2005년도 대한전기학회 하계학술대회 논문집 2005. 7. 18 - 20

공장 자동화에서의 온라인 다운로드 기능 구현

최선아, 박기웅, 이창원, 류명선
포스콘 기술연구소 설계 기술팀

The implementation of Online downloading function for Factory Automation

Choi SunAh‡, Park KiWong, Lee ChangWon, Ryou MyoungSeon
POSCON R&D Center Design Technology Team

Abstract - 본 논문에서는 공장 자동화에서 연속 공정 제어를 함에 있어 Multi-tasking을 보다 효율적으로 수행 하기 위해 온라인 다운 로드 기능을 제안하였다.
이는 연속 공정 제어를 위해 LD, ST, IL, FBD 등과 같은 제어 언어로 초기 프로그램 개발 시 잦은 프로그램 변경 및 수정을 요하는 곳에서 매우 유용하게 활용 될 수 있으며 실제 개발 시간의 단축도 가져온다. 또한 철강라인의 가열로나 자동차 조립라인등과 같이 생산라인을 도중에 멈출 수 없는 경우 온라인 다운 로드를 이용하면 생산 라인에 지장없이 프로그램을 수정후 변경된 프로그램을 바로 실행하여 적용할 수 있다.
이처럼 한 번 프로그램을 실행하면 도중에 멈출 수 없는 생산라인이나, 빠른 처리 속도를 요하는 곳에서 온라인 다운 로드기능을 이용하여 여러 개의 태스크를 분할 관리 함으로써 전체적으로 시스템의 성능을 향상 시키는 결과를 보여주고 있다.

1. 서 론

공장 자동화(Factory Automation)란, 컴퓨터와 각종 계측 장비를 이용하여 공장의 전 생산 공정을 자동화한 시스템을 일컫는다. 일반적으로, 공장 자동화의 주목적은 공장에서 제품 생산 시 최소한의 인력 투입으로 보다 안전하고, 정확하게 생산을 증대 시키는데 있다. 이때 임베디드 시스템을 적용하여 RTOS(Real Time Operating System)를 적당한 하드웨어에 탑재함으로써 OS의 특징이 적용된 시스템을 구현 할 수 있는 것이다[1][3].
본 논문에서는 공장 자동화 중에서 연속 공정과 관련하여 실시간 운영체제의 특징을 설명하면서 제안하고자 하는 온라인 다운로드의 기능의 필요성과 쓰임새, 구체적인 알고리즘의 흐름도를 말하고자 한다.

2. 본 론

2.1 실시간 운영체제(RTOS)에 대한 설명과 특징

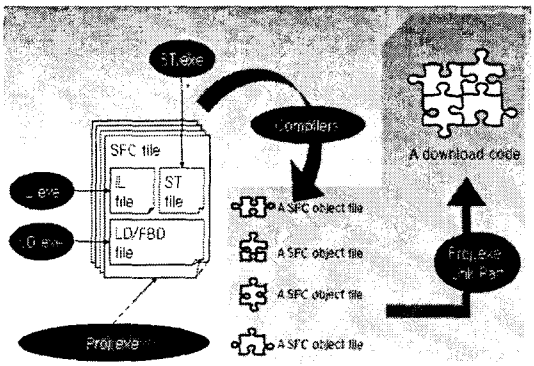
실시간 운영체제라고 하면 대개 많은 데이터량을 무조건 빠른 속도로 처리만 하면 RTOS라는 선입견을 가지고 있다. 물론 시간적인 정확성을 확보하는 것이 RTOS의 특징 중의 하나이나, 단순히 빠른 처리 속도만이 전부가 아니다. 정해진 시간 내에 일을 처리하기만 하면 된다. 예를 들어, 제어기내 태스크 한 개의 처리 시간을 하루를 줬다면, 이를 하루 안에 끝내면 RTOS의 조건을 만족한 셈이다. 그리고, RTOS는 주로 특정 하드웨어를 기반으로 하는 경우가 많기 때문에 임베디드 시스템과도 밀접한 관련을 지니고 있다.
현재까지 상용화 되고 있는 대표적인 실시간 운영체제를 꼽는다면, vxWorks, pSOS, VRTX, OSE, Nucleus, MC/OSII 등을 들 수 있다. 이들 실시간 운영체제는 공통적으로 특정 태스크를 중단시키고 다른 태스크를 수행할 수 있도록 하는 선점형 멀티태스킹을 지원한다.
본 논문에서 사용한 vxWorks의 특징을 살펴보면, vxWorks의 커널은 선점형(preemptive) 멀티 태스킹으로 두개 이상의 태스크가 같은 우선 순위를 가진다면 라운드 로빈 방식의 스케줄링을 이용한다. vxWorks의 구조는 멀티쓰레드(multi-thread)모델로 운영체제의 커널과 응용 프로그램이 합쳐져서 서로의 구분이 없는 하나의 큰 프로그램이 되어 작동하는 구조로 커널과 응용 프로그램이 공통의 작업 영역인 메모리를 자유롭게 접근 할 수 있다. 운영체제의 크기가 작고, 비교적 작은 크기의 시스템에서 구현이 쉽고 빠르다는 장점이 있지만, 커널과 응용프로그램이 하나의 프로그램 모듈로 동작하기 때문에 사소한 버그가 시스템 전체를 파괴할 수 있다는 단점이 있다[3].

