工學碩士 學位論文

최적화된 큐 기반의 NMEA 멀티플렉서 설계 및 구현

Design and Implementation of NMEA Multiplexer based on the Optimized Queue

指導敎授 林 宰 弘

2005年 2月

韓國海洋大學校 大學院

電子通信工學科

催 正 勳

목 차

표 목 차

그 림 목 차

Abstract

The National Marine Electronics Association(NMEA) is nonprofit-making cooperation composed with manufacturers, distributors, wholesalers and educational institutions. It could be defined as the interface and data protocol regarding to the electronic signal for the communication among equipments on the sea, which is widely used protocol for the interfaces in such as marine equipment including depth recorder, alpha radar, ECDIS, and GPS receiver.

The equipment use a basic port in order to process the signal from equipment using NMEA signal. When port of equipment don't have enough, use the multi-port for processing. However, we need to have module development simulation which could multiplex and provide NMEA related signal that we could solve the problems in multi-port application and exclusive equipment generation for a number of signal.

For now, we don't have any case or product using NMEA multiplexor so that we import expensive foreign equipment or embody NMEA signal transmission program like software, using multi-port. These have problems since we have to pay lots of money and build separate processing part for every application programs. Besides, every equipment generating NMEA signal are from different manufactures and have different platform so that it could cause double waste and loss of recourse. For making up for it, I suggest the NMEA multiplexor embodiment, which could

independently move by reliable process and high performance single hardware module, improve the memory efficiency of module by designing the optimized Queue, and keep having reliability for realtime communication among the equipment such as main input sensor equipment Gyrocompass, Echo-sound, and GPS.

제 1 장 서 론

순항하는 선박과 항공기가 일정 항로에 따라 목표지점까지 가기 위해서는 여러 가지 제반 사항이 따른다. 그중에서도 운송물을 안전하고 빠르게 이동하기 위해서 현재의 위치 정보를 알려주는 GPS 수신기와 같은 장치가 필요할 것이며, 또한 자이로콤파스(Gyrocompass), 풍향풍속계(Anemometer), 스피드 로그(Speed Log), 타각지시기(Rudder) 등 해상전자장치들도 필요할 것이다.

그래서 해상전자장치들 간에 통신을 하기 위한 표준 통신 프로토콜이 필요하며, 이러한 해상전자장치들 간의 표준통신 프로토콜을 정하는 목적으로 미국해상전자통신협회(NMEA : National Marine Electronics Association)라는 위원회가 발족하였다. NMEA는 해양전자산업의 발달 및 교육, 판매 시장에 공헌하는 위원회이며, 해상전자장치 사이에 통신을 위하여 전기적 신호에 관한 인터페이스와 데이터 프로토콜로 정의할 수 있다[1],[2].

NMEA 프로토콜을 사용하는 장치로부터 출력되는 신호를 처리하기 위해서, 장치의 기본 포트를 이용하면서 부족할 경우 멀티포트를 사용하여 처리하고 있다. 그러나 수많은 출력신호에 대한 부가 적인 멀티포트의 적용과 전용장치로서 출력에 대한 제약사항의 문제점이 발생 하므로, 이를 해결하기 위하여 NMEA 관련 신호들을 다중화(Multiplexing) 할 수 있는 모듈인 멀티플렉서가 필요하다.

현재 국내에서는 NMEA 멀티플렉서를 개발한 사례와 제품이 없기 때문에 외국의 고가장비를 수입하여 사용하거나, 장치의 입, 출력 부분은 멀티포트를 사용하고 NMEA 신호처리 부분에는 소프트웨어를 이용한 응용프로그램으로 구현하여 사용하고 있다. 그런데 이러한 방법은 고비용이 지출되거나 각 응용프로그램 제작 시 제어

하는 별도의 프로그램을 작성해야하는 문제점이 발생한다. 또한 NMEA 신호를 출력하는 각각의 장치들은 제조회사 및 플랫폼이 다르므로 이중의 자원낭비 및 손실 등도 초래할 수 있다. 그래서 이를 보완하기 위하여 NMEA 신호의 다중화와 신뢰성 있는 신호처리 방법 등 고성능의 단일 하드웨어 모듈로서 독립적인 동작을 할 수 있게 하고, 외국의 고가장비 구입에 따른 비용을 줄이기 위한 국산화된 NMEA 멀티플렉서의 구현이 필요하다.

본 논문에서는 NMEA 멀티플렉서의 모듈 설계 및 제작에 앞서 NMEA 멀티플렉서가 신뢰성 있는 NMEA 신호를 처리하기 위한 최적화된 큐의 설계를 이용하여 모듈의 메모리 효율을 높이며, 중요 입력 센서 장치인 자이로콤파스, 에코 사운드, GPS 등의 장치들과 실시간 통신의 높은 신뢰성을 유지할 수 있는 NMEA 멀티플렉서의 기능적 구현에 대하여 제안하였다.

본 논문의 구성은 제 2 장에서 NMEA의 개요와 멀티플렉서의 개요 및 NMEA 멀티플렉서의 기능에 대해서도 기술하였고, 제 3 장에서는 NMEA 멀티플렉서 설계를 기술하면서 대기-큐 알고리즘의 내용을 중점 두어 기술하였으며, 제 4 장에서는 NMEA 멀티플렉서의 구현 내용을 기술하였다. 그리고 제 5 장에서는 결론과 향후 연구계획에 대하여 기술하였다.

제 2 장 NMEA 멀티플렉서

2.1 NMEA 개요 및 신호

풍향 풍속계, 스피드 로그, 자이로콤파스와 같은 해상전자장치들이 보내는 NMEA 신호의 개요와 NMEA 신호 종류 및 형식에 대해서 설명한다.

2.1.1 NMEA의 개요

해상전자장치들은 일반 전자장치와 달리 제각기 다른 통신 신호의 형식을 가지고 있다. 그래서 해상전자장치 간에 통신을 하기위해 상호간 통신 규약이 필요하며, 이러한 해상전자장치 간의 통신을 위한 전기적인 인터페이스 및 프로토콜 표준을 정하는 목적으로 설립된 위원회가 미국해상전자협회이다. 일반적으로 NMEA란 약어로 불리며 제조업자들, 배급업자들, 도매업자들 그리고 교육 기관들로 구성된 비영리적인 제휴이다[1],[2].

2.1.2 NMEA 신호 종류 및 형식

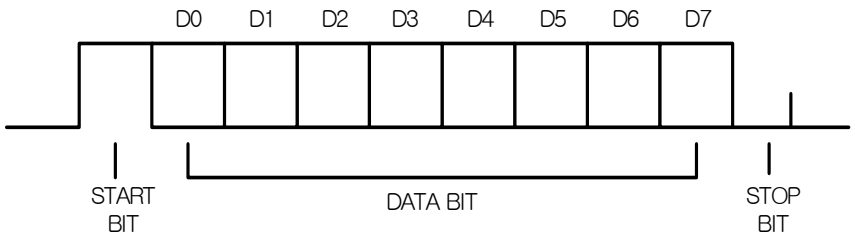
(1) NMEA 신호 종류

NMEA 신호는 NMEA-0180, 0182, 0183 등의 종류가 있으며, 현재 국제 표준으로 쓰이는 NMEA-0183을 중심으로 설명한다.

우선 NMEA-0180과 NMEA-0182의 차이점은 단순 데이터 포맷인가 아니면 복합 데이터 포맷인가 하는 차이점뿐이다. 단순 데이터

포맷은 총 7비트로 비트 0에서 비트 5까지는 크로스 트랙 에러를 나타내며, 비트 6에서의 값이 1이면은 데이터가 유효함을 나타내는 것이고, 비트 7에서의 값 0은 단순 데이터 포맷을 나타내는 것이다. 복합 데이터 포맷은 아스키코드로 나타내어진 크로스 트랙 에러와 방위각, 현재의 위도/경도 및 2진 상태 바이트로 구성된 37바이트의 데이터 블록으로 이루어져 있다. 복합 데이터 포맷에서의 모든 바이트는 비트 7의 값이 1로써 단순 데이터 포맷형식과 구별이 되어진다. 그리고 단순 데이터와 복합 데이터를 동시에 전송이 가능하며, 복합 데이터 블록 중간에 단순 데이터를 삽입도 가능하다[3].

그리고 NMEA-0183은 데이터 전송을 ANSI 표준에 따라서 직렬 비동기 방식으로 전송한다. <그림 2-1>에서처럼 첫 번째 비트는 시작 비트이며, 다음에 따라 오는 비트는 8비트의 데이터 비트이며, 마지막 비트는 정지 비트로서 이루어져 있다[2].



<그림 2-1> NMEA-0183 데이터 전송

<Fig. 2-1> NMEA-0183 data transmission

(2) NMEA 신호 형식

NMEA-0183은 ASCII 코드의 모든 문자를 사용하며, 데이터는 문장형태로 전송이 되어진다. <그림 2-2>는 일반적인 NMEA-0183 데이터 형식을 보여주고 있다. 각 문장은 “$”로 시작을 하며, 송신자 ID를 나타내

는 두 개의 문자와 문자 3개로 표현이 되는 데이터 ID, 콤마로 구분된 데이터 필드가 따라오고 추가적으로 체크섬과 캐리어 CRLF로 종결이 된다. 그리고 한 문장은 $와 CRLF를 포함해서 82개의 문자로 구성되어 있다[1].

만약에 필드의 데이터가 유효하지 않으면 해당 필드를 생략하거나 콤마로 범위를 정하여 전송을 하며, 이때 문자 사이에 빈 공간은 없어야 한다. 체크섬은 “*”와 두 자의 16진수로 이뤄지는데, “$”와 “*”를 포함하지 않는 모든 글자의 배타적-OR의 결과이다[2].

$	GP	APB,	189.0, 03.6, M, 04, W	CRLF
헤더	송신측 ID	데이터 ID	데이터	종단

<그림 2-2> NMEA-0183 데이터 형식
<Fig. 2-2> NMEA-0183 data format

⑶ NMEA 신호의 문장 분석

<표 2-1>은 NMEA-0183의 데이터 형식에서 데이터 ID 종류와 그에 해당하는 데이터 ID에 대해서 설명하고 있으며, 일반적으로 데이터 ID를 표준문장이라 정의하고 있다[2].

