연속 공정 자동화를 위한 라인 제어기에서의 실시간 작업 스케쥴링에 관한 연구

o 이 준 수*, 조 영 조*, 임 미 섭*, 박 정 민*, 최 익*, 임 준 홍**, 김 광 배*
* KIST 제어시스템 연구실, **한양대학교 전자공학과

Design of Real Time Task Scheduling for Line Controoler of Continuous Manufacturing Process Automation

o Joon Soo Lee*, Young Jo Cho*, Mee Seub Lim*, Jung Min Park*, Ick Choy*, Jun Hong Lim**, Kwang Bae Kim*
* KIST Control Systems Lab., ** Hanyang Univ. Dept. of Electronics Eng.

ABSTRACT

This paper presents an approach to the design of real time task scheduling for a line controller of continuous manufacturing process automation. The line controller has multiprocessor-based architecture with shared memory and is operated by firmware. This firmware contains menu-driven software supporting real-time database management and fuction-block control language. The multitasking line control processor performs the following three functions: 1) interprets the function block control language by virtue of shared memory in the database; 2) invokes an interupt service routine as required by external hardware; 3) detects errors and notifies the user. We propose real time task scheduling method.

1. 서 론

1960년대 디지탈 컴퓨터가 산업제어에 사용된 이후 1974년 마이크로프로세서의 출현으로 거의 모든 산업공정에서 생산성 향상과 생산단가 감소를 위해 제어용 컴퓨터가 사용되어 왔다.[1]

많은 생산공정 중에서 제철공장의 냉간, 열간 압연공정이나 강관 공정, 금속공장의 도금공정등은 특히 연속공정시스템으로 여러개의 작업공정들이 서로 연관되어 작업되야 하며 불규칙적으로 발생하는 공정도 수반한다. 이러한 연속공정시스템을 위한 제어 시스템은 제어작업이 서로 연관되어 제어되어야 하며 불규칙적인 제어 요구에 대해서도 작업을 수행하여야 한다.[2]

이러한 제어 시스템을 구현하기 위해서 기존의 제어 시스템들은 특별한 하드웨어나 소프트웨어를 사용하였는데 이는 확장성과 유연성이 나쁘다는 단점이 있다.

실시간 multitasking 오퍼레이팅 시스템은 개방형 구조를 갖는 오퍼레이팅 시스템과 software 개발환경으로 구성되어 있어 새로운 제어용 컴퓨터에도 쉽게 이식할 수 있고 개발하기가 편리하다. 또한 multitasking과 intertask communication을 근본으로 응용 프로그램이 독립된 task로 이루어져서 각각의 실행 코드와 system resource를 쓸 수 있도록 해주고 task들이 서로 동기를 맞추고 통신할 수 있도록 해준다.[3][4]

본 논문에서는 금속공정과 같이 기계적공정 뿐만 아니라 화학공정도 포함한 공정에 대해서도 제어가 가능한 연속공정제어를 위한 라인제어기에 대해 실시간 작업 스케쥴링 방법을 제시한다.

2. 연속공정 자동화를 위한 라인제어기의 구성

제철공장의 냉간, 열간 압연공정이나 강관공정, 금속공장의 도금공정 등 연속공정의 자동화 시스템은 생산관리 및 최적화를 위한 관리제어컴퓨터 (SCC: Supervisory Control Computer)를 상위계층으로 하여, 다수 전동기들의 속도 및 장력제어등 복합적인 기계 자동화를 담당하는 라인제어 유닛 (LCU: Line Control Unit)과 세척, 도금, 탈수, 회석등 계장제어 기능을 담당하는 공정제어유닛(PCU: Process Control Unit)이 실시간 네트워크를 통해 하위계층으로 연결되는 계층 분산구조를 갖는다.[5] 본 논문에서 대상으로하는 연속공정 자동화를 위한 라인제어기 (Line Controller)는 복합 기계 자동화 기능을 수행하는 라인제어 유닛의 해심부로, 수백점의 아나로그/디지탈/펄스 신호를 직법 입출력하고 각종 시퀀스 처리를 위한 PLC (Programmable Logic Controller)와 전동기의 속도/전류 루프제어를 위한 MDC(Motor Drives and Controllers)들을 필드버스를 통해 연결하여, 시퀀스 및 루프 복합 제어 알고리즘을 실시간 온라인으로 처리한다.

라인 제어기는 그림 1과 같이 VMEbus와 공유 메모리를 통해 시스템 제어 및 조작자 인터페이스 (SC&OI: System Control and Operator Interface), 라인제어 (LC: Line Control), 네트워크 인터페이스(NI: Network Interface) 등 3종류의 32비트 프로세서 모듈들이 연결되는 다중 프로세서 구조를 갖는다.

