FMS 의 실제 시간 제어에 관한 연구

이 석 희
부산 대학교 공과대학 생산 기계 공학과

Real - time Control Software for Flexible Manufacturing System

Seok-Hee Lee
Department of Mechanical and Production Engineering
Pusan National University

Abstract

This paper gives the detail of the work carried out to develop real-time control software for Flexible Manufacturing Systems. A basic design philosophy to implement such software is proposed. The major features are the partitioning of complicated control actions into simplified ones, structured programming and multi-threaded transaction-based tasks.

The software operates on the basis of passing task-to-task messages via mailboxes, causing appropriate actions to be taken by each task.

Each task represents a separate subprocess so that the subprocesses can be run simultaneously. The task-to-task message could be easily replaced by computer-to-computer communication, using LAN, demonstrating that the software methods developed produce a flexible designs for control software of an FMS. A method of linking such software to simulation software is suggested as a potentially powerful additional design-tool.

1. 서론 (Introduction)

지난 몇년동안 F.M.S (Flexible manufactuing-System) 의 개념이 널리 소개되며 이에 관한 전문잡지 및 회의도 많아지고 있다.

이 개념의 중심점은 주로 여러개의 공작기계의 연관 운용, 부품 조작 장치, 자동측정 장치, 컴퓨터등이 에서보는 바와같이 연결되어 있어 컴퓨터의 감독 제어가 행하여 지는 것이다.

새로운 F.M.S 를 개발하는데 있어서 가장 많은 노력이 드는 것중의 하나는 그 제어 소프트 웨어 이다. 그 이유는 FMS 의 구성 요소를 연결하며 고차원의 기능성을 부여하여야 하기 때문이다. 넓은 응용범위에서도 적용될 수 있는 모듈별 설계가 가장 가능한 것이 이 분야인데 비해 실제적으로는 별로 시도되지 않았다.

F.M.S 제어에 관한 논문은 주로 두가지로 분류된다. 첫째는 소프트웨어 설계의 기본이 되는 이론적인 제어구조를 제안하는 논문이고 둘째는 짧은 실례를 소개하는 형태의 논문이 있다. 이것은 시스템 구성요소와 시스템이 무엇을 할 수 있는지를 간단히 소개하고 있을뿐이며 그것의 구체적인 방법에 대해서는 설명하지 않고 있다. 이것은 관련 소프트웨어 개발업체에서의 상업적 독점성 때문으로 기인된다. 따라서 이러한 실제 시간 제어에 대한 상세한 정보를 제공하는 경우는 매우 드문 실정이다. /1, 2/

이 논문은 관련된 기술자료 및 현존하는 논문을 검토하여 위 저자가 개발한 실제시간 제어 소프트웨어 시스템으로 설명함으로 위의 문헌 사이의 간극을 메우고자 한다./3/

이 논문은 또한 장래의 연구방향은 시뮬레이션 분야와 실제 시간 제어 및 그래픽스 등이 통합된 형태의 소프트웨어 도구를 개발하는데에 비중을 두고 있다.

2. 실제시간 제어 기법

대개의 컴퓨터는 그 하드웨어나 소프트웨어의 구성은 매우 복잡하다. 공장의 자동화를 고려할 경우 많은 다른 회사 제품의 컴퓨터를 연결하여야 한다. 여와같이 복잡한 컴퓨터를 연결하는데는 체계적인 접근방법 없이는 관리하기가 힘들어진다.

대형 프로그램 안의 복잡한 소프트웨어를 다루기 위해서 구조적인 설계방법을 사용한다. /4/.

이 방법은 대형 프로 그램을 모듈로 분할해서

(부프로그램 혹은 procedure 별로), 모듈내의 연관을 최대로 하며 (단순 논리 기능을 수행하는 코우드를 선택함으로써) 모듈사이의 상호 연관을 최소화 시킨다. 그리하여 시스템에 변경이 있을때 그에 따른 파급효과를 최소화 한다. 이와같이 모듈내부의 상세한 내용은 그 모듈을 불러내는 모듈로 부터 나타나 있지 않으므로 복잡한 프로그램을 보다 간단한 것으로 나눌수가 있게 된다.

다작업 (multi-tasking) 실제시간 제어 시스템은 소프트웨어 기능을 대화 가능한 분리된 작업 (프로그램 또는 프로세스)로 나눔으로 더욱 확장 될수 있다. 이렇게 함으로써 비동기적 (비계획적)으로 일어나는 사건을 다룰 수 있게 된다. 결과적으로 이것은 가상적 다 프로세스 운용을 가능케하며 실제의 다프로세스 운용에 대한 전환은 비교적 간단해 진다.

개발된 소프트웨어는 FORTRAN 77 언어로 쓰여졌으며 VAX 11/750 의 VMS 의 운용 시스템과 시스템 서비스 루틴이 사용되었다.

