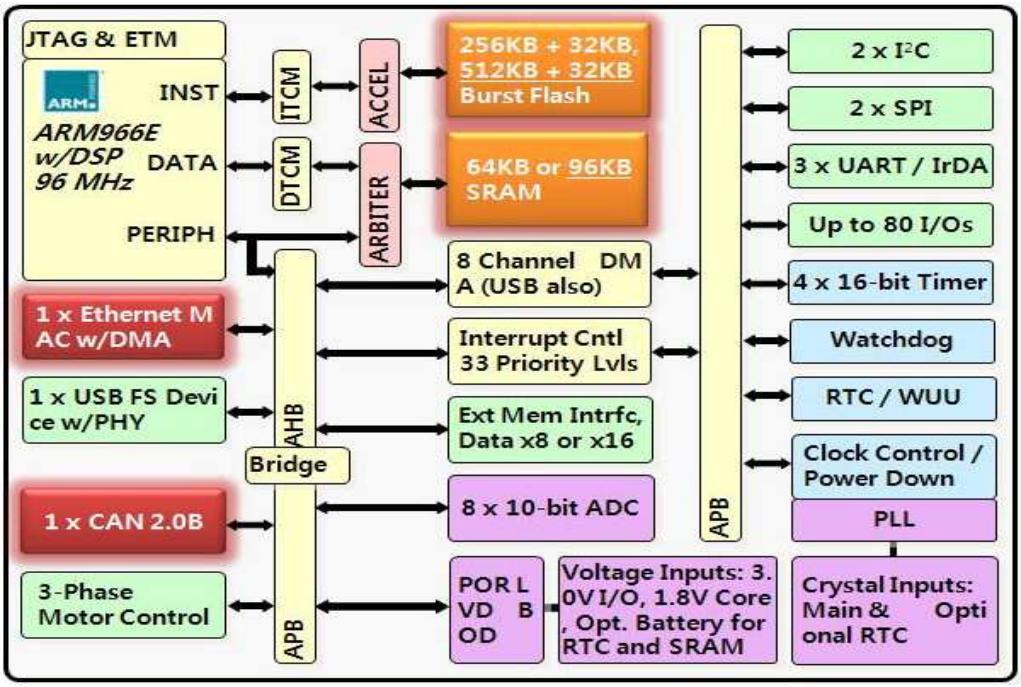
7) UART 칩

NMEA0183 인터페이스를 위해서 UART지원 칩을 사용하였으며, 칩 1개에서 4채널을 지원하며, 16채널을 지원하기 위해서 4개의 칩을 사용하였다. 메인 컨트롤러와의 연결은 Address/Data에 의한 메모리 인터페이스 방식을 사용하였다. 칩 내부에 각 채널별로 송•수신 데이터 처리를 위하여 16 Byte의 FIFO를 지원하고 있다. 이를 이용하여 Byte 단위를 처리를 하지 않게 되어서 메인 컨트롤러의 오버헤드를 줄일 수 있다.

8) 메인 컨트롤러 (STR912)

ST사에서 출시한 칩셋으로 ARM9 Core에 96Mhz의 속도를 제공한다. 512 KByte의 Flash 메모리 및 92 KByte의 SRAM 메모리가 내장되어 있으며, DMA를 지원하는 ETHERNET Core와 CAN Ver.2.0B를 지원하는 CAN Core부분을 제공한다. 추가적으로 소프트웨어 개발에 있어 MCU 내부의 제어를 쉽게 할 수 있도록 소프트웨어 라이브러리를 API형태로 제공하고 있으며, 특히 ETHERNET에서 유용하게 사용할 수 있는 최적화된 메모리 블록 복사 기능의 API도 제공된다.

<그림 3-20>에 메인 컨트롤러에 대한 내부 구조를 소개하였다. STR912에 대해서 좀 더 이해를 하기 위해서는 기본적으로 ARM Core에 대한 부분과 STR912에 대한 부분을 함께 공부하여야 할 것이다.



<그림 3-20> 메인 컨트롤러

<Fig 3-20> Main Control Unit

9) USB 포트

향후에 PC와의 인터페이스를 용이하게 하기 위하여 USB 포트를 구성해 두었다. 이를 이용하여 추후에 NMEA Gateway와 같은 장비를 개발할 경우 PC와 쉽게 연결하여 많은 용도로 이용될 수 있을 것으로 보인다.

