⊙ 速報研究論文

선박 주기관 원격제어시스템을 위한 실시간 제어알고리즘 구현에 관한 연구

정경열* · 김종화** · 류길수**

A Study on Implementation of a Real-Time Control Algorithm for Ship Main Engine Remote Control Systems

K. Y. Chung · J. H. Kim · K. S. Rhyu

Key words : Main Engine Remote Control System(주기관 원격제어 시스템), Real-time System(실시간 시스템), Multi-Task(멀티 태스크), Event-driven Method(이벤트구동 방식), Round-robin Method(라운드로빈 방식), Speed Control(속도제어)

Abstract

This paper presents a real-time control technique for the development of a ship main engine remote control system. In general, several tasks are executed by the event-driven method in real-time systems. However, when some tasks have time delay components, it is difficult to achieve good real-time performance. To cope with this problem, a number of timers in most conventional systems have been used. In this paper, we introduce a real-time control methodology of dealing effectively with tasks including time delay components using one hardware timer. And also, a speed control method of main engine which includes critical revolution range, a crash astern and a emergency ahead function, a switching method of remote control position, and a flickering method for the indication of multi-stage alarm are discussed. As long as functions and methods are implemented as forms of tasks, the development of main engine remote control systems can be easy for different types of engines.

1. 서 론

주기관 원격제어시스템이란 선박주기관을 멀리 떨어진 기관제어실이나 선교로부터 조종하기 위한 실시간 시스템을 말한다. 이 시스템은 주기관의 기본적인 제어, 즉 주기관의 시동 및 정지, 비상

* 한국기계연구원(원고접수일 : 98년 10월)
** 한국해양대학교

정 · 역회전, 증 · 감속의 6가지 운전조작을 기본 기능으로 하고 있으며, 이외에도 주기관의 이상운전시의 자동감속 및 자동정지, 위험진동대역의 자동회피 등의 안전기능을 포함하고 있다.[1],[2]

최근에 들어서는 디지털 기술의 발달과 함께 노르웨이, 일본, 독일 등의 회사에서 이러한 주기관 원격제어 시스템을 마이크로프로세서 기반형 시스템으로 개발하여 생산하고 있다.[3],[4] 각 회사의 시스템들은 독특한 방식에 의하여 설계되어 있을 뿐만 아니라, 제어 알고리즘도 통일된 규격 없이 각기 달라 이들을 비교하여 최적인 기법을 찾는 것은 거의 불가능하다.

일반적으로 실시간 시스템은 시스템에 필요한 여러 기능들을 태스크별로 분류하여 이들을 이벤트구동 방식, 즉 각각의 태스크가 메시지를 발생하였을 경우에 실행시켜 주도록 구현된다.[5]~[8] 그러나 시스템에 따라서는 태스크수가 많고 시간지연을 요구하는 경우가 있다. 그렇다고 하나의 태스크에 시간지연을 포함하여 너무 많은 실행시간을 할당하여 주면 다른 태스크의 실행에 지장을 초래하므로 이와 같은 시간지연을 포함한 태스크에 대해서는 인터럽트에 의한 타이머를 이용하여 백그라운드로 실행이 되도록 구현하지 않으면 안된다.[9],[10] 지금까지 개발되어 있는 주기관 원격제어 시스템들은[3],[4] 타이머 이용을 위해 하드웨어적으로 전용 타이머소자를 사용하는 방법을 채택하고 있으며, 이런 경우 하드웨어가 복잡하게 된다.

본 논문에서는 하나의 타이머를 사용하여 시간지연을 가진 태스크들을 처리하여 주는 실시간제어방법에 대하여 논한다. 이외에도 주기관의 속도제어 방법, 특히 임계 회전수영역이 존재하는 경우의 속도제어 방법, crash astern, emergency ahead, 제어위치 절환방법, 경보를 다단계로 구별하여 표시하기 위한 플리커링(flickering)의 구현에 대하여 논하기로 한다.