2.2 온라인 다운로드 기능 제안

2.2.1 온라인 다운로드의 필요성

산업용 컴퓨터가 시퀀스 제어용 PLC(Programmable Logic Controller)와 루프제어용 DDC(Direct Digital Controller)등으로 전용화 됨에 따라 응용 목적에 따른 공정제어 전용의 제어 언어들이 생성되었다. 현재 공정제어용 디지털 컴퓨터에 사용되는 제어 언어 중 대표적으로 어셈블리어와 비슷한 IL(Instruction List), 고급언어로 구성된 ST(Structured Text), 그래픽 언어인 LD(Ladder diagram), FBD(Function Block Diagram)등을 들 수 있다[2][4][5][6].
실제 위와 같은 제어언어를 이용하여 프로그램을 하는 사용자들은 연속 공정에서 라인 하나를 제어하기 위해 수천 라인에 달하는 코드를 만들고, 수만에 달하는 태그를 사용한다. 완벽한 코드를 완성하기 까지 잦은 프로그램 변경을 거쳐 테스트를 해 봐야 하는데, 이제 까지는 여러 개의 태스크 중 단 하나의 태스트를 수정하더라도 전체 태스크를 멈춰서 삭제 후에 다시 전체 태스크를 다운 받아 실행해야 하는 불합리한 구조로 되어 있었다. 이는 다름 아닌, 태스크를 실행하는 중간코드 파일이 단 하나의 파일로 만들어져 있기 때문이다.
아래에 보이는 [그림 1]처럼 컴파일러를 거쳐 생성된 각 태스크의 중간코드는 하나의 단일 파일로 만들어 진다. 때문에 아주 사소한 태그나 변수명 하나를 수정하더

라도 전체 태스크를 멈추고, 삭제하는 등의 비효율적인 수행을 하게 되는 것이다. 하지만, 철강산업 분야에서 가열로나 자동차 조립 라인, 제지 공정과 같은 곳에서는 수행 중인 태스크를 멈추거나 삭제할 수 없는 곳이다.



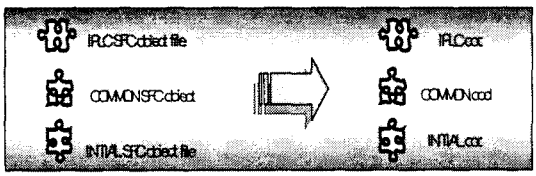
[그림1] 중간 코드 생성 과정

이처럼 특정한 곳에서 태스크를 수정하고 싶어도 주위 환경이나 여건이 허락하지 않을 때 본 논문에서 제안하는 온라인 다운로드를 사용함으로써, 태스크의 수행 중단 없이 수정된 태스크를 안전하게 교체 수행할 수 있다.

2.2.2 온라인 다운로드를 위한 중간 코드 생성 분리

본 논문에서 사용하는 제어기는 IEC61131-3에 기반하여 제어언어를 지원하고 있다. 그림에서 나오는 SFC는 하나의 태스크로 동일시 한다.

온라인 다운로드 기능을 수행하기 위해서는 먼저 시스템 구조 및 중간코드의 포맷이 이에 맞게 수정해 주어야 한다. [그림2]와 같이 각각의 제어 언어로 만들어진 SFC 파일을 컴파일러를 거쳐 object file로 생성 시키는데, 이때 SFC 파일별로 중간코드를 관리하도록 한다.