10) ARM JTAG 포트

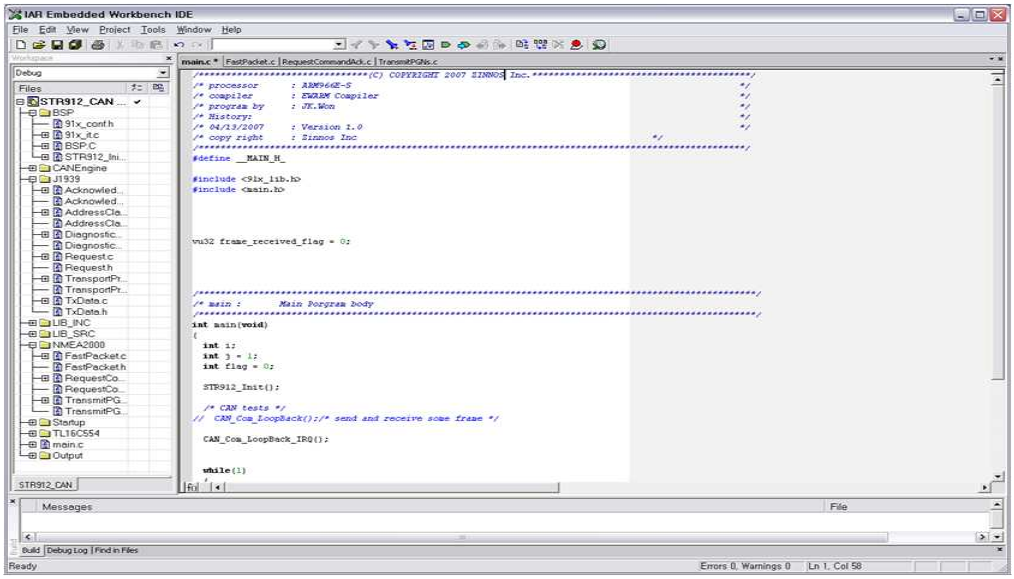
STR912 메인 컨트롤러의 디버깅 환경을 위하여 ARM JTAG포트를 구성하였으며, ARM JTAG툴을 이용하여 PC에서 쉽게 개발을 할 수 있다. ARM JTAG의 경우는 표준 규격이기 때문에 쉽게 이용이 가능하다. 물론 개발을 위해 각 컴파일러 환경에서도 지원이 되고 있다.

제4장 실험 및 검토

4.1 개발 환경

1) 소프트웨어 개발 환경

본 논문에서는 ST사에서 제공되는 STR912관련 API 함수 및 uCOS-II 관련 포팅 예제를 쉽게 적용해서 사용하기 위해서 IAR사의 ARM 컴파일러를 이용하였다. 그리고 IAR 컴파일러를 이용하여 ARM을 디버깅하기 위하여 J-LINK라는 IAR 전용 ARM JTAG 툴을 사용하였다.



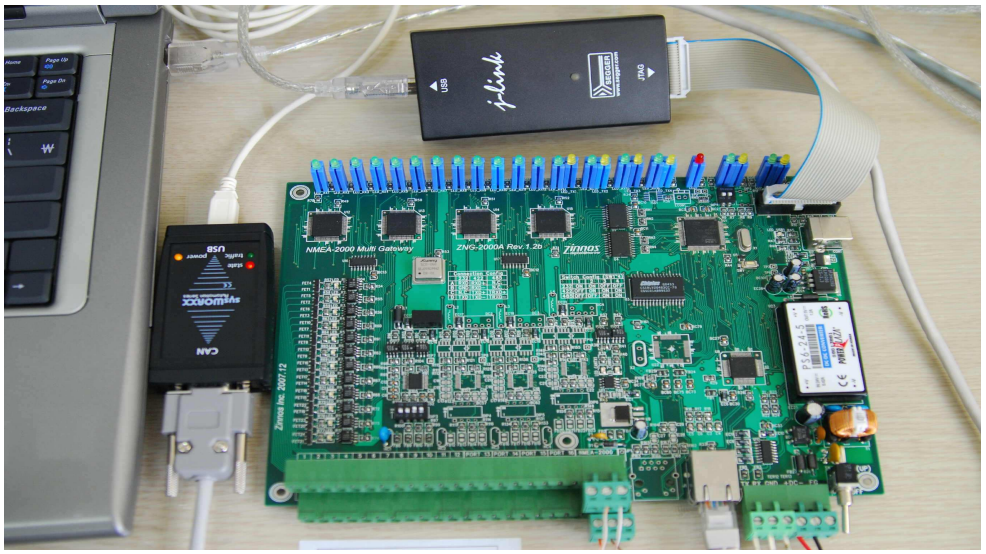
<그림 4-1> IAR사 ARM 컴파일러
<Fig 4-1> IAR ARM Compiler

<그림 4-1>은 IAR사의 ARM 컴파일러에서 제공하는 통합 개발환경의 모습이며, 소프트웨어 개발에 필요한 향상된 에디터 환경과 J-LINK 및 기타 호환되는 ARM용 JTAG 장비를 이용한 디버깅 환경을 제공하고 있다. 추가적으로 Plug-In으로 제공되는 uCOS-II 포팅에 따른 디버깅 환경을 제공하고 있다. 이를 이용

하면 메모리 사용에 대한 부분 및 각 TASK들의 CPU 점유율 및 각 TASK들의 동작 상태를 확인할 수 있다.