2. 실시간제어방법

제어를 수반하는 실시간 멀티태스크 시스템에서는 주로 이벤트구동 방식을 이용하고 있다. 이러한 시스템에서는 실행중이던 태스크를 일시적으로 정지시킨 후 다른 태스크에 실행권을 넘겨주게 되므로 태스크 절환시에 발생하는 오버헤드시간이 수반된다. 따라서 빈번하게 이벤트가 발생하게 되면 많은 오버헤드시간이 수반 되므로 실시간 제어 성능을 저하시키는 요인이 된다. 또한 태스크내에 시간지연 요소를 포함하고 있는 경우에는 하나의 태스크에서 많은 시간을 소비하게 되므로 역시 실시간 성능이 떨어지게 된다.

따라서 본 논문에서는 이벤트구동 방식의 단점을 보완하기 위해 전체시스템을 독립적으로 동작해도 무방한 기능들을 하나의 태스크로 하여 이들을 순차적으로 실행시키는 라운드로빈 방식을 도입한다.[6] 이때 하나의 태스크의 실행시간이 너무 길면 다른 태스크의 실행에 문제가 발생하므로 이에 대한 보완책을 마련해야 한다. 실행시간이 길게 되는 이유로는 ㉮태스크의 전체 실행시간이 긴 경우와 ㉯시간지연을 포함하는 경우가 있다.

㉮에 대한 보완책으로는 다음과 같은 원칙에 의해 각 태스크를 다시 서브태스크로 나누어 이것들을 라운드로빈 방식으로 실행권을 이양받았을 때 한번에 하나의 서브태스크만을 실행시켜주는 방식을 도입한다.

(1) 각 태스크가 논리적으로 분리되지 않는 범위까지를 하나의 서브태스크로 한다.
(2) 시간지연을 요구하는 부분까지를 하나의 서브태스크로 한다.
(3) 충분히 적은 실행시간까지를 하나의 서브태스크로 한다.

예를 들어 독립적으로 실행가능한 태스크 A, B, C가 존재하며, A 태스크의 서브태스크로는 A1, A2가 있고, B 태스크의 서브태스크로는 B1, B2, B3가 있고, C태스크의 서브태스크로는 C1, C2, C3, C4가 있다고 가정할 때, 전체시스템은 Fig. 1의 번호 순서에 의해 실행하게 된다.

㉯의 경우에는 하나의 태스크에 너무 많은 실행시간이 할당되어 다른 태스크의 실행에 지장을 초래하게 되므로, 이에 대한 보완책으로는 시간지연에 대한 계산을 인터럽트에 의한 타이머방식으로 처리해 주는 특권 타이머태스크를 만들어 이 태스크에서 지연 시간을 계산하도록 하고 지연 시간이