<표 2-1> NMEA-0183 표준문장

<Table 2-1> NMEA-0183 standard sentence

표준문장	설 명
APB	오토파일럿 형식 B
BOD	방위각 - 목적기 항행지점에 대한 실제 값
BWC	항행지점에 대한 방위각과 거리 - 대형 원주
BWR	항행지점에 대한 방위각과 거리 - 항정선
DBT	방향 깊이 변환기
GGA	세계측위시스템 고정 데이터
GLL	지리적 위치, 위도와 경도
GSA	GPS DOP와 능동 위성
GSV	시계 내의 위성
HDM	방향, 자기
HSC	조타에 대한 방향 명령
MTW	해수 온도
ROO	현재 유효한 경로에서의 항행지점 ID 리스트
RMB	권고된 최소 항행 정보
RMC	권고된 최소 특수 GPS/Transit 데이터
RTE	유효한 경로에서의 항행지점
VHW	해수의 속도와 방향
VWR	관련된 바람의 방향과 속도
VTG	최적항로설정 그리고 그라운드 속도
WCV	항행지점 종단 속도
WPL	항행지점 위치
WDC	항행지점에 대한 거리
WDR	항행지점 거리, 항정선
XTE	크로스-트랙 에러, 측정값
XTR	크로스-트랙 에러 - 필요 없는 값 계산

<그림 2-3>은 실제 GPS들로부터 전송되는 데이터들 중 표본 추출하여 보여 주는 그림이다.

```
$GPGLL,3957.3089,N,12651.1101,E,074405.359,A*33
$GPGGA,074405.36,3957.3089,N,12651.1101,E,1,04,2.0,-0017,M,,,,*31
$GPRMB,A,0.01,L,SIM001,SIM002,3959.1290,N,12651.0980,E,001.8,000.,0
21.7,V*05
$GPRMC,074405.36,A,3957.3089,N,12651.1101,E,21.7,002.7,050704,08.,W*
78
$GPAPB,A,A,0.0,L,N,V,V,0.0,T,SIM002,359.7,T,357.0,T*7E
$GPGSA,A,3,01,02,03,04,,,,,,,,,2.0,2.0,2.0*34
$GPGSV,3,1,08,22,67,215,,14,53,315,,30,50,161,,05,49,071,*7E
$GPGSV,3,2,08,18,41,170,,09,33,055,,25,16,251,,15,13,230,*72
$GPGSV,3,3,08,,,,,,,,,,,,,,,,*71
```

<그림 2-3> NMEA-0183 출력 형식

<Fig. 2-3> NMEA-0183 output format

여기서 전송 데이터의 문장을 자세히 설명하기 위해 <그림 2-3>에서 박스로 표시된 데이터를 표본으로 분석한다.

$GPGGA, 074405.36, 3957.3089, N, 12651.1101, E, 1, 04, 2.0, -0017, M, ,,,,*31

GGA	세계측위시스템 고정 데이터
074405.36	07 : 44 : 05.36 UTC(우주표준시) 고정값
3957.3089, N	위도 39。 57.3089′ N
12651.1101, E	경도 126。 51.1101′ E
1	1 = GPS 고정(위치 고정지시자)
04	위성궤도의 수
2.0	위치의 수평도
-0017, M	고도, 미터, 해수면 평균 이상
*31	체크섬

“*” 뒤의 문자는 체크섬을 나타내며 이는 대부분의 문장에 추가적으로 삽입하고 표준에 따른다. <표 2-2>는 GPS 수신기의 모드 상태정보를 나타내주는 위치 고정지시자를 설명한 것이다[6],[7].

<표 2-2> 위치 고정지시자

<Table 2-2> Position fix indicator

값	설 명
0	부적합한 모드임
1	GPS, SPS 모드임
2	차동 GPS, SPS 모드임
3	GPS, PPS 모드임
4	GPS, RTK 모드임

그리고 제조업자들이 독자적인 문장을 만드는 것을 허용하는데, 독자적인 문장은 GPS로부터의 출력 될 수도 있고, 제어 정보를 지닌 입력으로도 사용될 수 있다. 이 경우의 포맷은 다음과 같다. “$”로 시작하고, 3글자의 제조자 ID, 제조자 데이터, 그리고 일반적인 문장의 포맷으로 구성이 되며, ″M″ 또는 “Z″는 특수 문장 형식을 나타낸다. 다음은 독자적인 문장의 예를 보여주고 있다[3].

$PGRME, 15.0, M, 450., M, 25.0, M*22

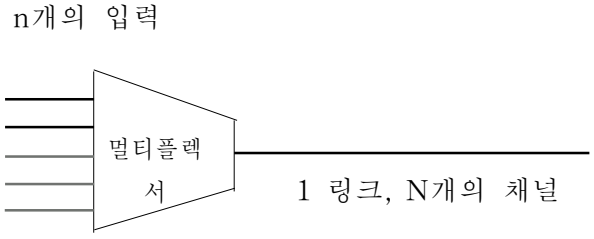
GRME	Garmin의 상인코드
15.0, M	미터단위로 평가된 수평위치 에러
450., M	미터단위로 평가된 수직위치 에러
25.0, M	전체 구형 위치 에러

2.2 NMEA 멀티플렉서

멀티플렉서 개요와 시스템 구성 시 NMEA 멀티플렉서 기능에 대해서 설명한다.

2.2.1 멀티플렉서의 개요

제한된 용량을 갖는 통신 자원에 대하여 다수의 독립적인 사용자들이 서로 공동의 자원으로 채널을 공유하기 위하여 다중화가 필요하다. 즉, 다중화란 몇 개의 DTE(Data Transmission Equipment)가 하나의 통신 회선을 통하여 결합된 형태로 신호를 전송하고 이를 수신 측에서 원래의 형태로 나누어주는 방식으로, 두 개 혹은 그 이상의 신호를 결합하여 물리적 회선이나 링크를 통하여 전송해 주는 방식이다. 다시 말해서 다중화란 1개의 전송로에 복수의 데이터 신호를 중복시켜서 전송하는 것을 말한다. <그림 2-4>와 같이 멀티플렉서는 n개 입력회선으로부터 데이터를 받아들여 하나로 결합하여 고용량 데이터 링크로 보낸다[4].

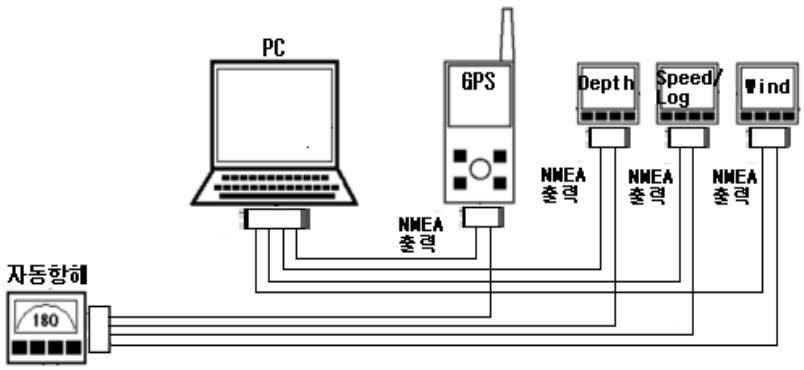


<그림 2-4> 멀티플렉서 기능
<Fig. 2-4> Multiplexer function

현재 많이 사용되고 있는 다중화 방식에는 크게 주파수 분할 다중화 방식과 시분할 다중화 방식, 코드분할 다중화 접속, 광파장 분할 다중화 방식이 통신에 이용되고 있다[5].

2.2.2 NMEA 멀티플렉서 기능

<그림 2-5>는 GPS, 풍향풍속계, 스피드 로그와 같은 해상전자장치들과 NMEA 신호를 필요로 하는 자동항해 장치와 NMEA 신호 확인용 컴퓨터의 연결 구성을 보여주고 있다.



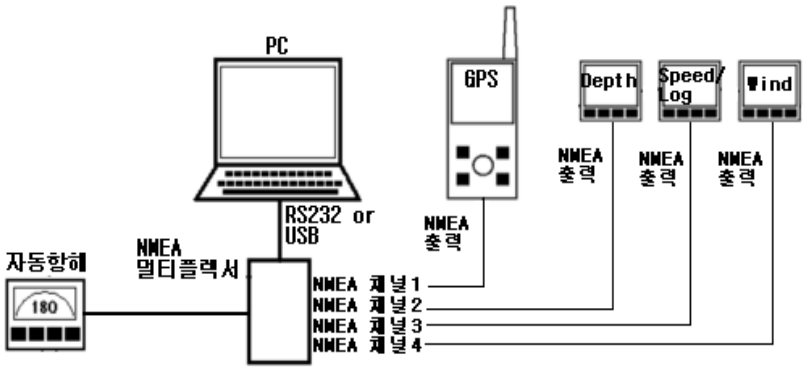
<그림 2-5> 해상전자장치들의 연결

<Fig. 2-5> Connection of the marine electronic equipment

각 해상전자장치들과 자동항해 장치나 NMEA 신호 확인용 컴퓨터가 서로 연결하기 위해서 1대1 상태로 연결이 되어있다. 그런데 자동항해 장치가 여러 개의 NMEA 출력장치들과 연결을 하기위해서는 통신포트가 증가하는 것을 볼 수 있으며, 그래서 각 장치들은 출력부분이나 입력부분에 여러 개의 통신포트를 수용하기 위해서 멀티포트의 사용이 불가피 해진다. 또한 자동항해 장치는 여러 개의 장치들로부터 들어오는 NMEA 신호를 처리하기 위해서 응용프로그

램도 필요하게 될 것이다. 만약 NMEA 신호를 필요로 하는 자동항해 장치나 다른 장치들이 늘어난다면 통신포트의 증가와 응용프로그램 증가로 많은 비용이 들 수밖에 없을 것이다.

그래서 <그림 2-6> 같이 해상전자장치와 자동항해 장치나 NMEA 신호 확인용 컴퓨터의 사이에 NMEA 멀티플렉서를 두어 각 장치들로부터 오는 NMEA 신호를 받아 집중화하고, 집중화된 NMEA 신호를 필요로 하는 자동항해 장치 등에게 보내게 한다.