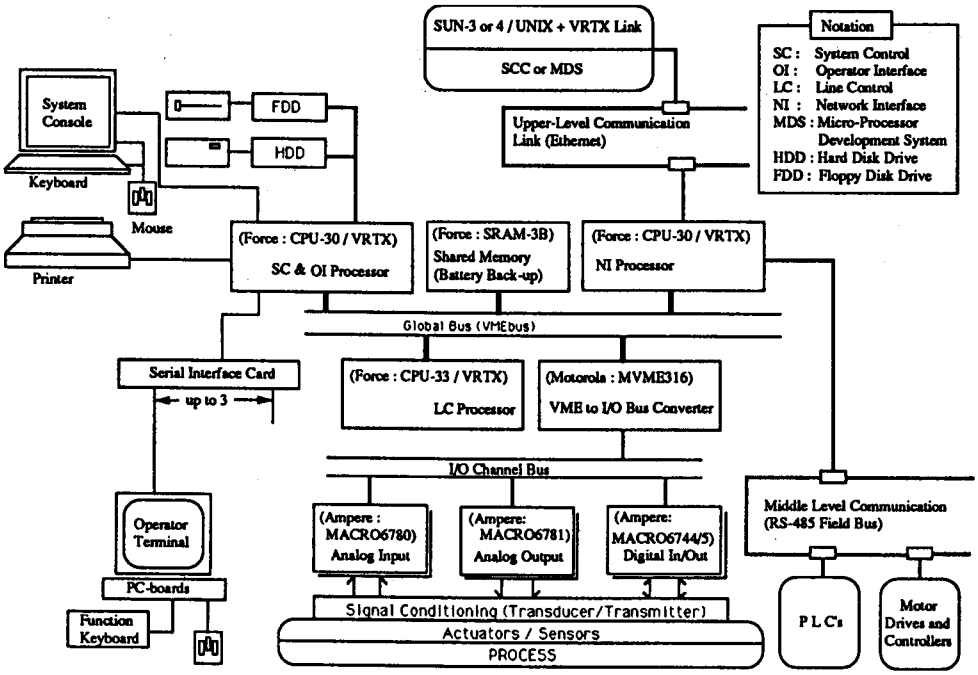

그림 1. 라인제어기의 하드웨어 구조

SC&OI 프로세서는 전체 database 관리와 Function Block Diagram 언어를 컴파일하는 기능을 한다. 사용자 인터페이스를 위해 2종류의 디스크없는 IBM-PC호환 컴퓨터가 Serial 광통신으로 연결된다. 사용자가 쉽게 조작할 수 있도록하기 위해 MS-Window를 사용하였다. LC 프로세서는 실시간 multitasking으로 컴파일된 Function Block Language를 실행한다. NI 프로세서는 Ethernet을 통한 상위 레벨 컴퓨터와의 통신과 필드버스를 통한 PLC, MDC와의 통신을 담당한다.

라인제어기는 복잡한 제어기능을 수행하기 위해 많은 데이타의 처리와 실시간 multitasking을 요구하느데 이를 위해 VRTX 실시간 오퍼레이팅 시스템을 사용한다.

3. 실시간 제어 알고리즘 구성 및 실행

라인제어기에서는 제어알고리즘의 수행에 사용되는 입출력이 입출력 데이타 영역을 통해서 되기 때문에 입출력의 정의가 중요하다. 입출력은 DIO (Direct 입출력 장치를 통한 데이타), PLC (PLC 를 통한 데이타), MDC (Motor Drive를 통한 데이타), SET (Console 이나 터미날을 통한 set-point data), ETH (Ethernet 을 통한 SCC와의 통신데이타) 로 나누어진다.

이들 데이타들은 그림 2와 같이 SC&OI, LC, NI 프로세서의 task들에 의해 처리된다. 모든 입출력 데이타는 SC&OI 프로세서의 I/O description procedure 에서 정의된다. 시스템 콘솔의 사용자가 모든 입출력 데이타의 정보를 메뉴방식으로 입력하면 I/O description files 에 저장한다. I/O data mapping procedure는 모든 입출력 데이타에 대한 정의를 모아서 shared memory에 넣는다. memory mapping이 끝나면 graphic monitoring procedure는 미리 정의한 라벨이름에 따라 입출력 데이타를 가져다가 operator terminal에 정의된 형태로 표시한다. 그외 데이타를 입출력하기 위해 set-point handling procedure, Direct input/output procedure,