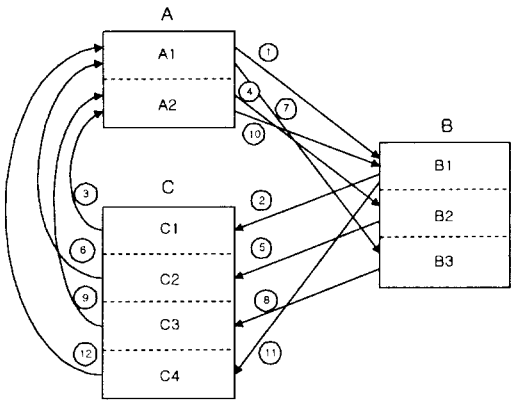

Fig. 1 Execution sequence by sub-tasks of A, B and C tasks

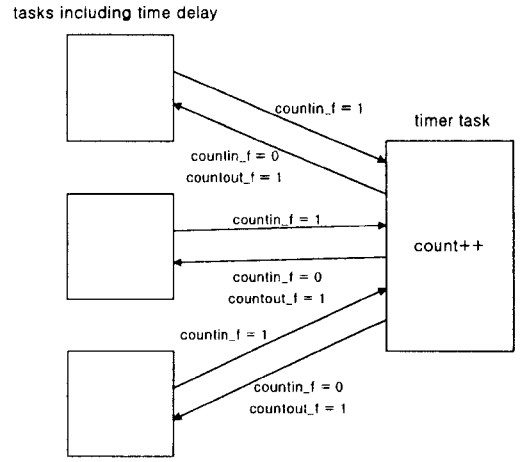

Fig. 2 Relation between the timer task and tasks including delay time

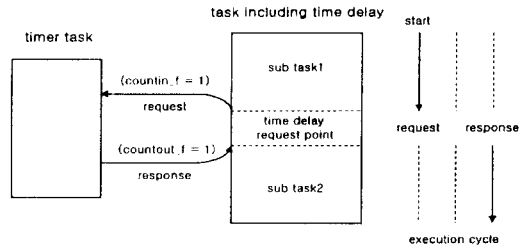

Fig. 3 Execution cycle of a task including time delay

완료되면 다시 각 태스크에 알려주도록 하는 방법을 도입한다.

우선 시간지연 계산용 특권 타이머 태스크내에 unsigned int 형의 count변수를 선언한 후, 50ms 간격으로 이 변수를 1씩 증가시킨다. 시간지연 요소를 포함한 태스크들은 각각 다음과 같은 4개의 변수를 가지고 있다.

countin_f : 시간지연 계산 요청 플래그
countout_f : 시간지연 계산 완료 플래그
DEMANDVALUE : 시간지연량
timerstart : 시간지연 계산 시작시의 count 값

태스크가 시간지연 계산을 요청하게 되면 Fig. 2처럼 countin_f=1로 초기화하고 timerstart에는 현재의 count 값을 저장한다. 특히 타이머태스크는 다른 태스크들로부터 시간지연 계산에 대한 요청이 들어오면 ((count - timerstart)>DEMANDVALUE) 조건을 검사하여, 이 조건이 성립하면 countin_f=0, countout_f =1로 한다.

시간지연을 포함한 태스크는 시간지연 계산을 요청하는 위치를 기준으로 하여 서브 태스크로 분할되므로 각 서브태스크의 실행은 Fig. 3과 같이 이루어진다. 즉, sub task1을 실행한 후 시간지연 계산을 요청한 다음에는 sub task2는 실행하지 않으며, 이후 실행권이 주어진 경우에도 countin_f가 0이고, countout_f가 1이 될 때까지는 실행하지 않는다. 특권 타이머태스크가 countout_f=1로 변경시키게 되면 그때 sub task2를 실행하게 된다.

3. 시스템의 태스크 구성

주기관의 상태로는 stop과 run이 있으며, run상태는 다시 ahead와 astern상태로 세분된다. 주기관 원격제어시스템을 구축하는 것은 주기관의 각 상태의 보존동작과 상태간의 전환동작을 구현하는 것을 말한다. 이를 위해 본 논문에서는 주기관의 동작을 다음 6가지 태스크로 분류하였다.

① stop상태의 보존동작(stop 태스크)
② run상태의 보존동작(running 태스크)
③ stop상태에서 run상태로의 전환동작(starting 태스크)
④ run상태에서 stop상태로의 전환동작(stopping 태스크)