<그림 2-6> NMEA 멀티플렉서 기능

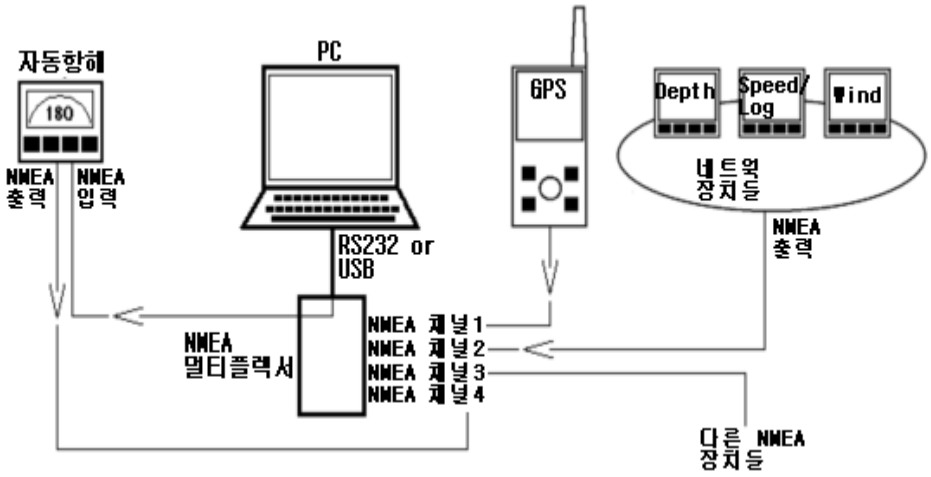
<Fig. 2-6> NMEA multiplexer function

이렇게 NMEA 멀티플렉서는 각 장치에서 사용하는 NMEA 신호를 집중화하여 제공함으로써 포트의 자원 낭비를 막을 수 있고, 또한 부가적인 멀티포트를 구입하거나 이를 이용한 응용프로그램의 제작 시 제어하는 별도의 프로그램을 작성할 필요가 없어진다.

제 3 장 NMEA 멀티플렉서의 설계

3.1 NMEA 멀티플렉서 요구사항

<그림 3-1>은 모듈로 된 NMEA 멀티플렉서를 이용한 NMEA 통신 시스템 구성을 보여주고 있는 그림이다.



<그림 3-1> NMEA 통신 시스템 구성도

<Fig. 3-1> NMEA communication system structure

NMEA 통신 시스템 구성은 네트워크로 구성된 GPS, 풍향풍속계, 스피드 로그 등 해상전자장치들과 그 외의 NMEA 신호를 보내는 장치들로부터 NMEA 신호들을 NMEA 멀티플렉서로 보내고, NMEA 멀티플렉서에서 이 신호들을 받아 집중화 하여 자동항해 장치나 NMEA 신호 확인용 컴퓨터로 신호를 전송하는 시스템으로 구성하고 있다. 여기서 NMEA 멀티플렉서는 본 논문에서 제시할 최적

화된 큐 기반의 NMEA 멀티플렉서를 독립적으로 동작 할 수 있는 모듈을 말한다.

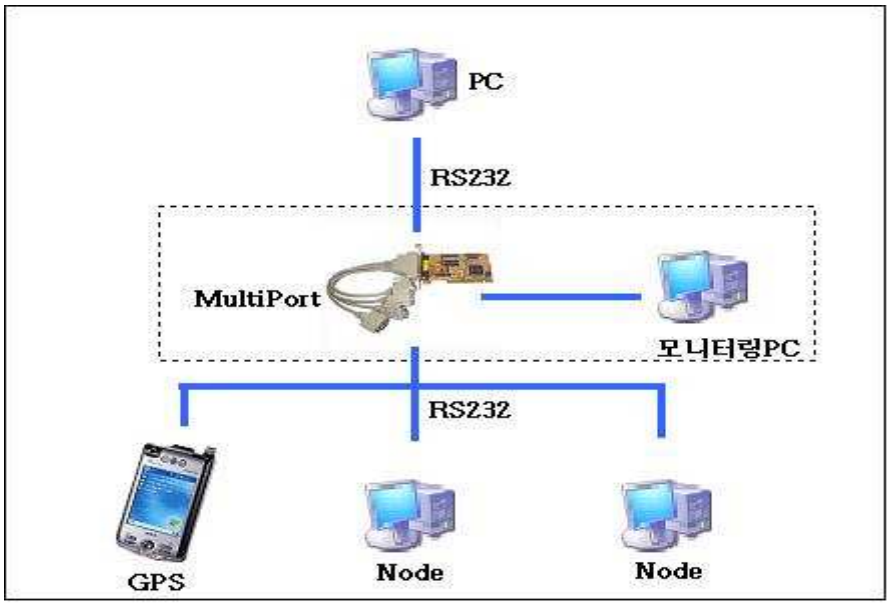
NMEA 멀티플렉서를 설계하기 전 먼저 유념해야 할 사항들이 있다. NMEA 통신 시스템에서 사용될 NMEA 멀티플렉서의 모듈을 구현 한다면, 그 NMEA 멀티플렉서는 모듈에 맞는 한정된 메모리를 가질 수밖에 없다. 따라서 본 시스템을 구현하기 전에 각종 장치들이 보내는 NMEA 신호들을 효율적인 큐 처리로 최소한의 용량으로 최대한 정확한 NMEA 신호를 모니터링 컴퓨터가 받을 수 있도록 큐 설계에 많은 주의를 요할 필요가 있으며, 또한 NMEA 멀티플렉서의 모듈 설계 시 그 적용이 쉬워야 한다.

3.2 NMEA 멀티플렉서의 설계

NMEA 멀티플렉서 설계 시, 시뮬레이션 시스템 구성과 인터페이스 및 큐 구조 설계에 관하여 설명한다. 큐의 설계는 메모리 효율을 높이기 위하여 버퍼의 크기를 수신된 다양한 NMEA 신호의 크기에 따라 가변적으로 변형 될 수 있도록 중점을 두어 설계를 했다.

3.2.1 시뮬레이션 시스템

<그림 3-2>은 시뮬레이션 시스템 구성도로서 NMEA 멀티플렉서의 기능적 구현과 시뮬레이션을 위해서 모듈이 아닌 멀티포트와 모니터링 컴퓨터로 구성한다.



<그림 3-2> 시뮬레이션 시스템 구성도

<Fig. 3-2> Simulation system configuration

<그림 3-1>의 NMEA 통신 시스템 구성과 비교하여 시뮬레이션 시스템 구성은 각 노드 컴퓨터들이 GPS, 풍향 풍속계, 스피드 로그 등 장치들이 되고 상단의 컴퓨터가 자동항해 장치나 NMEA 신호 확인용 컴퓨터를 대신한다. 그리고 점선의 박스로 표시된 부분이 NMEA 멀티플렉서를 대신하여 멀티포트와 모니터링 컴퓨터로 구성할 것이고, 각 장치들과의 인터페이스는 RS232로 구성하였다.

시뮬레이션 시스템의 각 장치들을 기능 정의를 한다면, 각 노드 컴퓨터들은 해당 해상전자장치들의 NMEA 신호를 전송 할 수 있도록 NMEA 신호 형식의 데이터를 만들어 NMEA 멀티플렉서로 전송을 하고, 상단의 NMEA 신호 확인용 컴퓨터는 각 노드 컴퓨터들로부터 전송된 NMEA 신호들이 NMEA 멀티플렉서를 통하여 정상적인 NMEA 신호가 잘 수신이 되는지를 확인 할 것이다. 그리고 NMEA 멀티플렉서의 한부분인 멀티포트는 각 노드 컴퓨터들로부터 NMEA 신호를 받아 집중화 하여 모니터링 컴퓨터로 전송하고 모니터링 컴퓨터에서 최적화된 멀티 큐의 알고리즘에 의해 NMEA 신호의 이상 유무를 판별하여 정상적인 신호들만 출력 할 수 있도록 한다.

3.2.2 인터페이스 설계

(1) 인터페이스

시뮬레이션 시스템의 인터페이스는 컴퓨터에서 통신을 위한 각종 인터페이스의 표준 또는 옵션인 직렬(serial)로 전송하는 RS232를 사용했다. 물론 데이터를 병렬로 보낼 수도 있지만 병렬 통신은 구현하기 힘들고 고가이고, 거리 또한 제한이 된다. 이에 반해서 직렬 통신은 구현하기 쉽고, 저가이며 병렬 보다 거리 제한을 덜 받는다.

그리고 무엇보다도 RS232의 인터페이스 제안은 NMEA 신호의 전송이 직렬 전송으로 되어지기 때문에 NMEA 멀티플렉서가 NMEA 신호를 RS232로 받아야 하고, NMEA 멀티플렉서에서 출력 이후의 인터페이스도 RS232를 사용함으로서 시스템의 전체적인 인터페이스 통일로 시뮬레이션 시스템과 실제 시스템 적용 시 구성을 쉽게 하고자 하였다.

(2) RS232 인터페이스

RS232 인터페이스는 미국의 EIA(Electronic Industries Association)에 의해 규격화된 것으로 정확하게는 EIA-RS232 규격이라고 불리며, 전기적 특성, 기계적 특성, 인터페이스 회로의 기능 등을 규정하고 있다. RS232의 제한 거리는 15M이며 이들 규격은 2개의 송수신 신호선과 5개의 제어선, 그리고 3개의 어스선이 필요하다[9].

RS232 인터페이스 규격은 본래 데이터 단말장치와 모뎀(MOdulator DEModulator:변·복조기)을 접속하기 위한 것으로, 이 경우 캐리어 수신의 확인 등 송신측과 수신측이 모뎀의 상태를 1대 1로 대응시켜서 접속하여야 한다. 컴퓨터 등에서는 RS232의 규격의 일부를 사용하여 그 접속을 간략화하고 있다.

컴퓨터 간의 통신에서는 모든 제어 선을 사용하지 않아도 최소 3개의 송신 데이터선(TxD), 수신 데이터선(RxD) 및 시그널 그라운드선(SG)이 있으면 통신이 가능하지만, 실제로는 커넥터 내부에서 4+5번과 6+8+20번 핀을 연결해 둘 필요가 있다. 이 방식은 컴퓨터 간을 케이블로 직접 접속하는 경우로서 서로 상대방의 하드웨어의 상태를 확인할 필요가 없는 경우의 가장 간단한 인터페이스 방법이다[9],[10].

다음은 <표 3-1>는 RS232의 주요 핀 기능에 대해서 설명한다.