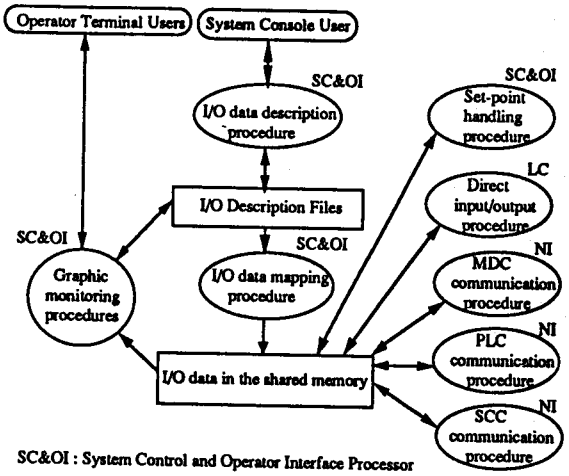

그림 2. 입출력 데이타 관리의 개념도

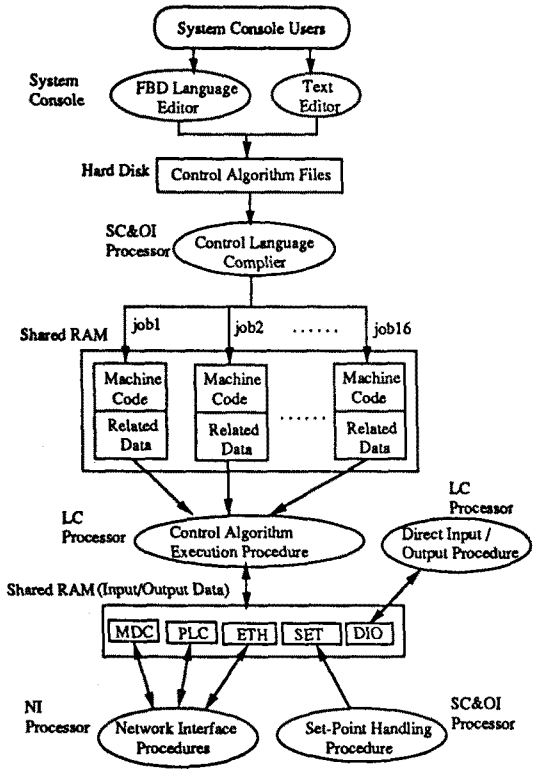

그림 3. 프로그램의 구성과 실행의 개념도

MDC communication procedure, PLC communication procedure, SCC communication procedure가 있다.

연속공정의 실시간 라인제어를 위한 프로그램을 작성하기 위해서 Function Block Diagram을 사용한 그래픽 제어 언어가 사용된다. 라인제어 알고리즘을 위해 60여가지의 function block 이 정의가 되었고 프로그램 수행시 실행될 수 있도록 firmware 로 Line Control Processor에 들어 있다.

그림 3 은 function block 언어를 구현하기 위한 소프트웨어 레이아웃과 실행 과정을 보여준다. system console은 프로그램 작성을 위해 function block diagram (FBD) editor 와 text editor를 제공한다. 사용자는 메뉴방식의 editor를 이용해서 그림 4 와같은 프로그램을 작성할 수 있다. 이 프로그램은 그림 5 와 같은 형태로 control algorithem file에 저장된다. SC&OI 프로세서의 하드디스크에 저장된 프로그램은 control language compiler에 의해 machine code와 related data의 형태로 compile된다. 시스템 콘솔에 있는 작업 구성 메뉴에서 제어 언어 컴파일러의 작동, 작업 번호 지정, 정해진 영역으로의 machine code와 data저장등을 할 수 있다. 라인 제어 프로세서에서는 이들 machine code와 data를 가지고 프로그램을 수행한다.

4. 실시간 작업 스케쥴링

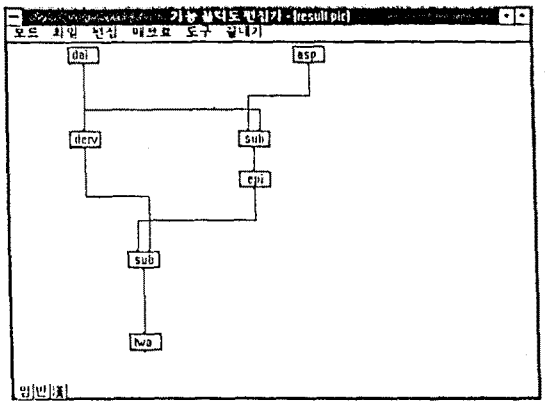

그림 4. Function Block Diagram 예제의 Editor 화면

```
%%
F1 = dai(d0, c0, "PV_Msp")
F2 = derv(F1, 1)
F3 = asp(d0, c0, 2, "SV_Msp")
F4 = sub(F3, F1, 1)
F5 = epi(F4, 3, 3)
F6 = sub(F5, F2, 1)
twa(F6, d0, c0, "MV_Msp")
END
%%
```