2-1 전언전달

다작업 실제 시간제어 시스템의 중요 요소는 전언(message)를 메일박스 /6,7,8/를 통하여 전달 받을 수 있는 분리된 작업들로 나눌 수 있는 점이다.

이것은 각 작업이 간단한 인터페이스를 통하여 복잡한 제어운동을 할수 있게 한다. 예를들면 중앙 절삭 공구 창고의 관리는 매우 복잡한 과정이다. 그러나 그 시스템의 나머지 부분에서 그것은 입력 메일박스와 출력 메일박스로 간단히 나타날 수 있다.

전언은 "GET TOOL SVK123" 으로 보내지며, 회답전언은 "GRANTED" 나 "NOT AVAILABLE" 로서 되돌아 온다.

작업은 서비스를 제공하는 제공자와 서비스를 요구하는 관리자로 나누어 간주될 수 있다.

관리자가 제공자의 관계는 Fig.2 와 같이 수직구조와 Fig.3와 같이 수평구조가 될수 있다.

전자는 중앙통제 및 계단적 명령체계에서 볼 수 있으며 후자는 체계내의 여러 단계에서 필요되는 예를들면 시스템 활동 이력 기록 작업과 같은 보조적인 서비스 제공에서 나타난다.

VMS 메일박스 기구에서 생기는 실제적인 문제로는 (a)전언의 출처와 그내용에 따라 선택적으로 받아들이는 간단한 우선 순위기구가 없다.

이러한 우선 순위 전언은 제 2 메일박스와 비동기 시스템 트랩 (AST)을 사용해서 해결하여야 한다.

(b) 전언을 받아두는 가변공간이 없기때문에 단지 크기가 한정된 전언 대기 공간만이 허용된다. 이러한 대기 공간이 전언으로 가득찰때 시스템 에러가 발생되며 이는 쉽게 발견되지 않는다.

가변 공간이 가능하더라도 전언을 읽어 수행하는 작업은 충분이상으로 높은 우선도가 부여 되어야 한다.

이를 위하여 대기가능한 공간의 함수로 표시되는 가변우선 순위 할당이 필요하게 된다./5/ 이러한 기능이 없으면 시스템의 전언 대기열에서 부터 각 작업 내부의 보다 넓은 영역에 전언을 복사함으로 해결할 수 있다.

2-2 다중 나사식 처리

보통의 프로그램 처리 방식에서 Fig.4a 의 형태의 구조가 종종 발견된다. 이러한 형태는 각 지부가 완전한 처리를 행하기 때문에 설계하기가 비교적 쉬우나 최대 응답 속도는 얻게 되지 못한다.

그 이유는 서비스 A 의 제공 (작업자의 키보드 입력이나 사이클 종료와 같은)을 기다리는 동안 형태 2나 형태 3의 처리를 할 수 없게 된다. 다중 나사식 처리 방법은 Fig.4b 와 같이 기다림이 없는 여러 지부로 나눈다.

이것은 가능한한 빨리 처리를 행할 수 있도록 하는 방법이다. 그러나 이러한 형태는 설계 하기가 힘이 들고 잘못 되었을때 이를 고치는 방법이 특히 어렵다. 실제 시간 제어 소프트웨어를 설계하고 개발하는데 후자의 방법을 사용한다.

2-3 인터럽트

FMS 를 실제 시간 제어를 하기 위해서는 시스템 소프트웨어는 발생하는 사건의 중요성에 따라서 중단될 필요가 있다. VMS 운용 시스템에서 어떤 특별한 사건이 발생하는 경우 프로그램 수행이 중단되도록 요구될 수가 있다. 이러한 인터럽트는 프로그램이 수행될때 특정시간에 순차적으로 발생하는 것이 아니기 때문에 비동기 시스템 트랩 (AsynchronousSystem Trap) 이라고 부르며 여기서 사건을 다룰, 사용자가 지정한 부프로그램이 수행되게 된다. AST 기구를 설정하는 시스템 서비스는 어떠한 사건이 일어날때 수행될 부프로그램의 주소가 하나의 아규먼트로 되어

요청된다. 예를들면, 시스템 서비스 루틴중의 "queue I/o request" 는 메일박스를 신속하게 읽는데 사용되며, 만약 메일박스에 전언이 들어 있으면 AST 루틴이 수행되어 전언을 읽고 버퍼에 저장한다.

전언이 없으면 주 프로그램은 전언이 도착하기를 기다리지 않고 다음의 지령문을 수행하게 된다.