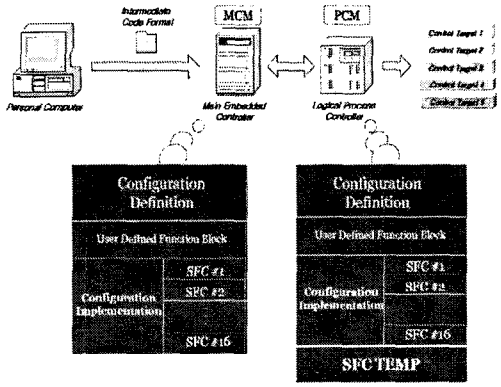
[그림 2] 중간 코드 분리

이런 구조를 가지게 되면 만약 INTIAL SFC 파일을 수정한다고 가정 했을 때, 부분 컴파일과 빌드를 통해 INTIAL SFC 파일만을 온라인 다운로드 하여 제어기 내에서 수행되고 있는 다른 태스크에 전혀 영향을 미치지 않고 교체를 할 수 있게 되는 것이다.

2.2.3 온라인 다운로드 흐름도

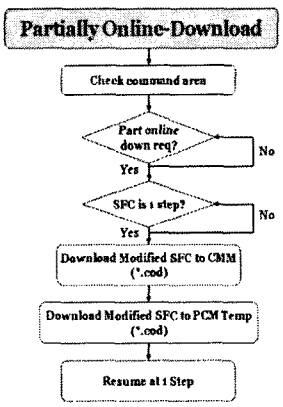
온라인 다운로드의 흐름을 전체적인 시스템 구조와 함께 보다 구체적으로 살펴보기로 한다.

먼저, [그림 3]를 참고로 전체적인 시스템 구조를 보면 사용자가 제어언어로 프로그램을 만들고, 여기서 생성한 태스크를 수행할 수 있도록 지원하는 PC 기반의 그래픽 디버깅 툴이 있다. 이것은 일반 PC에서 작업이 가능한 것으로 프로그램 다운로드 및 태스크를 실행, 삭제등과 같은 모든 명령은 여기서 내려진다. 그 다음 MCM에서는 상위에서 받은 명령을 전달해주고, 중간 코드 파일을 받아 공유 메모리에 저장을 하게 된다. PCM 은 공유메모리를 통해 상위에서 받은 명령을 구분, 처리하게 된다. 공유메모리는 각종제어기와 관련된 정보를 저장하는 곳으로, 상위의 command 뿐만 아니라, 컴파일된 중간코드도 함께 저장되는 곳으로 구조는 PCM 구조에서 SFC Temp 영역을 제외한 나머지가 같은 구조이다.



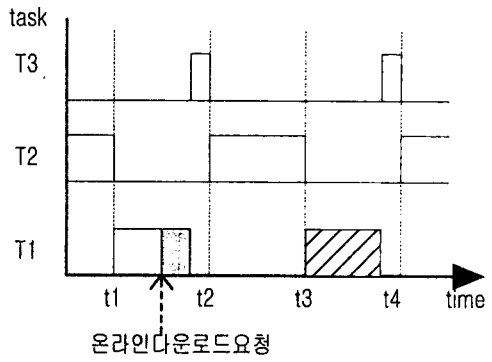
[그림 3] 전체적인 시스템 구조

온라인 다운로드의 명령이 넘어올 경우 [그림 3]에서 보이는 PCM에서는 공유메모리의 command 영역을 검사하고 Online Download 요청이 있는지 확인한다. 만약 있다면, SFC 의 STEP이 첫 번째 이전인지 확인한다. 이 확인 작업은 매우 중요한 일인데 만약 첫 번째 STEP 이 아닌 곳에서 수정된 프로그램의 절환이 이루어 지면 오동작을 일으킬 수 있다. 이러한 오류는 공정 제어에서 치명적인 사고로 이어질 위험이 있기 때문에 신중한 안정 장치가 필요하다. 때문에 온라인 다운로드가 요청된 때가 SFC의 중간 STEP이 실행 중이라면 첫 번째 STEP이 시작되기 전 즉, 마지막 STEP이 수행될 때까지 기다린다. 모든 STEP이 수행되고 첫 번째 STEP 이전이면, Part Online Download 관련 작업을 수행한다. 작업이 끝나면 새로 수정된 실행하고자 하는 SFC를 다시 시작시킨다. 그러면 SFC에서는 첫 번째 STEP 이전에서 자연스럽게 기존 코드와 새로운 코드가 절환이 되고 새로운 코드는 첫 번째 STEP에서 수행하게 된다. 이 일련의 과정을 흐름도로 정리를 해보면 [그림 4]와 같이 나타낼 수 있다.