<표 3-1> RS232 주요 핀 기능

<Table 3-1> RS232 main pin function

명 칭	기 능
Protect Ground	외부의 잡음에 대한 보호용 접지선. 보통 7번 선과 연결
DCD (Data Carrier Detect)	DCE(Data Communication Equipment)가 상대편 DCE와 잘 접속되어 있음을 알림
RxD (Receive Data)	데이터 수신 라인
TxD (Transmit Data)	데이터 전송 라인 (DTE 중심의 명칭임)
DTR (Data Terminal Ready)	DTE(Data Terminal Equipment)가 power on 상태임을 알림
SG (Signal Ground)	모든 신호의 기준 전압 (0 volt)
DSR (Data Set Ready)	DCE가 DTE에게 전송할 데이터가 있음을 알림
RTS (Ready To Send)	DTE가 전송할 데이터가 있음을 알리고 허락을 기다림
CTS (Clear To Send)	DTE의 RTS의 요청에 대한 데이터 전송 허락 신호
TxC (Transmit Clock)	TxD 전송에 사용되는 클럭 (필요시만 사용)
RxC (Receive Clock)	RxD 전송에 사용하는 클럭 (필요시만 사용)
DTE Clock	클럭(TxC, RxC)은 DCE가 제공한다. DTE가 클럭을 제공해야 하는 경우 사용됨

<표 3-2>은 본 시뮬레이션 시스템의 인터페이스 구성 시 RS232의 핀 사용을 나타낸 것으로 9핀의 RS232를 사용하였다. 앞에서 언급한 것과 같이 7번 핀(RTS)과 8번 핀(CTS)은 사용하지 않고도 2번, 3번, 5번 핀인 RxD, TxD, SG의 3개의 핀으로도 구성이 가능하다.

<표 3-2> RS232의 핀 사용

<Table 3-2> Pin utility of RS232

핀 번호	명 칭	사 용 설 명
1	DCD	입력. 사용하지 않음
2	RxD	입력. 상대방 TD에 접속
3	TxD	출력. 상대방 RD에 접속
4	DTR	출력. 사용하지 않음
5	SG	그라운드. 상대방 SG에 접속
6	DSR	입력. 사용하지 않음
7	RTS	출력. 상대방 CTS와 접속
8	CTS	입력. 상대방 RTS와 접속
9	RI	입력. 사용하지 않음

3.3 대기-큐 알고리즘

3.3.1 큐의 개요 및 제약 사항

큐는 선형 리스트의 한 종류로서 데이터를 입력할 때는 rear 또는 tail 이라고 불리는 선형 리스트의 한쪽 끝으로 일어나고, 데이터를 출력할 때는 front 또는 head라고 불리는 반대쪽 끝에서 수행 되어진다. 그러므로 rear에서 입력된 데이터는 들어온 순서대로 front를 통해서 출력된다. 다시 말해 먼저 삽입한 것은 먼저 삭제되고, 나중에 삽입된 것은 나중에 삭제되는 선입선출(FIFO : First in First out) 리스트라고 한다[11].

큐의 연산 형태는 스택과 마찬가지로 크게 생성(Create), 삽입(Insert), 삭제(Delete), 큐의 상태를 알아보는 네 가지 함수가 있다.

· create(Q) : Q라는 이름을 갖는 빈 큐를 생성한다.

· empty(Q) : 큐가 비어있는지 아닌지를 조사하는 연산으로 비어있으면 참을 그렇지 않으면 거짓 값을 갖는다.

· insert(E, Q) : 큐 안에 원소 E를 삽입시키는 연산이다.

· remove(Q) : 큐 안의 전면 원소를 제거하는 연산이다.

스택과 마찬가지로 큐도 프로그래밍 언어에서 생성 또는 연산하는 함수를 갖고 있지 않으므로 프로그램 내에서 큐의 특성대로 자료를 처리하여야 한다. 큐를 표현할 수 있는 가장 쉬운 방법은 1차 배열을 이용하는 것이다. 하지만 큐를 배열을 이용해서 표현할 때는 몇 가지 주의해야할 점이 있다. 첫째는 배열의 특성상 큐에 수용되는 자료는 서로 같은 자료 구조여야 한다. 둘째는 큐를 정의할 때 그

크기가 무한할 수도 있지만 컴퓨터의 기억 공간의 유한성 때문에 프로그램에서 큐의 크기를 결정해야 한다는 것이다[12],[13].

프로그램 내에서 변수명 Q인 큐를 나타내기 위한 선언문을 프로그램으로 작성해보면 <그림 3-3>과 같다. 여기서 Q는 1차원 배열로 표시되며 Q의 각 요소는 정수 값을 가지고 배열의 크기는 100으로 제한되어 있다. 또한 Q의 front와 rear를 각각 지시할 수 있도록 2개의 포인트를 설정하였고 이들 2개의 포인트의 변수명을 각각 FRONT, REAR로 표현하였다[12].

```
struct stack_struct {
                int queue[100] ;
                int front = 0 ;
                int rear  = 0 ;
          }
```

<그림 3-3> 큐의 선언

<Fig. 3-3> Declaration of the queue

앞에서 큐의 선언문이 작성된 것을 가지고 큐의 삽입과 삭제 처리 과정의 프로그래밍 언어로 표현해 보면 <그림 3-4>와 같다.

여기서 가장 중요한 것은 포인트로 사용하는 front와 rear가 큐의 안에서 정확하게 자료의 앞과 뒤를 가리킬 수 있게 프로그램에서 조정하여 큐의 밖으로 벗어나지 않게 하여야 한다. 다시 말해 삽입 시에는 overflow가 발생하지 않는지를 삭제 할 때는 underflow가 발생하지 않는지를 살펴보아야 한다[11],[13].

```
void insert(int q[], int data, int n, int rear)
{
     if(rear == n - 1)  {
          printf("Queue is Overflow \n") ;
          exit(1);
     }
     q[++rear] = data ;
}

int remove(int q[], int front, int rear)
{
     if(front == rear)  {
          printf("Queue is Underflow \n") ;
          exit(1);
     }
     return q[front] ;
     front++ ;
}
```

<그림 3-4> 큐의 제약

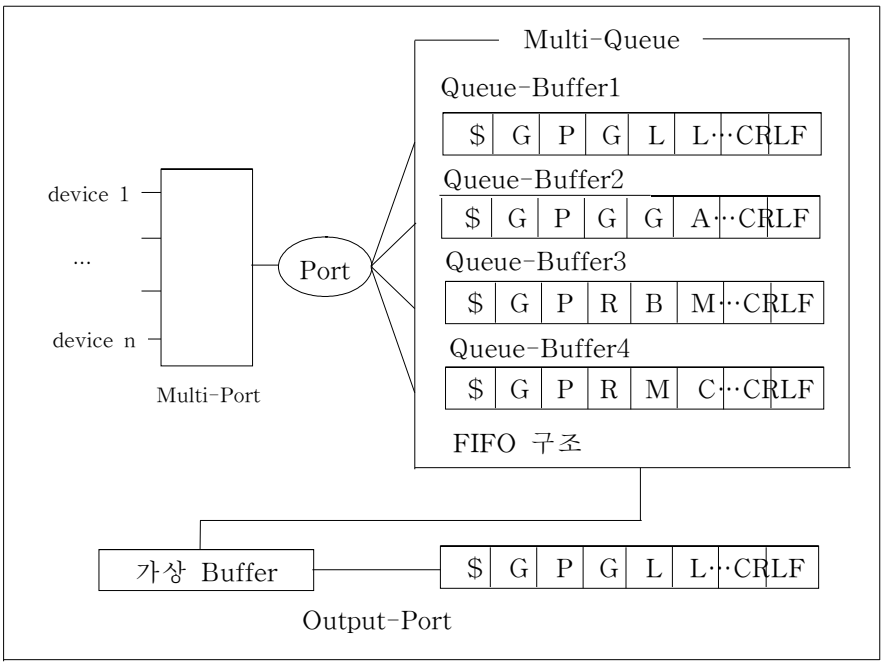
<Fig. 3-4> Restriction of the queue

3.3.2 멀티 큐의 구조 및 신호 처리

본 논문에서 멀티 큐를 사용 하여 최적화 된 큐를 설계 하였으며, 다음 <그림 3-5>는 멀티 큐의 구조를 나타낸 것이다. 멀티 큐에서 여러 개의 device로부터 NMEA 신호가 멀티포터를 통하여 입력되면, 내부 멀티 큐에서 처리하게 된다.

NMEA 신호가 버퍼로 입력 될 때, 효율적인 메모리 관리를 위하여 버퍼의 길이는 가변적인 크기를 가지게 되고, NMEA 멀티플렉서의 특성상 device로부터 오는 데이터 중 최근의 정보가 가장 중요하

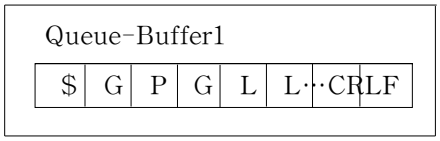
므로, 최근의 정보가 계속 갱신 되도록 신호 처리 알고리즘을 설계하였다.



<그림 3-5> 멀티 큐의 구조

<Fig 3-5> Structure of the multi queue

<그림 3-6>은 큐 버퍼로서 “$”구분자 수를 판별하여 정상적인 데이터인지 잘못된 데이터 인지를 판별하도록 구성되어 있다.