2) 하드웨어 개발 환경

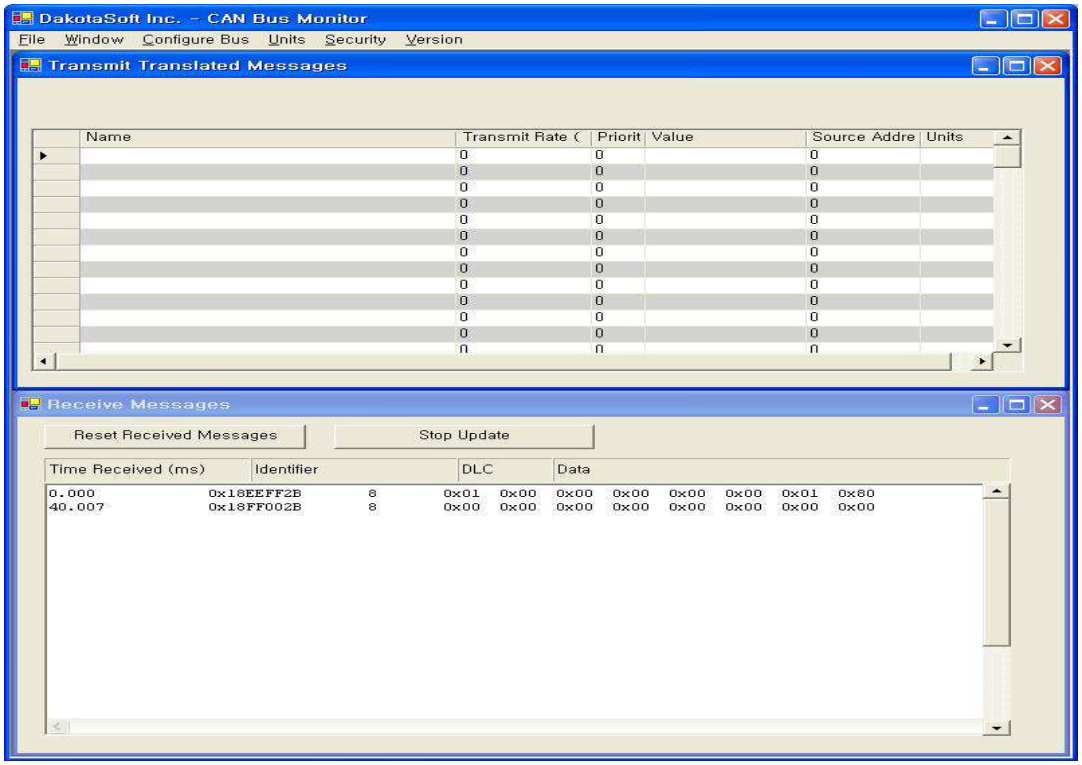
IAR사의 ARM 컴파일러 환경에서 사용하는 ARM용 JTAG툴인 J-LINK 및 NMEA2000 패킷 분석을 위한 CAN to USB 모니터링 모듈을 이용하였다.



<그림 4-2> 개발 툴
<Fig 4-2> Debugging for Hardware

<그림 4-2>에서 PC와 개발 보드간의 연결 상태를 보여주는 것으로 윗부분에 ARM JTAG 디버깅 툴인 J-LINK와 왼쪽에 CAN 모니터링 툴을 볼 수 있다.

<그림 4-3>은 CAN 모니터링 툴과 함께 제공하는 PC에서 볼 수 있는 프로그램으로 실제 보드에서 구현된 NMEA2000 패킷을 송신하여 PC에서 제대로 송신되는지의 여부를 판단할 수 있다.