2-4 타이머 작용

대부분의 실제시간 제어 소프트 웨어에서는 규칙적인 시간마다 어떤 프로세스를 작용시켜야 된다. 이와같이 기구의 대표적인 사용에는 다음과 같다.

a) 각장비의 현재의 상태를 알고저 할때 - 이력기록 및 결함의 점검

b) 알려질때까지 경보를 반복 발생 시킬때

c) 시험시 주위장비에 대한 시간 지연이 요구될때

이러한 작업의 형태는 VMS 운용시스템의 SET-TIMER 나 비동기 시스템 트랩을 사용하거나 시간 간격의 특성이 포함된 수행 명령어를 사용한다.

2-5 동기 신호기

실제시간 제어 상황에서는 어떠한 사건들의 동기가 요구되게 된다. 동기 신호기는 시스템 소프트웨어에서 사건의 순서를 제어하게 된다. VMS운용시스템에서는 이벤트프래그를 조작하기 위한 시스템 서비스 루틴이 있다. 어떠한 특정한 이벤트프래그를 설정, 대기, 취소등을 조합 사용함으로써 작업과 작업 사이에 동기 작용을 얻을 수가 있다.

2-6 실제시간 제어 운용 시스템의 실질적인 문제점

실제시간 제어 운용시스템 같이 복잡한 것들을 문제에 적용 시키지 않고 평가하기란 아주 어려운 일이다.

소프트웨어의 적용으로부터 특히 다음의 2가지점이 분명해 졌다.(3)

2-6-1 운용시스템에 어떤 특정한 기능이 없거나 제한되어 있으면 이와 비슷한 효과를 내기 위하여 매우 뒤꼬인 해법이 생기게 된다. 전언의 선택적 수령 방법이 없는 것과 제한된 전언의 대기수 등이다.

2-6-2 실제시간에 입각한 좀더 좋은 통합적 프로그램밍을 위한 환경조성이 필요하다. 다 작업을 보다 분명히 지원해주는 프로그램밍 언어와 복잡한 상호연관 관계를 해결할 수 있는 프로그램밍 오류 검색 방법 등이다.

3. 설계 방법

이미 언급한 바와 같이, 모듈별로 구조적인 프로그램을 짜는 방법은 복잡한 프로그램을 다루는데 매우 효과적인 방법이다. 이 방법은FMS 제어 시스템을 설계하는데 일반적인 방법으로 알려져 있다. 참고자료 10, 11, 12, 13에서 제안된 대부분의 모델들은 계단적 구조이며 저단계의 세부적인 내용은 고단계에서 부터 다루어 지지 않는다. 이로 인해 여러 군데 배치 되어진 컴퓨터 시스템에서 운용될 수 있는 유연성있는 소프트웨어 설계가 가능해진다.

소프트웨어는 여러개의 층으로 이루어진 구성물로 볼수 있으며 여기서 각각의 층들은 프로그램 작성자에게 그 추상성에 따른 가상적인 기계로 간주되어 작성자로 하여금 컴퓨터 하드웨어나 실제 공장의 세부사항들을 일일히 고려해야 되는 수고를 덜어주게 된다.

이러한 층들은 다음과 같이 분류될 수 있다.(10)

단계 0 - 실제 컴퓨터와 그 지령들.

단계 1 - 단계 0로 컴파일되는 포트란 77언어와 파스칼 언어등.

단계 2 - 단계 1과 비슷한 기본 언어 및 지령들.
기본 운용 시스템의 동작을 불러내는 라이브러리 루틴 등 다작업 수행 및 데이타 베이스등이 이 단계에 속한다.

단계 3 - 가장 간단한 시스템의 작동이나 작업이 포함된다. 작동카트의 위치 파악, 로보트프로그램의 전송, 수치제어 공작기계에 싸이클 개시 신호의 전송등.

단계 4 - 이것은 실제 제작 작업에 관한 것으로 일련의 단계 3의 작업이 된다. 어떠한 제작워크센터를 통괄하는 프로그램은 가공물의 장착, 수치제어 프로그램의 하향 전송 및 파악 공구의 공급 등의 연속 작업이 된다.

단계 0-2는 표준 하드웨어나 소프트웨어로 구성되어 있으며 단계 3-4는 사용자가 만들어야 한다. 제작 단계별 구조에는 단계 4가 많이 있는데Fig.5와 같다. 여기서 작업기능별 단계는 가상기계 단계의 상부에 위치하게 된다.

그와같은 시스템에 대해 명확히 구분되는 구조로 나타내는 것은 쉽다고 하더라도 실제로 소프트웨어를 분할하는 것은 상당히 어렵다.

연구를 통하여 얻었던 교훈은 다음과 같다.

a) 고차원에서 저차원의 실제적인 문제를 전혀 다루지 않는다면 시스템은 단순하게 되지만 시스템의 최적화에 대한 가능성을 축소시키게 된다.

b) 위의 경우, 공장 전체에 걸함이 생길때, 문제가 될 수 있다.