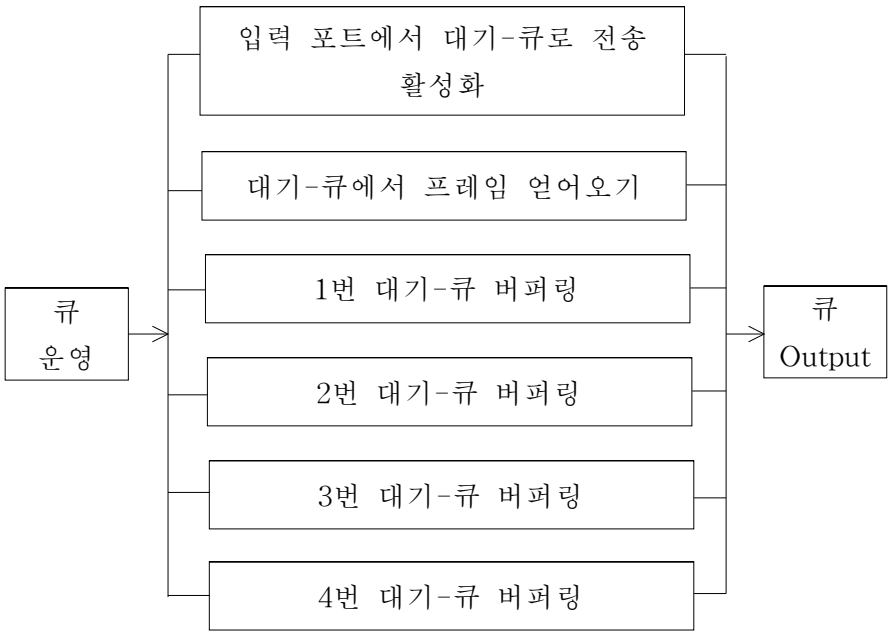
<그림 3-6> 큐 버퍼

<Fig 3-6> Queue buffer

제 4 장 NMEA 멀티플렉서의 구현

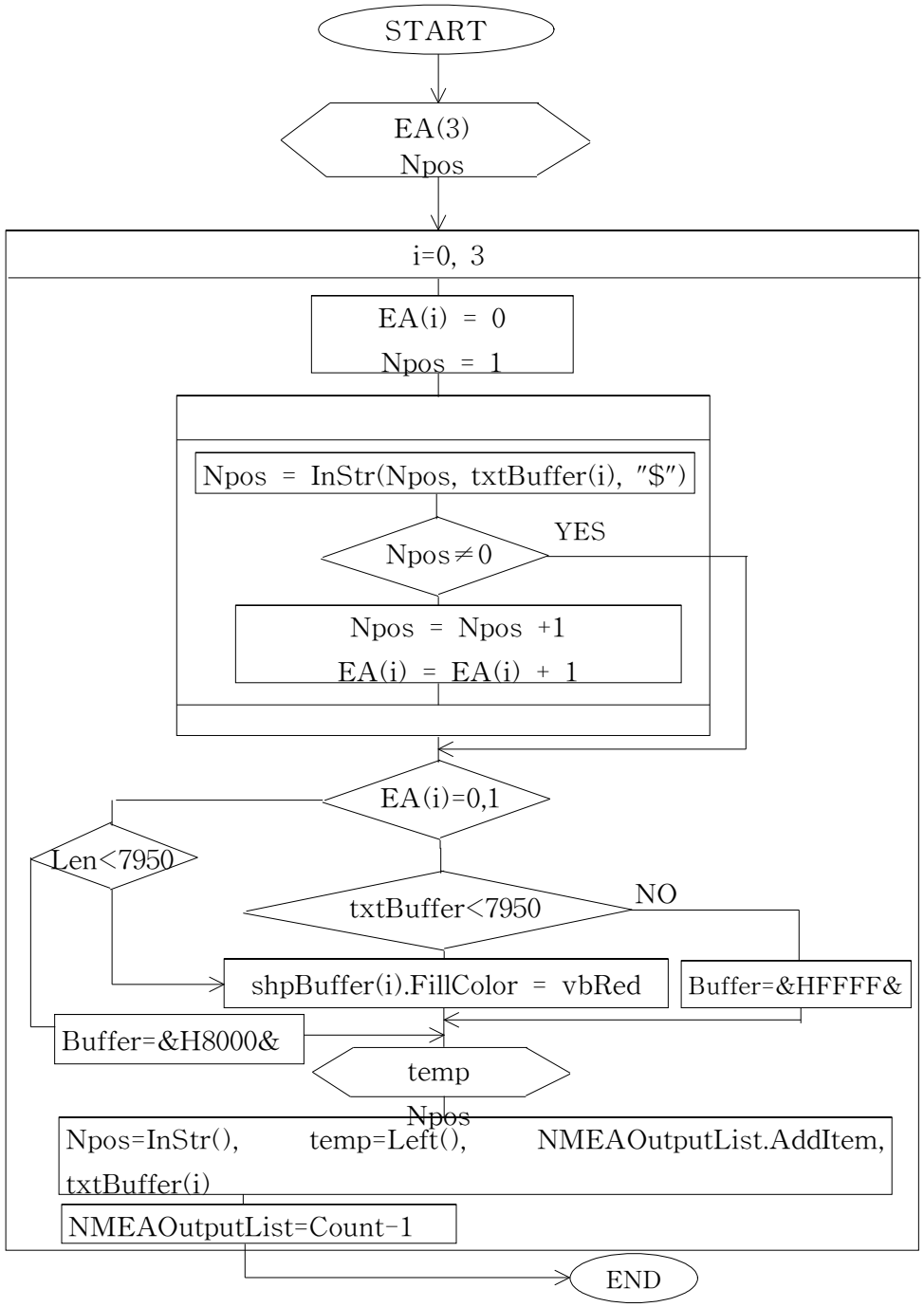
4.1 대기-큐 알고리즘의 구현

각각의 장치들로부터 들어온 데이터를 입력포트로 받아들여 대기-큐를 활성화 시키고, 그 대기-큐에서 프레임을 얻어온 후 각 포트에 해당하는 버퍼에 버퍼링 시켜주는 과정을 <그림 4-1>에서 블록 다이어그램으로 보여주고 있다. 큐의 운영에 있어서 입력 포트에서 대기-큐로 전송을 활성화하면 대기-큐로 프레임이 입력된다. 이때 대기-큐의 버퍼의 크기를 동적으로 할당하여 최적의 상태를 유지하며 변동이 생길 경우 큐의 크기를 재조정한다.



<그림 4-1> 큐 블록 다이어그램

<Fig. 4-1> Queue block diagram



<그림 4-2> 대기-큐로부터 프레임을 얻는 흐름도

<Fig. 4-2> Flow chart wait-queue frame

<그림 4-2>은 대기-큐에서 프레임을 얻어오는 과정을 나타내는 흐름도이다. 프레임을 얻기 위하여 먼저 “EA” 라는 배열을 선언하고 노드의 위치를 식별하기 위한 “Npos” 를 지정한다.

반복 루프내에 해당 버퍼의 개수만큼 처리 루틴을 구현하여야 하며, 서브루틴으로 프로시저를 작성하였다. 주요 부분으로 프레임이 인터럽트가 발생할 때 자동적으로 수신된 바이트만큼 확보된 대기-큐에 입력이 완료되면 검색 루틴에서 “$”신호를 검사한다. 그 이유는 NMEA 신호의 시작문자로 유효 프레임을 식별하기 용이하기 때문이다. 다음에 마지막에 들어오는 CRLF가 포함이 되어 있다면 유효 프레임이 대기-큐내에 존재한다는 것을 의미한다.

다음 과정에서는 정상 프레임과 미완료된 상태의 프레임과 트렁케이션된 프레임을 구분해주는 부분으로 색상을 통하여 화면에 나타내도록 하였다. 이렇게 얻어진 유효 프레임은 단일 출력 포트를 통하여 출력될 수 있도록 이전에 얻어진 유효프레임 저장장소에 계속 추가 시킨다.

<그림 4-3>의 소스는 입력포트에서 대기-큐로 전송 활성화하는 소스로 NMEA 멀티포터에서 입력된 데이터 중 리스트 박스에서 한 줄 얻어온다. 그런 다음 GPS, 에코사운드, 자이로콤파스 등과 같은 데이터 프레임들을 종류별로 버퍼링한다. 각각의 Case문 0번부터 3번까지는 각 포트들로부터 입력될 큐 4개를 나타낸다. 이 Case문에서는 데이터 프레임들이 한 번에 전송될 때, 전송한 프레임의 양(버퍼 크기)을 대기-큐에 뿌려주는 양을 체크 하기위해 코딩된 부분이다. Case문을 빠져 나오면 마지막으로 활성화 큐의 버퍼 량을 체크하고 시뮬레이션 한다.

```
'각 큐에게 프레임 전송하기
Select Case index
  Case 4 '입력포트에서 대기큐로 전송 활성화
    Counter = Counter + 1
    NMEAmsgList.ListIndex = Counter

    '리스트박스에서 한 줄 얻기 (실제: NMEA멀티포트에서 입력됨)
    TempList = NMEAmsgList.List(NMEAmsgList.ListIndex)
    Select Case Left(TempList, 6)
      Case "$GPGSA", "$GPGGA", "$GPGLL": N = 0
      Case "$GPRMB", "$GPRMC":           N = 1
      Case "$GPAPB":    N = 2
      Case "$GPGSV":    N = 3
    End Select

    '종류별 버퍼링하기(실제: GPS, Echo Sounder, Gyro Compass, Radar 등등)
    ArrIN(N) = ArrIN(N) + TempList
    'Debug.Print ArrIN(N)
    shpBuffer(N).Width = Len(txtBuffer(N)) * BIYUL
  Case 5 ' 대기큐에서 프레임 얻어오기
      Dim EA(3) As Integer '$의 개수
      Dim Npos As Integer '조사 위치

      '완전한 프레임이 있나 조사
      For i = 0 To 3
         '각 대기큐별 $의 개수를 모두 조사함
         EA(i) = 0
         Npos = 1
         Do
            Npos = InStr(Npos, txtBuffer(i), "$")
            If Npos = 0 Then Exit Do
            Npos = Npos + 1
            EA(i) = EA(i) + 1
         Loop
```

<그림 4-3> 각 큐에 프레임 전송하기

<Fig. 4-3> Transfers to frame on queues

대기-큐에서 프레임을 얻어오는 소스를 살펴보면, 우선 각각의 프레임 구분자인 “$”의 개수를 정의하는 “EA” 배열과 조사의 위치를 위한 변수를 정의한다. 그 다음으로 들어온 데이터 프레임 중 완전한 프레임이 있는지 조사하기 위하여 For문을 사용하고 각 대기-큐별 ’$‘의 개수를 모두 조사한다. 유효한 프레임이 있는지 조사하여

각 버퍼의 색상을 결정하여준 뒤 유효한 프레임을 파싱하여 출력하고 출력 리스트박스 하단에 보여주는 소스구조로 이루어져 있다.

<그림 4-4>의 대기-큐에서 프레임을 얻어오는 소스코드는 디자인 폼에 배치된 컴포넌트인 텍스트박스를 컨트롤 배열화 하여나타내었다.

```
Sub CompleteFrameCheck()
        Dim EA(3) As Integer '$의 개수
        Dim Npos As Integer '조사 위치

        '완전한 프레임이 있나 조사
        For i = 0 To 3
            '각 대기큐별 $의 개수를 모두 조사함
            EA(i) = 0
            Npos = 1
            Do
                Npos = InStr(Npos, txtBuffer(i), "$")
                If Npos = 0 Then Exit Do
                Npos = Npos + 1
                EA(i) = EA(i) + 1
            Loop

            '버퍼의 색상 결정
            Select Case EA(i)
              Case 0, 1 '유효한 프레임이 없음
                 If Len(txtBuffer(i)) > 99 Then
                    shpBuffer(i).FillColor = vbRed
                 Else
                    shpBuffer(i).FillColor = &HFFFF&
                 End If
              Case Else '유효한 프레임이 존재함
                 If Len(txtBuffer(i)) > 99 Then
                    shpBuffer(i).FillColor = vbRed
                 Else
                    shpBuffer(i).FillColor = &H8000&
                 End If
            End Select

            '버퍼량 다시 보여주기
            N = Len(txtBuffer(i))
            shpBuffer(i).Width = N * BIYUL
            Label(i) = Str(N)
        Next
End Sub
```

<그림 4-4> 대기-큐에서 프레임 얻어오기

<Fig. 4-4> Get frame to wait-queue

이 컨트롤 배열의 이름은 txtBuffer로 시뮬레이션에서는 4개로 설정하였으며, 각 텍스트박스 컨트롤 버퍼에 표시되는 NMEA 신호를 시각적으로 표현하기 위하여 스트림 데이터를 직접 나타내었다. 따라서 보여지는 데이터내에 “$”를 검색하여 대기-큐에서 프레임을 얻어올 수가 있다. 유효 프레임의 크기가 결정되면 첫 바이트부터 검색된 “$” 식별문자의 위치 값까지 파싱하여 정상 프레임을 얻게 된다.