<그림 4-3> CAN 모니터링 화면

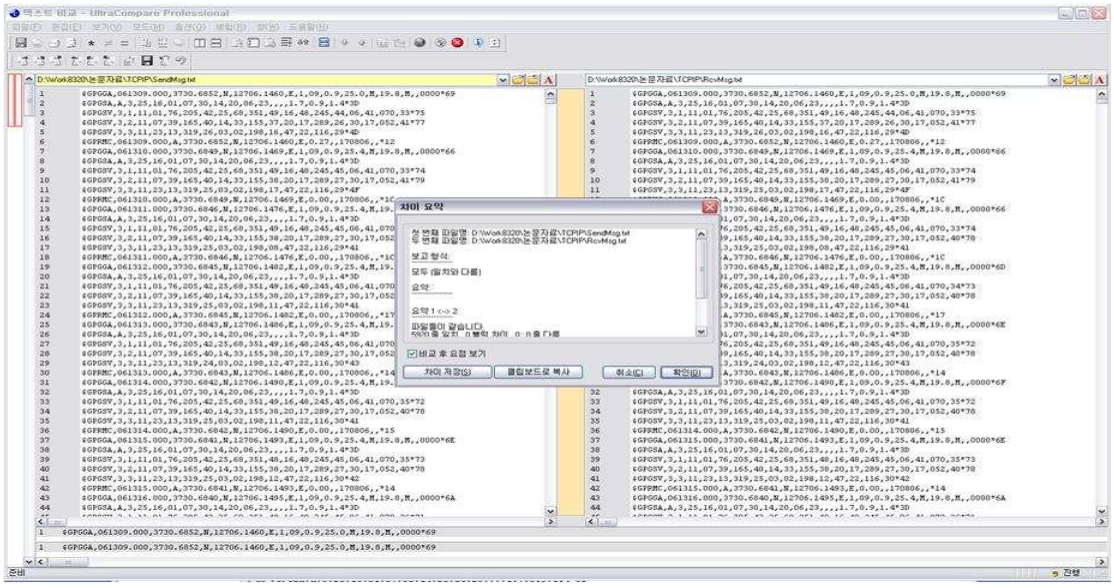
<Fig 4-3> CAN Monitoring

이를 통하여 기본적으로 보드에서 구현된 내용을 검증할 수 있었다. 하지만 NMEA2000 패킷 자체가 바이너리 데이터이기 때문에 사용자의 입장에서는 무슨 내용이 전송되는지를 알아보기 어렵다. 그래서 추가적으로 NMEA2000 모니터링 프로그램을 개발하였다. 이를 통해 전송된 NMEA2000 메시지를 Parsing하여 패킷의 내용을 쉽게 알아 볼 수 있도록 하였다.

4.2 실험 및 검토

1) NMEA0183 패킷 수신 및 TCP-IP 패킷 전송

본 논문의 결과를 확인할 수 있도록 PC에서 NMEA0183 패킷을 저장해 둔 파일(SendMSg.txt, 385 KByte)을 읽어서 패킷 단위로 송신하여서 TCP-IP 데이터로 다시 PC에서 받아보았다. 이 과정을 통해서 NMEA0183 데이터의 수신 처리 루틴의 확인 및 TCP-IP 데이터 송신 루틴을 확인 할 수 있었다. 실험을 결과의 확인을 위해서 송신한 NMEA0183 패킷이 저장된 파일과 TCP-IP를 통해 수신한 패킷을 저장한 파일(RcvMsg.txt)을 파일 비교기 프로그램을 통하여 비교하였다. 그 결과 <그림 4-4>와 같이 틀린 곳이 없이 모두 일치함을 확인 할 수 있었다. NMEA0183 패킷으로 대략 5900 문장 이상을 송신하여 얻은 결과이다.