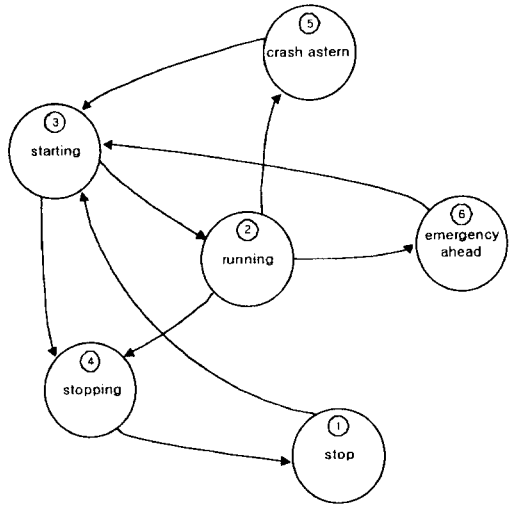

Fig. 4 State transition of tasks

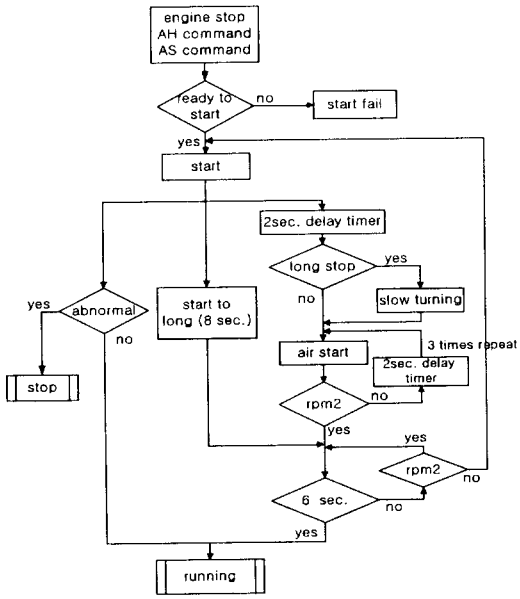

Fig. 5 Starting task

⑤ ahead run상태에서 astern run상태로의 전환 동작(crash astern 태스크)

⑥ astern run상태에서 ahead run상태로의 전환 동작(emergency ahead 태스크)

Fig. 4는 태스크들간의 상태 천이도를 보여주고

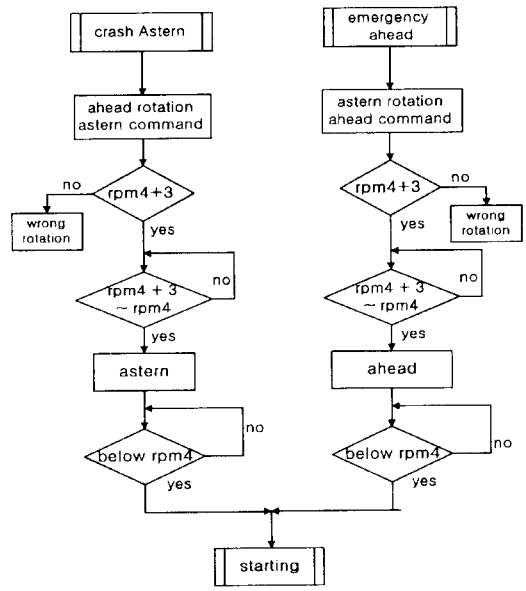

Fig. 6 Crash astern and emergency ahead tasks

있으며, 여기에서 ①번은 안전조건을 감시하고 governor output 값을 항상 0으로 유지하여 주면 된다. ④번은 stop command, engine tripped, wrong rotation의 조건에 의해 기동되고 governor output값을 0으로 유지해 주면 된다.

또한 ③은 Fig. 5, ⑤와 ⑥은 Fig. 6과 같은 흐름도로 나타낼 수 있지만, 이들 동작은 시간지연 요소 등을 포함하고 있어서 많은 실행시간을 요구하므로 다시 서브태스크로 나누어 실행시킨다. 여기에서 rpm2, rpm4는 엔진 기종에 따라 서로 다르게 셋팅한 일정 회전수를 의미한다. ②는 주기관의 속도제어를 의미하며 이것에 관해서는 다음 장에서 논하기로 한다.

4. 주기관의 속도제어

running 태스크에서는 command RPM에 대응하는 governor output값을 출력해 주는 동작을 수행하면 된다. 그러나 command RPM은 순간적으로 변화되기 때문에, 여기에 맞추어 governor output값을 즉시 변화시키면 엔진에 무리한 속도변화

를 초래하여 엔진 고장의 원인이 된다. 따라서 command RPM에 대해 일정시간 간격으로 변화시켜야만 한다. 그러나 대부분의 주기관의 경우 임계영역을 포함하고 있기 때문에 이 영역을 빨리 회피할 수 있도록 governor output값을 출력해 주어야 한다.

이를 위해 본 논문에서는 Fig. 7과 같이 임계영역의 중간지점에 도달했을 때에 변화시키는 방법을 이용하기로 한다. 즉 RPM이 임계 영역내에 들어간 경우 cn1 RPM에 해당하는 governor output값을 출력해 주다가 RPM이 임계 영역의 중간지점을 넘어서는 시점에서 cn2 RPM에 해당하는 governor output값으로 출력해 주도록 한다.