4.2 시뮬레이터의 모델링

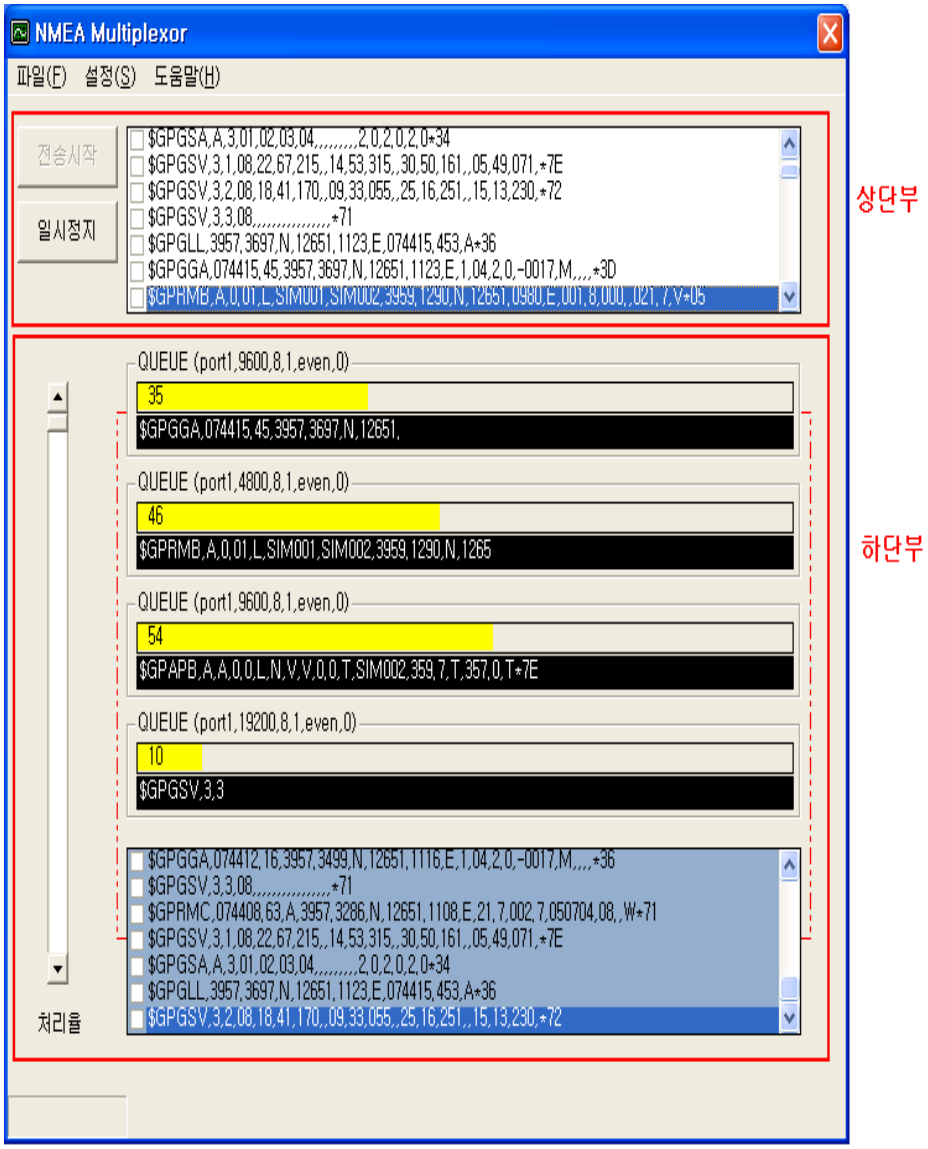
자이로콤파스나 풍향풍속계와 같은 장치들을 연결하여 시스템을 구현하기 위한 시뮬레이션으로 각 장치들이 NMEA 신호 확인용 컴퓨터에 보내는 NMEA 신호들을 각 장치에 해당하는 노드 컴퓨터가 자이로콤파스나 풍향풍속계와 같은 장치들을 대신하여 보내도록 <그림 3-2>와 같이 시스템을 구성하였다. 이 노드 컴퓨터들이 멀티포트로 NMEA 신호들을 보내면 여러 포트별로 들어온 NMEA 신호들을 취합하여 하나의 출력으로 묶어 NMEA 신호 확인용 컴퓨터에 데이터들을 보내게 되는데 이때, 멀티포트에 연결되어 있는 모니터닝 컴퓨터에 확인에서는 들어온 데이터들을 버퍼 상에 저장한 뒤, 다시 이 데이터들을 파싱 한다든지 하는 과정을 겪게 된다. 또한 이러한 과정에서 각 장치에 해당하는 노트 컴퓨터들이 보내는 신호 전송속도 보다 처리하는 속도가 느리다고 한다면 이때는 데이터들이 버려지는 경우가 생기게 된다. 이 때 최근 정보를 계속 유지하기 위해서라든지, 버려지는 데이터들의 양이라든지 하는 것을 확인 할 필요가 있다. 이를 위해 모니터링 컴퓨터상에 NMEA 멀티플렉서라는 프로그램을 구성 하였다.

4.2.1 시뮬레이터의 화면 구성

전체적인 화면 구성은 <그림 4-5>과 같다. 각각의 노드 컴퓨터들이 보내온 데이터들을 에러 없이 정상적으로 처리하고 있는지, 들어온 데이터들이 어떤 내용인지를 확인하고 가장 효율적인 처리를 할 수 있는 처리 상태를 알 수 있게 설계되어 있는 폼 구조 이다.

<그림 4-5>의 메인 폼 상단의 신호는 GPS 수신기를 통하여 수

신되고 있는 NMEA 신호이며, 멀티포트와 연결되어 있는 기타 장비로부터 입력되는 신호를 포함한다.



<그림 4-5> 모니터링 PC의 메인 폼

<Fig. 4-5> Main form of monitoring PC

4.2.2 시뮬레이터의 기능

시뮬레이션을 하기 위하여 대기-큐에 대한 정의가 필요하다. <그림 4-6>의 소스코드에서는 NMEA 파일에 대한 입출력을 지정하는 openFile과 saveFile을 지정하고, 큐의 사용을 위한 임시 버퍼 역할을 하는 ArrN 배열을 확보한다. 이렇게 확보된 대기-큐의 임시 버퍼를 실제 대기-큐 알고리즘을 이용하기 위한 스트림 데이터 처리 배열인 WQ(Wait Queue)가 선언된다.

```
Option Explicit

'외부의 오픈할 NMEA 파일 이름
Public openFile As String

'외부로 저장할 NMEA 파일 이름
Public saveFile As String


'시뮬레이션을 위한 NMEA 입력용 데이터 임시 배열
'멀티플렉서이기 때문에 입력은 N
Public ArrIN(4) As String

'시뮬레이션을 위한 NMEA 출력용 데이터 임시 배열
'멀티플렉서이기 때문에 출력은 1
Public ArrOUT As String


'멀티포트 지원을 위한 대기 큐(Queue) * 큐 사이즈 1000 Byte
Public WQ(4) As String * 1000

'리스트 박스 인덱스
Public Counter As Integer
```

<그림 4-6> 큐의 선언

<Fig. 4-6> Declaration of the queue

WQ는 시뮬레이션을 하기 위해 임시로 4개를 지정하였으며, 최대 버퍼 사이즈는 고정크기인 1,000 Bytes로 확보하였다. 차 후 Redim 명령을 통하여 재지정을 통한 유동적으로 버퍼의 크기를 조정한다.

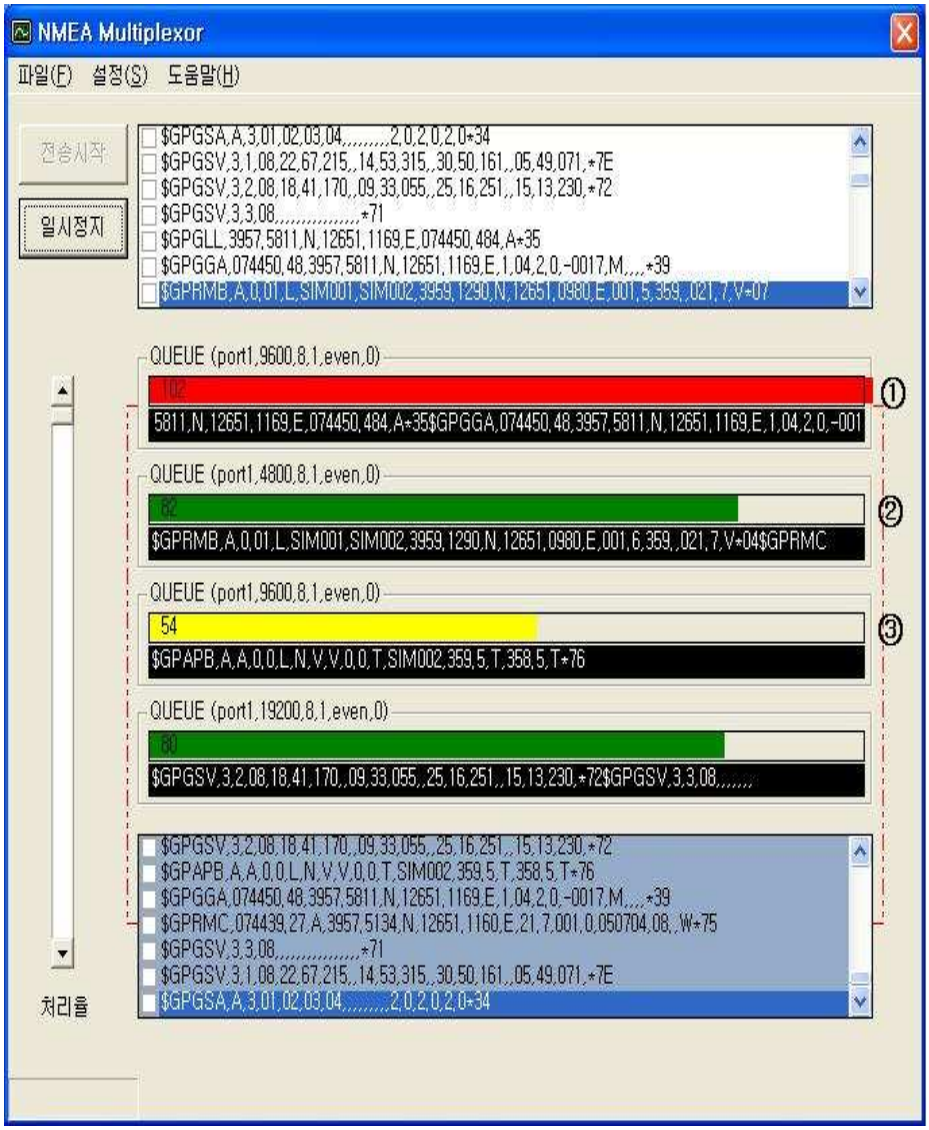
폼의 상단부에 있는 [전송 시작] 버튼에 이벤트가 발생할 경우, 각각의 노드 컴퓨터들과 일제 연동하여 NMEA 신호를 텍스트 박스에 추가하여 리스트박스 창에서 그 신호를 보여주게 된다. 프로토타입을 정의하기 위해서 먼저, 시뮬레이션 모델링을 할 수 있는 파일에 NMEA신호에 대한 내용을 다중화하여 읽어 들인다. 그 후, 타이머들은 정해진 대기-큐들을 활성화 시킨다.