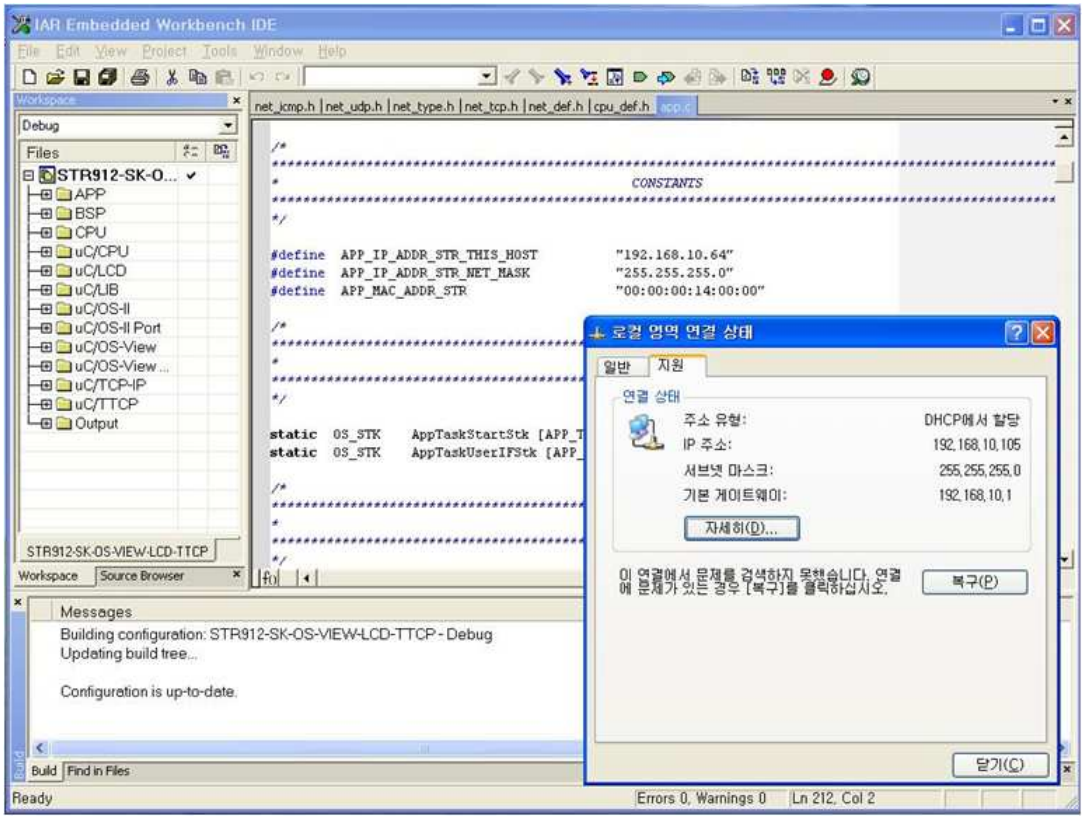
<그림 4-4> 송·수신 파일 비교

<Fig 4-4> Sending·Receiving Data Comparison

2) TCP-IP 부가 기능 테스트

TCP-IP의 부가 기능에 대한 정상 동작여부를 판단하기 위하여 추가적으로 ARP 및 PING 테스트를 실시하였다. PC에서 다른 호스트에 연결을 시도하기 위

해서는 처음에 ARP 테이블을 비교하여 상대 호스트 주소에 해당하는 MAC 주소를 확인하게 되어 있다. 그래서 ARP 테이블에서 사용한 개발 보드의 MAC 주소를 삭제한 후에 PING 테스트를 시도함으로써 ARP 및 PING 테스트를 함께 할 수 있었다. TCP-IP 패킷을 PC에서 캡처하여 분석하기 위하여 무료로 제공되는 ETHEREAL 이라는 프로그램을 이용하였다. 이 프로그램은 추가적인 하드웨어를 필요로 하지 않기 때문에 쉽게 이용이 가능하다. 그리고 원하는 프로토콜의 패킷만 필터링하여 수신도 가능하며, 특정 호스트에서 오는 패킷에 대해서만도 수신이 가능하다. PC에서는 다양한 응용프로그램이 돌아가기 때문에 많은 패킷이 유입되기 때문에 특정 호스트와 송·수신한 내용만 필터링하여 캡처하기 위하여 호스트 IP주소를 "192.168.10.64"로 지정하여 패킷을 수신하였다.