또한 주기관의 속도제어에는 임계영역을 피하는 기능 이외에도 다음과 같은 여러 기능들이 존재한다. 본 논문에서는 이 기능들에 다음과 같이 우선순위를 부여하여 중복되는 경우에 우선순위가 높은 쪽의 기능을 우선적으로 따르도록 하고 있다. 단, 번호가 큰 쪽이 우선순위가 높다.

① load UP/DOWN시의 governor output 결정기능
② Normal시의 governor output 결정기능
③ manual RPM limiter에 대응하는 governor output 결정기능
④ rough sea limiter에 대응하는 governor output 결정기능
⑤ critical RPM에 대응하는 governor output 결정기능
⑥ slow down시의 governor output 결정기능
⑦ minimum RPM에 대응하는 governor output 결정기능

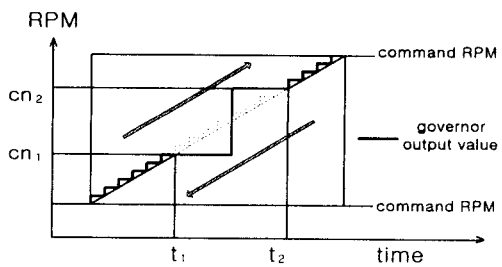

Fig. 7 Proposed speed control method

5. 플리커링 동작

각 데이터들의 상태를 모니터링하기 위해서는 이들 값을 LED등을 이용하여 표시해 주어야 한다. LED에 표시하는 방법으로는 ON 또는 OFF가 주로 이용되고 있지만, 경우에 따라서는 경보를 나타내기 위하여 LED를 일정간격으로 ON-OFF하는 플리커링 방법이 필요하게 된다. 여기에서는 이러한 방법의 구현에 대하여 논하기로 한다.

우선 Fig. 8과 같이 플리커링을 필요로 하는 데이터들을 별도로 관리하기 위해서 논리적으로 구성되어 있는 데이터들로부터 steady LED용 데이터와 flicker LED용 데이터로 분류한다. 그리고 flicker LED용 데이터들은 각각 Fig. 9와 같은 플래그 변수를 이용하여 flicker_request 비트를 1로 한 후, 다음과 같은 순서에 의해 LED에 점등해 준다.

(1) 플래그 변수의 flicker request 비트를 검사하여, 이 값이 1이면, 플리커링 주기가 정상주기인 경우 flicker_normal_start 비트를 1로 셋트하고 긴 주기인 경우에는 flicker_long_start 비트를 1로 셋트한다. 또한 반전 시기를 결정

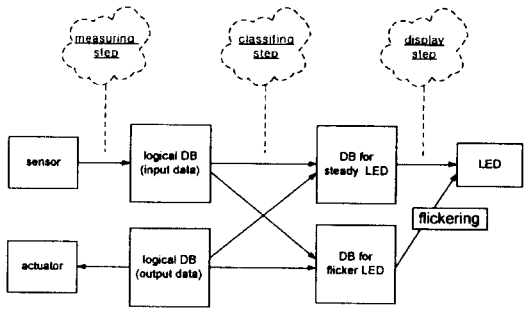

Fig. 8 Management of input-output data

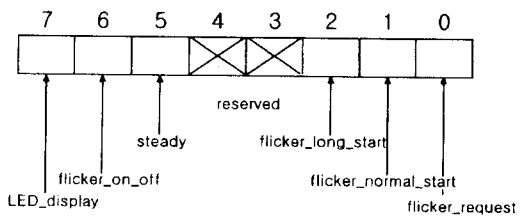

Fig. 9 Flag variable for flicker LED data

하기 위해 타이머를 작동시킨다.

(2) 타이머에 의해 반전시기가 결정되면 flicker_on_off 비트를 1 또는 0으로 한다.

(3) 외부로부터의 플리커링 정지신호를 감지하면 steady 비트를 1로 한다.

(4) flicker request 비트가 1이고 steady 비트가 0이면 flicker_on_off 비트값을 LED_display 비트에 복사하고, steady 비트가 1인 경우에는 LED_display 비트를 1로 한다.

(5) LED_display 비트값이 1이면 LED를 ON으로, 0이면 OFF 상태로 한다.