<그림 4-5> 폼의 하단부 구성은 멀티 큐의 운영에 대한 시뮬레이션이다. 본 실험에서는 4개의 대기-큐를 사용하여 배치하였다. 대기-큐의 처리량에서 대기-큐 버퍼로부터 정상 프레임이 발견되었을 때, 파서에 의해서 파싱된 문자열을 버퍼에 연속적으로 추가하여 나타낸 모습이다. 밑의 텍스트 박스는 실제로 채워지는 텍스트 스트림이다. 이 실험에서는 처리량에 대한 최적화 처리율 프로그레스바로 버퍼 사이즈 변화량을 확인할 수 있다.

<그림 4-5>는 대기-큐에 버퍼링되는 스트림이 정상 처리 안 될 때, 정상프레임을 파싱할 수 있는 상태, 버퍼의 크기보다 프레임이 커서 트렁케이션 현상(잘림 현상)을 나타내고 있다. 정상 처리 안 될 때는 Shape가 노란 색(그림 4-5의 ①)으로, 정상처리 될 경우는 Shape가 녹색(그림 4-5의 ②)으로, 트렁케이션 현상이 발생했을 때는 Shape가 빨강색(그림 4-5의 ③)으로 표현되어 있다.

트렁케이션 현상이 발생하였을 경우 해결책으로 유효 프레임을 다시 받기 위하여 손실된 프레임을 버리고 버퍼를 비워준다. 만약 유효 프레임과 손실된 프레임이 존재할 경우 유효 프레임을 인증하기 위하여 NMEA 신호의 마지막 부분의 순환 중복 검사를 위한 바이트를 파싱하여 앞 부분의 데이터와 연산을 수행한다. 이때 오류 검

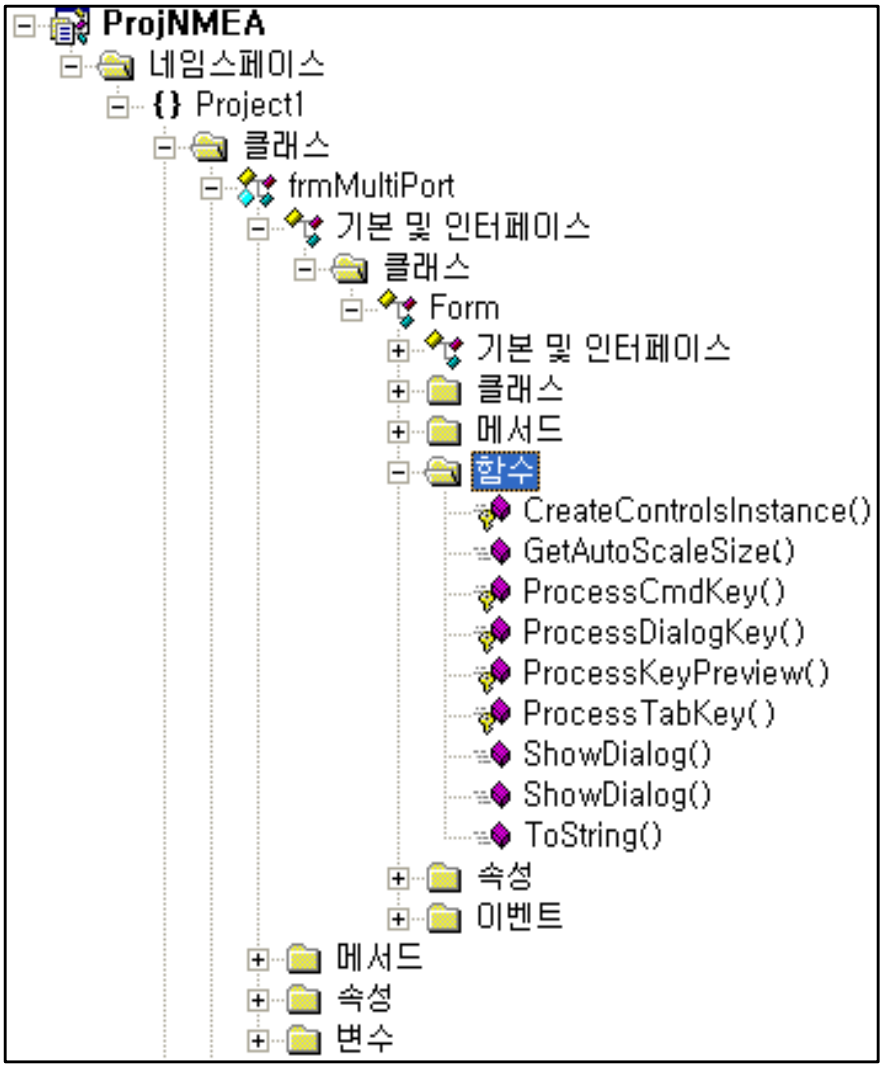
사에서 정상적인 프레임이 존재할 경우 대기-큐 알고리즘에 따라 처리하게 된다.



<그림 4-7> 대기-큐 처리와 버퍼의 잘림 현상

<Fig. 4-7> Wait-queue and truncation of buffers

<그림 4-7>은 NMEA 멀티플렉서 프로그램을 구현하기 위해 사용된 함수들을 보여주고 있다. 해당되는 이벤트가 발생되었을 경우 그림에 있는 함수들의 동작으로 인해 프로그램이 구동 되어진다.



<그림 4-8> 폼 구성을 위한 소스 함수들

<Fig. 4-8> Source functions for configuration of form

그리고 주요 함수를 보면 컨트롤 인스턴스인 컴포넌트의 생성과 프로시저의 구현에 대한 리스트를 확인할 수 있다. 기본적으로 프로젝트를 구성하는 네임스페이스 영역과 프로젝트를 구성하는 기본 및 인터페이스, 메서드, 속성, 변수 등으로 나열된다. 기본 인터페이스는 파상 클래스로 메인 폼을 나타내는 Form이 있으며, Form 자체가 클래스이므로 클래스에 대한 기본 및 인터페이스부, 클래스 정의부, 메서드 선언부, 함수, 속성, 이벤트로 구성된다.

4.3 실험 및 고찰

프레임의 실시간 처리를 위하여 구축한 시뮬레이션 노드들과 GPS 수신기, 특정 NMEA 신호들을 보내주는 노드 컴퓨터로 구성하였다. 다중 장치들로부터 동시에 입력되는 NMEA신호들은 멀티포트를 통하여 구현된 시뮬레이션에서 신호별 파싱과 유효 작업 분류 등을 통하여 다중화 되어 정상적인 신호를 처리하는 과정을 보였다.

<표 4-1>은 NMEA 가상 시그널과 GPS 수신기로부터 수신되는 입력데이터를 1시간동안 수집하고 대기-큐 알고리즘을 통하여 처리된 결과로서 포트별 전송속도와 버퍼크기를 지정하고 입력된 프레임의 유효 데이터를 검출한 것이다.

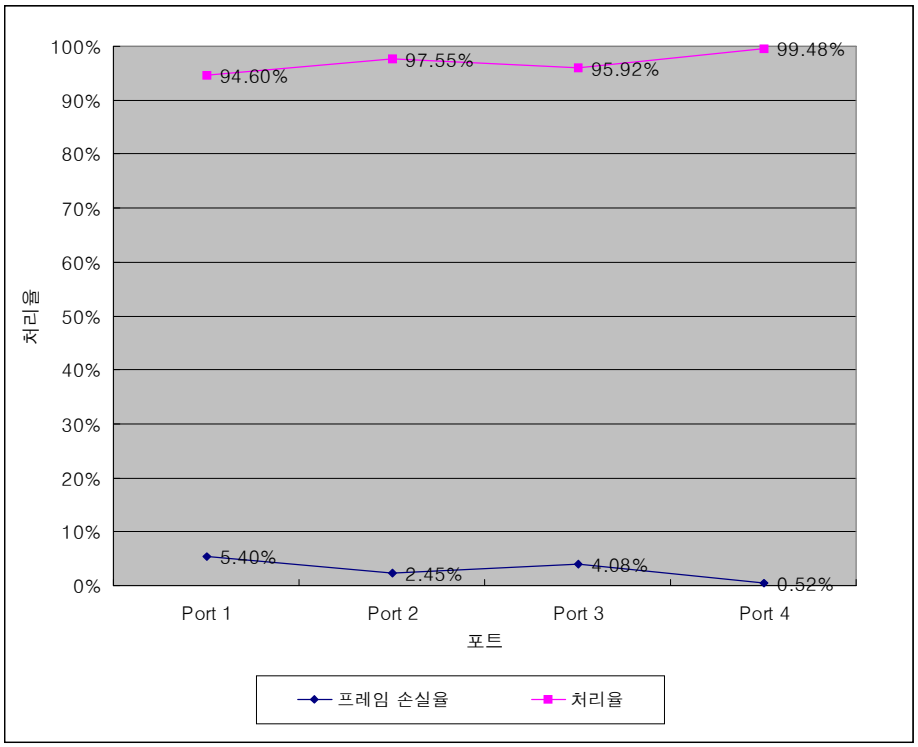
<표 4-1> 프레임 손실율과 처리율의 변화

<Table 4-1> Frame loss rate and change of processing rate

구분	Port 1	Port 2	Port 3	Port 4
전송속도	9600	4800	9600	19200
버퍼크기	1000	1000	1000	1000
프레임 손실율	5.40%	2.45%	4.08%	0.52%
처리율	94.60%	97.55%	95.92%	99.48%

프레임의 손실율은 버퍼의 최대크기를 초과하는 데이터 스트림이 발생할 경우 생기는 트렁케이션 현상으로 버려지는 프레임이 발생한 빈도이다. 처리율은 수신되는 입력 데이터를 파싱하여 유효 프레임을 얻었을 경우 대기-큐에서 단일 포트로 복사된 결과로서 손실되지 않는 정상 프레임의 추출에 대한 비율을 나타낸다.