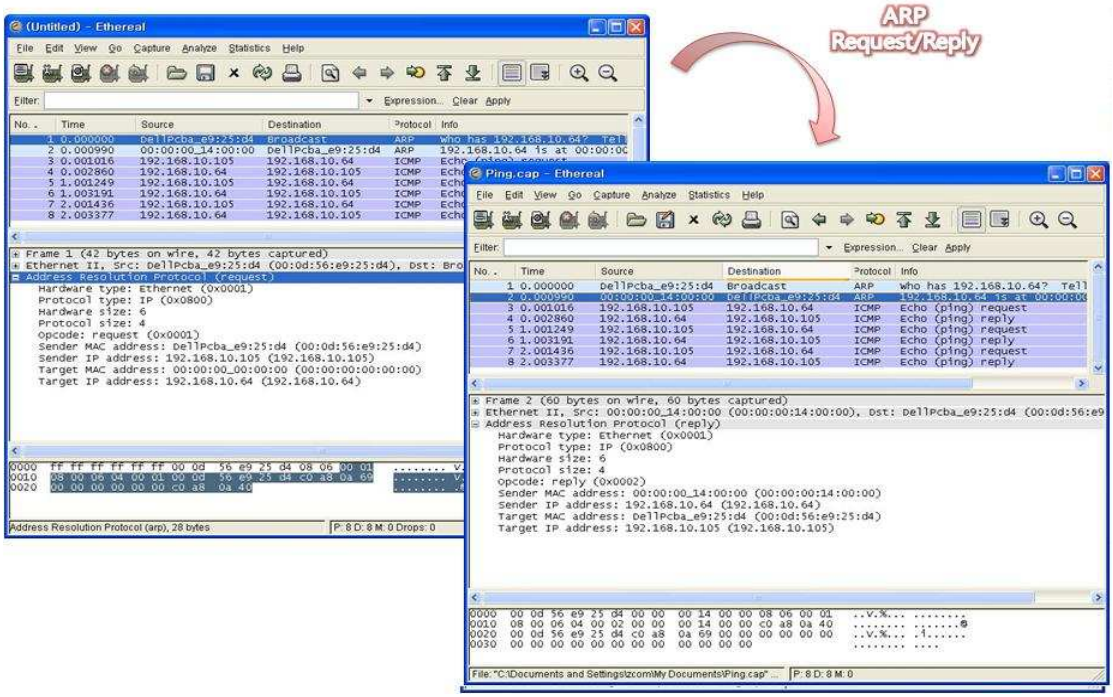
<그림 4-5> IP 주소 확인

<Fig 4-5> Check Configuration

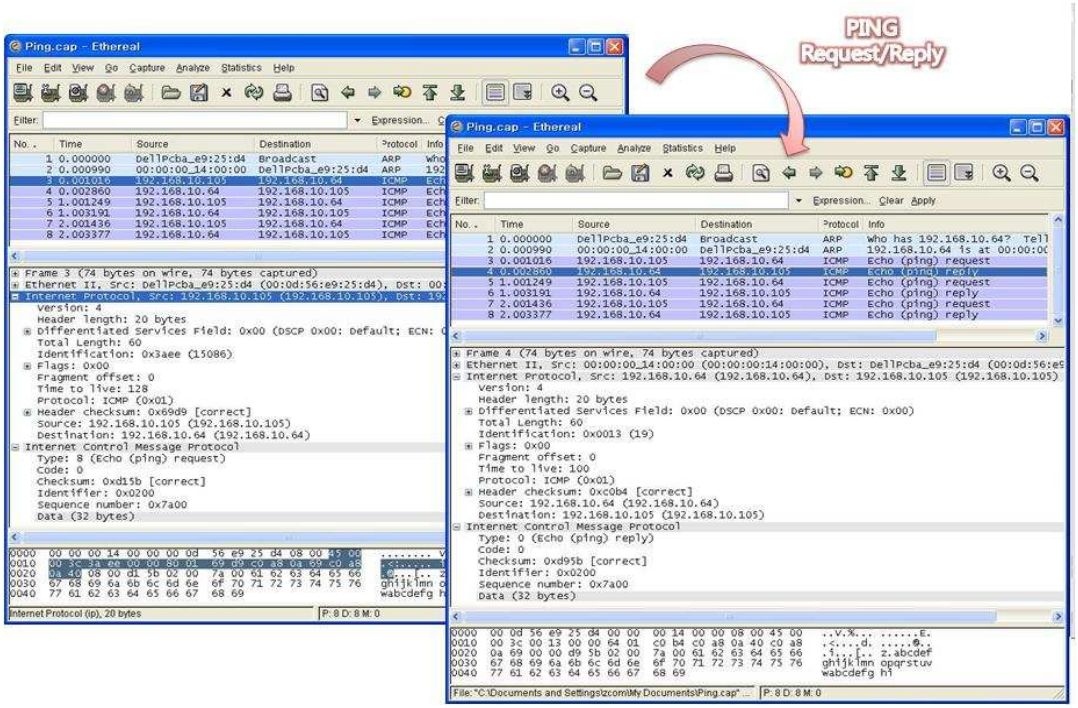
<그림 4-5>에서 보는바와 같이 PC의 IP주소는 "192.168.10.105"이며, 개발 보드의 IP주소는 "192.168.10.64"이며 MAC주소는 “00:00:00:14:00:00"으로 설정되어 있다.

PC에서 개발 보드에 PING 에코 요청을 시도하였으며, 그에 따라서 개발 보드에서 응답이 오는 과정을 ETHEREAL을 통해서 캡처하여 확인을 하였다.

<그림 4-6>은 ARP 요청 및 응답한 부분을 보인 것이다. 이를 통하여 개발 보드의 논리 주소(IP주소)를 통하여 물리 주소(MAC주소)를 획득하였으며, 다음으로 실제 테스트 하려고 한 PING 에코 요청 및 응답에 대한 패킷을 확인 할 수 있었다.