6. 제어위치의 절환동작

주기관 원격제어 시스템은 브릿지(BCS)와 기관실원격제어실(RCS)에 각각 한 대씩 놓여 있으며 이들이 RS-422 시리얼 라인을 통해 서로 통신을 수행하고 있어서 어느 곳에서라도 원격제어할 수 있다. 이때 나머지 하나는 단지 모니터링만을 수행한다. 실제 제어위치를 변경시키기 위해서는 RCS측에 있는 change_over switch를 이용하지만, 그 이전에 어디에서 원격제어를 수행할 것인가를 결정해 둘 필요가 있다. 두 시스템간에 제어위치를 절환하기 위해서는 두 개의 버튼을 이용하여 서로 약속된 신호에 따라 절환되도록 하여야 하며, 본 논문에서는 다음과 같은 규칙을 이용하고 있다.

① BCS측에서 bridge control position 버튼을 눌러 BCS에서의 원격제어를 요청하면 BCS와 RCS 양쪽의 해당 LED가 모두 정상주기로 플리커링하도록 한다. 이때 RCS측에서 bridge control position 버튼을 눌러 요청에 대해 응답을 하면 양쪽 LED 모두가 긴 주기로 플리커링하도록 한다. 이후 RCS측에 있는 change_over switch를 변경시켜 bridge control position으로 바꾸면 LED를 계속점등 상태로 한다.

② RCS측에서 bridge control position 버튼을 눌러 BCS에서의 원격제어를 요청하면 BCS와 RCS 양쪽의 해당 LED가 모두 정상 주기로 플리커링하도록 한다. 이때 BCS측에서 bridge control position 버튼을 눌러 요청에 대해 응답을 하면 양쪽 LED 모두 긴 주기로 플리커링하도록 한다. 이후 RCS측에 있는 change_ over switch를 변경시켜 bridge control position으로 바꾸면 LED를 계속점등 상태로 한다.

이상은 RCS에서 BCS로 제어위치를 절환하고자 하는 경우이며, 이 반대의 경우에는 값만 반대일 뿐 순서는 똑같다.

본 논문에서는 제어위치의 절환을 위해 Fig. 10과 같은 형식의 플래그 변수를 이용하고 있다. 또한, RCS에서 BCS로 송신할 때에는 BCS와 RCS용의 flicker_normal_start 및 flicker_long_start,

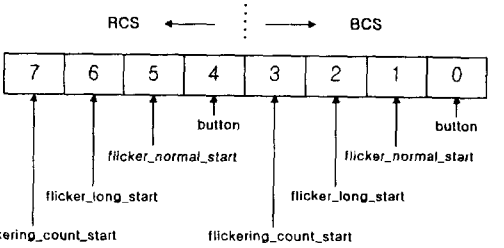

Fig. 10 Flag variable for changing the control position

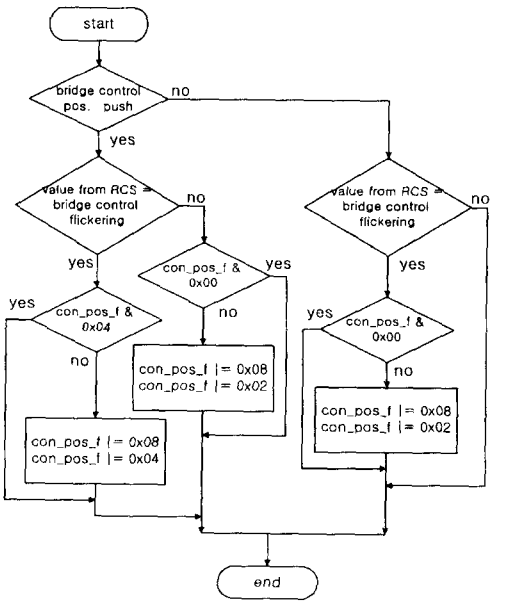

Fig. 11 Algorithm for changing the control position to the bridge control position

change_over switch의 값을 넘겨주고, BCS에서 RCS로 송신할 때에는 flicker_ normal_start 및 flicker_long_start의 값을 넘겨주도록 하고 있다. 이 플래그 변수값과 통신을 통해 수신한 상대측의 값 및 버튼의 상태값을 이용하여 상기 규칙에 대한 BCS측의 알고리즘을 나타내면 Fig. 11과 같다. 그림에서 con_pos_f는 플래그 변수를 의미하며, bridge control flickering은 BCS측의 flicker_ normal_start 또는 flicker_ long_start가 1인 경우를 의미한다.