<그림 4-9>에서는 처리율과 프레임 손실율의 관계를 나타내는 그래프로 처리율이 높을수록 프레임의 손실율은 낮아지는 것을 확인할 수 있다. 버퍼의 크기를 크게 하여 처리율을 높여 줄수록 프레임 손실율 없이 들어오는 데이터 신뢰성은 높지만, 전송시간이 길어진다는 단점이 있다. 따라서 최적의 전송시간 대비 적은 프레임의 손실율을 나타내려면 동적 큐의 할당 기법을 통해 많은 대역폭이 필요한 장치에 대하여 충분히 수용할 수 있는 크기의 버퍼 량을 부여하고, 전송속도가 느리거나 부하가 적은 포트에 해당하는 큐에게는 해당 크기를 수용하는 정도의 크기를 할당함으로써 적절한 메모리 관리가 가능하고 부하를 줄일 수가 있다.



<그림 4-9> 처리율 변화에 따른 프레임 손실률

<Fig. 4-9> Frame loss rate according to change processing rate

또한 장치들로부터 들어오는 데이터프레임의 크기보다 큐의 버퍼를 크게 한다면 낭비되어지는 공간이 발생하고 이는 많은 메모리를 낭비한다는 것을 의미한다. 그렇다고 무작정 처리율을 줄여 준다면 이는 버퍼의 크기가 줄어들게 되기 때문에 버퍼의 크기보다 큰 데이터 프레임들은 다 버려지는 문제가 발생한다.

본 논문에서는 버퍼의 크기를 조절할 수 있도록 큐의 크기를 가변적으로 처리했기 때문에 적당한 큐 크기를 설정할 수 있다면 최소의 프레임 손실과 많은 공간의 메모리를 확보할 수 있다는 장점이 있다.

제 5 장 결 론

해상에서 작업하는 장비들이 다양해지고 더욱 많은 센서 설비들로부터 입력되는 신호들을 처리하기 위하여 수용하는 장비들의 경우 많은 수의 포트를 지원해야 한다. 그러나 여건상 제한적인 포트를 지원할 경우 수많은 장비들을 연결하기 위하여 별도의 멀티 포트의 기능을 수행하는 장치가 필요하다.

본 연구에서는 4개의 멀티 큐와 가상 멀티 대기-큐를 설계하였고, 큐의 운영 알고리즘의 기법 적용과 가변 크기의 메모리를 동적으로 할당하여 적은 자원 환경에서도 다수개의 포트로부터 입력되는 신호들을 최소의 손실을 유지하고 단일의 포트로 신호를 통합하여 출력하여 주는 시스템의 시뮬레이션을 연구하였다.

입력되는 신호의 대역폭에 비례하여 수용하는 장치의 병목현상을 방지하고 최적의 신호처리를 수행할 수 있도록 하기 위한 대기-큐 기법의 연구와 버퍼링의 방법론들을 연구하고, 입력되는 다수 개의 포트로부터 신호를 처리하여 단일 또는 다수개의 포트로 출력시켜 줄 수 있는 시스템에 대한 시뮬레이션을 통하여 최적의 요건을 만족하는 경우 이상적인 출력을 기대할 수가 있었다.

NMEA 멀티플렉서가 임베디드 환경에서 개발될 경우 제한적인 메모리의 양에 따라 대기-큐의 운영 기법이 최적화 될 수 있도록 하기 위해서는 가변적인 큐의 운영 방식이 필수적이며, 처리량의 분산 기법을 통하여 최적의 메모리 운영 계획을 세울 수 있었다.

실험 결과에서 얻어진 통계를 분석해보면 실제 큐의 크기를 재조정할 때 트래픽의 부하를 줄이고 신뢰성을 높일 수 있는 방법에 있어서 동적으로 재할당하는 과정이 고정된 크기의 큐 버퍼를 이용한 방법보다 높은 효율과 프레임 처리 향상을 가져왔다.

향후 계획으로 지금의 연구에서 얻어진 결과를 토대로 임베디드 보드를 설계하여 실제 현장에서 사용될 수 있도록 하드웨어 모듈을 구현하고, 최적의 메모리를 활용한 설계와 큐 운영 기법의 적용으로 저비용, 고효율의 NMEA 멀티플렉서 장비를 개발해보고자 한다.

참 고 문 헌

[1] NMEA0183 Version 2.00, STANDARD FOR INTERFACING MARINE ELECTRONIC DEVICES.

[2] 조현덕 Homepage, "URL: http://pcptpp030.psychologie.uni-regensburg.de/trafficresearch/NMEA0183/", .1998.5.

[3] 덕청강 Homepage, "URL:http://biology.gsnd.net/NMEA.html"

[4] 김익수, 디지털 공학, 대림 출판사, 1998.

[5] 대한전자공학회, 디지털회로 및 시스템 실험, 청문원 출판사, 1998.

[5] GPS 연구실(건국대) "URL: http://gpslab.konkuk.ac.kr/", 2001

[6] D. Kozlov, M. Tkachenko, "Instant RTK cm with Low Cost GPS+GLONASS Receivers", Proc. of ION GPS-97, pp.1559-1569, 1997. 3.

[7] RTCM SPECIAL COMMITTEE NO.104 By Developed, "RTCM Recommended Standards for Differential Navstar GPS Service, Version 2.1", 1994. 1.

[8] Hoffman-Wellenhof, B., Lichtenegger, H., and Collins, J., "Global Positioning System Theory and Practice", Springer Wien New York, 1997.

[9] 가남사 편집부, RSC232C 인터페이스 사용법, 가남사, 1990.

[10] John Catsoulis, 임베디드 하드웨어 이해와 설계, 한빛미디어, 2003. 10.

[11] 이석호, C로 쓴 자료구조론, 회중당, 1996.

[12] Custom Computer Services Inc. C Compiler Reference Manual, "URL: http://www.ccsinfo.com/", 2002. 2.

[13] 오용철, 자료구조, 이프레스, 2003.

[14] Peter Bennett Homepage, "URL:http://www.vancouver-webpages.com/peter"

[15] SiRF Technology, Inc. NMEA Reference Manual, pp.1-5, 2002.

[16] 정태영, 볼랜드 C++ Builder 컴포넌트 프로그래밍, 가남사, 1999.

[17] 주경민, 박성완, 김민호, Visual Basic Programming Bible Ver. 6.x, 영진출판사, 1999. 11.

[18] 정일홍, 이경휘, 단계별 실습으로 배우는 Visual C++ 6.0, 생능출판사, 2002. 8.

[19] 정성훈, 이태오, 임재홍, “PDA 환경에서 RTK-GPS 보정 데이터 전송 에이전트의 설계 및 구현”, 한국 정보 처리 학회 제17회 춘계 학술 발표 대회, 2002. 4.

감 사 의 글

대학원 문을 열었을 때 열심히 공부하리라 다짐하고 들어왔지만 문을 다시 열고 나가려고 하니 아쉬움만 남습니다. 항상 목표를 정하고 그 목표를 향해서 노력하며 나름대로 열심히 살아온 나였지만 이번 대학원 생활만큼은 힘든 일은 없었든 것 같습니다. 학교도 자주 못가고 공부도 이것저것 조금씩, 실력이 없으면 노력이라도 해야 되는데 열심히 못한 저를 스스로 반성해 봅니다. 그래도 대학원 생활을 하면서 공부가 참 힘들고 대충대충 이라는 것이 없다는 것을 이 논문을 쓰면서 많이 느끼고 앞으로 대충하는 공부는 사절이라고 마음깊이 새기고 무슨 일이든 충실히 해야겠다고 다짐합니다.

본 논문을 쓰기까지 지도해 주신 임재홍 교수님께 감사하고 죄송하다는 말을 함께 전하고 싶습니다. 많이 부족한 점들을 너그럽게 넘어가 주시고 꾸중보다는 침묵으로서 용서 해주시는 것이 더 죄송하게 느껴집니다. 아마도 다른 별 말씀을 없었던 것이 제가 회사를 다니며 공부한다고 많이 이해해 주신 것이라 잘 알고 있습니다. 그리고 일일이 저의 논문을 지적해 주시고 관심을 보여 주신 이상배 교수님과 손경락 교수님께도 진심으로 감사드립니다.

아울러 저의 논문을 완성하기 까지 많은 도움을 준 성훈이 형과 창수형, 그리고 항상 내꺼 같이 저의 논문에 대해서 걱정 해주고 도와준 시형씨에게 고마움을 전하며, 많이 귀찮게 했던 원희씨, 말벗이 되어준 슬기씨, 그 외 연구실 사람들에게도 깊은 관심 고맙게 생각합니다. 그리고 공부하면서 힘들 때 마다 힘이 되어준 정래형, 세훈이, 봉욱이, 왕따애들, 항상 공부한다는 핑계로 회사 업무를 소홀히 하여 패를 끼쳐 드렸던 부산세관 컨테이너 검색기 직원 분들께도 깊은 감사를 드립니다.

항상 논문은 내가 도와줄게 말만하고 논문 출력만 해준 경미, 고맙다고 말해야 하는 건지 모르겠지만, 어쨌든 항상 옆에서 제가 잘 되기만을 기도한다는 것을 알기에 고마움을 전하며 그리고 물심양면으로 도와준 저의 어머니, 형들에게도 고마움을 전합니다.

끝으로 어머니 같이 저를 돌봐주시고 걱정 많이 해준 하늘에 계신 할머니께 깊은 감사를 드리며, 저에게 있어서 모든 일과 생활을 할 수 있게 정신적인 힘이 되어준 저의 아버님께 감사하고, 보고 싶고, 사랑한다는 말을 하늘에 부르며 이 논문을 바칩니다.