[그림 4] 온라인 다운로드 순서도

실제 태스크 수행 시 온라인 다운로드 요청이 들어왔을 때 태스크의 실행상태를 그림으로 살펴보면 다음과 같다.



[그림 5] 태스크 실행 상태

[그림 5]에서 보는 바와 같이 3개의 태스크가 실행 중 T1의 수정이 필요하여 프로그램 변경 후 T1을 온라인 다운로드 하고자 한다. 그림의 T1, T2, T3 세 개의 태스크는 현재 수행 중인 일을 그대로 진행하게 된다. 문제는 T1의 태스크를 그림에서처럼 태스크 실행 중간에 요청이 들어올 때가 있다. 이때 T1은 실행 중인 SFC의 마지막 스텝까지 실행한다. 한 스캔이 끝난 후 다음 SFC의 첫 번째 STEP이 실행 될 때는 t3 부터는 교체된 SFC가 실행 되는 것이다.

2.3 실험 개발 환경 및 결과

위에서 제안한 알고리즘을 적용한 개발 환경은 다음의 [표1]과 같다.

운영 체제		vxWorks
제조 회사		WindRiver
개발 툴		Tornado
구조		멀티 쓰레드 방식 (Multi-thread)
BSP (Board Support Package)		MV5100
CPU Board	보드 명	MVME5100-2163
	제조 회사	Motorola
	메모리	512MB
	Clock Rate	450MHz

[표1] 개발 환경

온라인 다운로드 실행 시 교체 절환 시간은 최대 5ms 이내로 이 시간은 현재 개발 시스템의 기본 실행 시간 단위가 1Tick으로 5ms를 기준으로 해서 나온 시간이다.

3. 결 론

실시간 시스템은 주어진 작업이 처리 되어야 하는 마감시간(deadline)이 주어지고, 이 시간이 지켜져야 하는 엄격성에 따라 종류가 나뉘는데, 공장자동화에서의 연속 공정제어와 같은 경우는 경성(hard)실시간 시스템이라고 볼 수 있다. 시스템이 주어진 마감시간을 만족시키지 못할 경우 막대한 재산적 손실이나 인명의 피해를 주기 때문이다. 철강 산업라인이나 자동차 조립라인과 같은 경우도 이와 같은 경우에 해당할 것이다. 때문에 시스템의 마감시간을 지키기 위해 수많은 알고리즘과 태스크 관리를 위한 다양한 스케줄링 기법이 개발되었다.

본 논문에서는 시스템의 마감시간을 지키면서 효율적인 태스크 관리를 하기 위해 온라인 다운로드 기법을 제안하였다. 본 논문에서 제안한 기법은 연속 공정과 같은 산업 현장에서 프로그램을 개발할 경우 온라인 다운로드와 같은 기능이 있음으로 초기 개발에 빈번하게 발생하는 프로그램 수정한 결과에 대해 신속히 알 수 있으므로 개발 속도가 현저히 빨라지는 효과를 얻을 수 있다. 또한 생산라인을 중단하지 못하는 부득이한 주위조건에도 개발자가 원하는 대로 프로그램을 수정하여 곧바로 실행할 수 있음으로 생산라인을 멈춤으로 해서 생기는 손실도 막을 수 있다.

[참 고 문 헌]

[1] 한국기계연구소, "공장자동화를 위한 통합 제어기술 개발", 1991.
[2] 권욱현, 변대규, "프로그래머블 콘트롤러", 전기학회지, 제37권 4호, 1988년 4월
[3] Windriver, "Tornado programmer guide", 2005
[4] 조영조 외, "연속공정 자동화를 위한 Function block Diagram 형 제어언어의 설계 및 구현", 한국자동화공학논문지, pp226-231, 1991.
[5] Tomas Pauly, "the ABB Master system Philosophy", Control Engineering, Vol II, pp 5-8, August 1989
[6] 김광배 외, "연속 공정 자동화용 다기능 제어 시스템의 개발", '91 로보틱스및 자동화연구회 워크샵, pp107-113