7. 결 론

본 논문에서는 시간지연을 수반하는 실시간제어 방법과 이를 위한 태스크 구현방법에 대하여 논하였다. 또한 주기관의 운전동작 및 속도제어 방법, crash astern, emergency ahead, 통신을 이용한 제어위치의 절환방법 및 플리커링 방법의 구현에 대하여도 논하였다.

이들 방법을 기능별 태스크 형태로 구현함으로써 주기관 원격제어 시스템의 수정과 프로그래밍이 간결하게 되었다. 특히 타이머를 하나만 이용하기 위해 타이머 태스크를 별도로 두어 시간지연 계산을 요청하는 태스크들과 플래그 변수만을 이용하여 처리하는 기법을 도입함으로써 하드웨어를 간결하게 할 수 있었다.

현재 타이머를 계산하기 위한 변수는 unsigned int 형으로 선언하였기 때문에 0~65535까지의 값을 가질 수 있으며 최대 시간은 3276초(약 54분)이다. 일반적으로 시동시 터닝 기어를 동작시킬 것인지를 계산하기 위한 시간이 30분 정도로 가장 긴 시간이므로 현재 이용하고 있는 변수의 크기로도 충분하지만, 만약에 이보다 더 큰 시간지연 요소가 존재하는 경우에는 변수를 long형으로 선언하든지, 아니면 2개의 변수를 사용해야 할 것이다.

참고문헌

[1] 細川成通, 主機關の自動化の現代と將來動向, 日韓學術交流講演會, pp.2 11, 1993.

[2] 中川司, 船用コンピュータシステムと國際動向, 日本舶用機關學會誌, Vol.32, No.2, pp.79-85, 1997.

[3] M-800B Bridge Maneuvering System Manual, NABCO.

[4] Auto-chief 4 Main Engine Remote Control System Manual, NORCON.

[5] 최진태·김종식, 실시간운영체제와 로봇제어기에의 적용, 대한기계학회지, 제36권, 제3호, pp.231-242, 1996.

[6] 佐竹秀己, リアルタイムシステムのOSとデバイスドライバ, インタフェース, 1996.

[7] 實踐リアルタイムプログラミング技法, 大原, 澤田, Peter Petrov, オーム社, 1991.

[8] マルチタスク・プログラミング入門, トラ技コンピュータ, CQ出版社, 1992.

[9] リアルタイム&マルチタスク・プログラミング, BootStrap No.6, 別冊インタフェース, 1992.

[10] 실시간멀티태스크 처리기법을 이용한 선박엔진 안전시스템 개발에 관한 연구, 황동희, 한국해양대학교 석사학위논문, 1995.

저 자 소 개

정경열(鄭暻烈)

1960년 1월생,1982년 한국해양대학교 기관학과 졸업,1984년 동대학교 대학원 졸업(석사),1996년 동대학원 박사과정 수료,1982년 1986년 기관사 승선근무,1987년 현재 한국기계연구원 환경설비연구부 선임연구원,당학회 종신회원.

김종화(金鍾和)

1958년 11월생,1981년 부산대학교 공과대학 기계공학과 졸업,1985년 동대학교 대학원 졸업(석사),1989년 동대학원 졸업(박사, 제어공학),1990년 현재 한국해양대학교 자동화·정보공학부 부교수,1996년 1997년 영국 WALES 대학교 연구교수,당학회 종신회원.

류길수(柳吉洙)

1953년 5월생,1976년 한국해양대학교 기관학과 졸업,1979년 동대학교 대학원 졸업(석사),1978년 1982년 기관사 승선근무,1982년 현재 한국해양대학교 자동화·정보공학부 교수,1984년 1986년 일본 동경공업대학 대학원 졸업(석사),1986년 1989년 동대학원 졸업(박사),당학회 종신회원.