<그림 4-6> ARP 테스트 패킷
<Fig 4-6> Packet for ARP Testing



<그림 4-7> ICMP 에코 요청·응답 패킷

<Fig 4-7> Packet for ICMP Testing

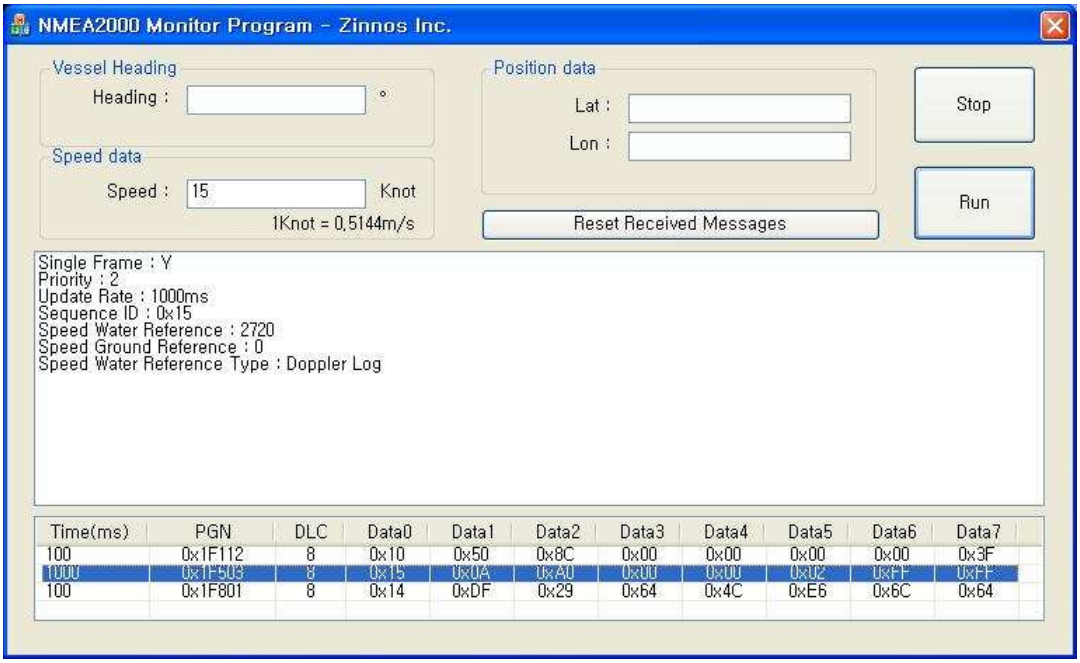
<그림 4-7>은 ICMP 프로토콜 중에서 에코 요청 및 응답 패킷을 보인 것으로 정상적으로 동작하는 것을 확인 할 수 있었다. 그리고 추가적으로 PC에서 ARP 테이블에 개발 보드의 정보가 추가된 것을 확인 할 수 있었다.

3) NMEA2000 패킷 전송

마지막으로 NMEA2000 프로토콜 스택에 대한 정상 동작여부를 확인하였다. 이를 확인하기 위하여 PC에서 NMEA0183 패킷을 송신하여 개발보드에서 NMEA0183 패킷을 수신하여 Parsing과정을 통하여 추출된 데이터를 다시 NMEA2000 패킷으로 생성하여 다시 PC로 보내는 과정을 수행하였다. 이를 개발 결과 테스트를 위하여 개발한 NMEA2000 모니터링을 이용하여 PC에서 확인을 하였다.

PC에서 NMEA0183 Speed 데이터 "$VMVBW,15.0,,A,,,V*hh<CR><LF>" 을 송신하여

NMEA2000 PGN 정보 128259(0x1F503)에 해당하는 패킷을 수신할 수 있었다. 이 패킷을 수신하여 화면에 Speed 정보에 15Knot가 표시된 것을 확인 할 수 있다.



<그림 4-8> NMEA2000 패킷 모니터링

<Fig 4-8> NMEA2000 Monitoring

제5장 결 론

본 논문의 목적인 선박 내에서 항해통신장비의 데이터 교환을 위한 표준 통신 프로토콜인 NMEA0183, TCP-IP, NMEA2000 규격에 대한 이해를 통하여 쉽게 사용을 할 수 있는 NMEA0183, TCP-IP, NMEA2000 프로토콜 스택을 구현하였다.

본 논문에서 설계한 하드웨어는 16 Channel의 UART 포트, CAN 인터페이스 포트, Ethernet 인터페이스 포트 및 소프트웨어에 의한 프로토콜 스택 구현 및 데이터 임시 저장 공간을 위한 외부 SRAM을 추가 구성하였다. 소프트웨어는 Micrium사에서 제공하는 uCOS-II라는 RTOS를 STR912 메인 컨트롤러에 포팅 하여, Multi-Tasking 기법을 이용하여, NMEA0183 메시지 송·수신 TASK, NMEA2000 메시지 송·수신 TASK 및 TCP-IP 메시지 송·수신 TASK를 생성하였으며, 각 TASK의 동기화를 위하여 세마포어를 이용하였으며, RTOS상에서 제공하는 메시지 큐 관리 기능을 사용하여 각 프로토콜별로 관리하여야 하는 메시지를 관리하였다. STR912 제작회사인 ST사에서 제공하는 최적화된 메모리 버퍼 복사 알고리즘을 이용하여 메시지 큐에서 메시지들을 이동하도록 하였으며, TCP-IP의 경우는 CPU에서 제공하는 DMA 기능을 이용하여 수신 메시지에 대해서 수신 버퍼에 적재되도록 하였다.

실험 결과, NMEA0183, NMEA2000 및 TCP-IP 프로토콜에 대한 규격을 만족하였으며, 선박용 항해통신장비 개발 업체들에서 프로토콜 스택을 이용할 경우 많은 기대효과를 볼 수 있을 것으로 보인다. 첫 번째 프로토콜 규격 분석에 대한 시간을 단축시키며, 두 번째 프로토콜 스택 구현에 대한 시행착오 및 개발 기간을 단축시키며, 세 번째 프로토콜 스택을 쉽게 이용할 수 있도록 표준 모듈화를 진행하였기 때문에 이용에 큰 어려움이 없을 것으로 판단된다.