그림 5. 제어 알고리즘 화일의 예

라인 제어 프로세서에는 사용자가 정의한 샘플링 타임에 의해 실행되는 20개의 제어작업 task와 외부 interrupt에 의한 6개의 제어작업 task, 그리고 error처리를 위한 1개의 task가 있다. 그림 6 에는 이들 task들을 실시간 multitasking으로 스케쥴링하기 위한 개념도가 있다.

각 task들은 creation되면 자신의 semaphore에 pending한다. 이때 pending 하고 있는 task들은 suspend 상태가 되어 CPU에 load를 주지않는다.

조작자가 설정한 샘플링 타임에 의한 제어 task의 실행은 timer interrupt service routine에 각 task에 대해 그림 6에 있는 time table을 갖고 있다. 샘플링 타임의 측정은 처음 설정한 값과 매번 interrupt 에 의해 증가되는 해당 task의 counter값이 일치 할 때 제어 task에 해당 semaphore을 post 한후 counter는 영으로 한다. task의 우선순위는 샘플링 타임에 의해 결정되는데 짧은 시간의 task가 높은 우선 순위를 갖는데 느린 task에 대해서도 거의 비슷한 주기적인 제어가 가능하다.

외부 하드웨어 카운터와 같은 외부 인터럽트에 의한 제어 task의 실행은 6 레벨의 인터럽트가 발생하면 interrupt service routine이 해당 task에 semaphore를 post한다. 인터럽트 처리용 제어 task는 타이머에 의한 task보다 높은 우선순위를 갖는다.

이상의 제어 task들은 동일한 프로그램 코드를

사용한다. 다만 creation된 task가 어느 semaphore에 pending하는냐에 따라 타이머나 외부 인터럽트에 의해 실행이 된다.

에라 처리 task는 direct IO board의 탈장이나 shard memory의 기능정지, 제어 프로그램의 오류 발생 등에 대해 Bus trap error를 받아 처리 하거나 주기적으로 각 error flag를 검사해서 error의 발생을 조작자에게 알려준다.

5. 결론

본 논문에서는 연속공정 자동화 시스템에서 실시간 온라인으로 제어가 가능한 라인제어기 개발에 대해 간단히 소개 하고 실시간 작업 스케쥴링에 대해 한 방법을 제시하였다. 제시된 방법은 복잡한 제어 공정에서 여러개의 제어작업이 실시간으로 수행될때 각각의 작업을 효율적으로 스케쥴링해줄 수 있도록 하였다. 실제 작업을하는 task이외에는 CPU시간을 소모하지 않도록하였으며 제어 task이외에는 creation하지 않아 메모리의 낭비도 줄이도록 하였다.

하지만 각 task상호간의 통신과 error 발생 task에 대한 스케쥴링이 남은 과제이다.

참고문헌

[1] Stuart Bennett, Real-time Computer Control , Prentice Hall, 1988

[2] T.J.Williams, "Recent Developments in the application of plant-wide computer control", Computers in Industry, Vol. 8, pp233-254, 1987

[3] James F. Ready, "VRTX : A Real-Time Operating System for Embedded Microprocessor Applications", IEEE MICRO, August, 1986

[4] VxWorks Programmer's Guid, Wind River Systems, 1990

[5] 김광배외, "연속공정 자동화용 다기능 제어시스템의 개발", '91 로보틱스및자동화연구회워크샵, pp 107-113, Mar.1991

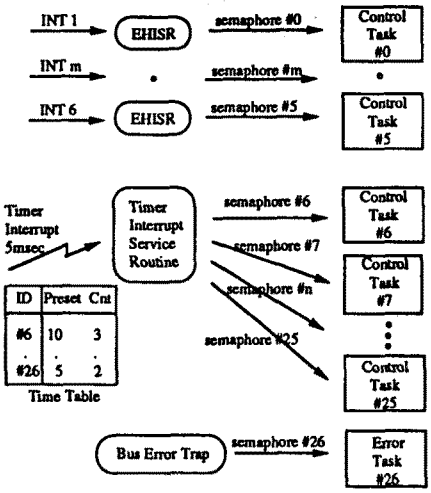

그림 6. 실시간 작업 스케쥴링의 개념도