단, STR912라는 특정 프로세서를 타깃으로 하였기 때문에, 하드웨어 의존적

인 부분에 대해서는 조금의 수정작업이 필요할 것으로 보인다. 하지만 같은 ARM Core를 사용할 경우 개발환경에 대한 변경이 불필요함으로 쉽게 적용이 가능할 것으로 생각된다. 그리고 NMEA0183 패킷의 처리에 있어서 6 Bit 이진 형태의 패킷에 대한 처리 부분을 구현하지 않았는데, 추가로 진행할 예정이다.

최근 선박용 제품들 중에서 NMEA2000 표준 규격을 따르는 제품들이 등장하고 있는 상황에서 NMEA2000 관련 제품을 개발하려고 할 경우 이 프로토콜 스택을 이용한다면 큰 파급효과를 가져올 것으로 생각된다. 그리고 NMEA REPEATER, NMEA MULTIPLEXER, NMEA CONVERTER, NMEA GATEWAY등 다양한 인터페이스 장비의 개발에 유용하게 이용할 수 있으며, 추가적으로 시스템 구축에 있어서도 하나의 모듈 형태로 추가하여 사용한다면 기대효과는 더욱 커질 것으로 판단된다.

참 고 문 헌

[1] Membership Information in NMEA, http://www.nmea.org

[2] Frank Cassidy - Chairman of NMEA, "NMEA2000 Explained - The Lastest Word", 1999.03.02

[3] NMEA0183(IEC61162-1), Standard for Interfacing Marine Electronic Devices, Ver 3.01, 2002.01.01

[4] NMEA0183-HS(IEC61162-2), 38.4K Baud Serial-Data Standard for Interfacing Marine Electronic Devices, Ver 1.01, 2002.11.01

[5] EIA/TIA-232-E: Interface Between Data Terminal Equipment and Data Circuit-Termination Equipment Employing Serial Binary Data Interchange.

[6] EIA/TIA-422B: Electrical Characteristics of Balanced Voltage Digital Interface Circuits (ITU-TRV.11)

[7] NMEA2000: Standard for Serial-Data Networking of Marine Electronic Devices, Ver 1.20, 2004.09

[8] ISO-11898 - Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for High-speed Communications, First Edition November 1993, Amendment 1 April 1994.

[9] ISO11898-1: Road vehicles-Controller area network(CAN) - Part1: Data link layer and physical signalling

[10] ISO11898-2: Road vehicles-Controller area network(CAN) - Part2: High-speed medium access unit

[11] ISO11783-3: Tractors and machinery for agriculture and factory-Serial control and communication data network - Part3: Data link layer

[12] ISO11783-5: Tractors and machinery for agriculture and factory-Serial control and communication data network - Part5: Network management

[13] Douglas E. Comer, "Internetworking with TCP/IP, Edition 5th", 2005.06

[14] IEEE802.3u: IEEE Standards for Local and Metropolitan Area Networks: Supplement to Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications : Media Access Control (MAC) Parameters, Physical Layer, Medium Attachment Units, and Repeater for 100 Mb/s Operation, Type 100BASE-T

[15] Behrouz A. Forouzan, "Data Communications and networking, Edition 2nd", 2002.06

감사의 글

지난 2년을 자식을 키우시듯 지도 편달해 주신 황승욱 교수님께 진심으로 감사드립니다. 그 동안의 교수님의 배려와 격려에 부응하지 못한 것 같아 죄송한 마음뿐입니다. 바쁘신 일정에도 세심하게 논문을 지도해 주시고 심사해 주신 김기문 교수님, 이서정 교수님께 감사의 말씀 올립니다.

논문을 완성하기까지 지원을 해 주신 지노스의 심진보 사장님과 연구 진행에 많은 도움을 준 해상통신 연구실의 정인 선배, 김정근 선배에게 감사의 마음을 전합니다. 그리고 연구 진행에 있어서 함께 해 준 여러 후배들에게도 고마움을 전합니다.

오늘이 있기까지 자식 걱정하시는 아버님·어머님, 그리고 항상 자식처럼 건강을 걱정해 주신 장인·장모님께 다시 한 번 감사드립니다. 그리고 지금껏 아무 말 없이 인내하며 뒷바라지 해 준 아내에게 진심으로 고맙다는 말과 항상 곁에서 함께 해주지 못 한 아들에게 미안한 마음을 전합니다.

오늘의 이 작은 결실을 출발점으로 삼아 앞으로 더욱 열심히 살아가도록 다짐을 해 보며, 끝으로 배움의 길에서 항상 옳은 길로 인도해 주신 지도 교수님께 다시 한 번 감사의 말씀 올립니다.

2008년 08월