예를들면, 어떤 작업량의 공정순서를 다루는 작업에서는 각각의 가공 워크센터가 기능을 수행하고 있는지 알아야 할 필요가 있다. 이것은 고차원에서 저차원 작업으로 질문을 보내어 해결이 될수 있다.

c) 작업단계의 저차원 구성요소에서 부터 같은 단계의 구성요소로 (작업 워크센터에서 부품이동 워크센터로) 전언이 전달되어야 한다면 전체 단계에서 상당한 량의 정보 전달이 생기게 된다. 같은 단계의 구성요소 사이에 직접 전달을 허용함으로써 이러한 현상을 줄일수 있게 된다. 이는 사용된 전언 교환 방법 체계를 제한하게 되며 고차원에서 현재 일어나고 있는 상황을 추적하지 못할 위험성이 있게 된다. 이는 결함이 생겼을때 더욱 더 중요한 요소가 된다.

d) 실제 가공공장과 소프트웨어 시스템은 모두 한정적인 재원을 가지고있으므로 소프트웨어의 어려부분에 순서대기 취급기구가 필요하게 된다. 그러므로 이를 위한 표준 부프로그램을 개발할 가치가 있다.

3-1 공정 순서에 관한 자료

보다 일반적으로 적용할 있는 소프트웨어를 개발하기 위해서 설계 기법이 필요함에도 불구하고 그것만으로서는 충분하지가 못하다. 이를 위하여 자료에 의한 구동기법 즉, 시스템 결정 기구가 데이터 파일에 들어 있는 방법이다. 소프트웨어 설계자는 어느정도 직관적으로 이것을 해나갈 것이다. 예를들면, 가공 순서 파일을 부품이 어느 워크센터에서 가공되어야 하는지를 관리할 것이다.

그러나 완전한 일반성을 갖추기 위해서는 간단한 순서 나열형 보다는 더욱 복잡한 구조가 되어야 할 것이다.

이러한 복잡성은 특징적으로 다음과 같다.

a) 동등한 기능을 가진 워크센터로 인한 변경된 작업배정

b) 워크센터 가공순서의 변경

c) 재작업 또는 일정 횟수의 반복작업

d) 일정제품의 분할작업 및 조립작업

이러한 형태의 실제적인 특징들은 매우 복잡한 데이터 구조를 띄며, 데이터 베이스 관리 방법으로 문제를 상당히 단순화 할수 있을 것이다.

자료에 의한 구동기법은 보다 광범위하게 활용될 수 있다. 다중나사식 전언처리 작업 구조를 나타내고 있는 Fig.4b로 부터 사건 동작의 연속을 볼 수가 있다. 여기서 사건은 전언의 도착을 의미하고 동작은 사건을 처리하기 위한 적절한 조치를 말한다. 더우기, 내부 혹은 외부의 조건에 따른 선택적 동작이 일 수도 있다. 그러하여 사건, 조건, 동작 등의 세 구성요소가 이루어지게 된다.

만약 이것들이 Fig.6 과 같이 데이터 파일로 구성된다면 작업구조는 Fig.7과 같이 간단하게 나타나 진다.

이러한 접근방법과 인공지능 프로그램에 사용되는 추론 규칙 표는 상당히 유사하다. 이러한 추세는 자료에 의해 구동되는 보다 새로운 시뮬레이션 패키지에서 볼 수 있다. (14,15)

4. 시뮬레이션/에뮬레이션 및 그래픽스

개발된 소프트웨어는 1개의 실제 워크센터를 포함하는 여러개의 워크센터를 제외한 나머지 워크센터는 고차원의 작업에서는 간단한 시뮬레이션을 행하는 가상 작업으로 나타내진다. 이는 두가지로 확대 해석을 가능케 한다.

4-1 시뮬레이션 모델과 연결운용

대다수의 FMS 시스템은 시중의 많은 시뮬레이션 패키지 중의 하나를 이용해서 개발된 시뮬레이션 모델과 연관되어 설계 된다. 이것은 종종 매우 값비싼 투자에 대한 위험 부담율을 줄여준다. 이러한 모델은 제어 시스템의 기능적 사양을 발전시킬 목적으로 사용될수 있으나, 통상적으로 제어 시스템 프로그램 작성자에 의해서 사용되지는 않고 있다. 적당한 연결구 (예를들면 메일박스)를 시뮬레이션에 사용하여 그 시뮬레이션 시스템을 제어하는 소프트웨어를 개발하므로 실제 시스템의 제어 소프트웨어의 조기 개발을 가능케 한다.

실제 제작공장의 구성요소를 이루는 기계 및 장비 들이 구비되면 그 각각 장비에 대한 시뮬레이션 대응부와 교환 대치 되게 된다. 이렇게 하여 상당한 양의 자본 투자에 대한 위험 부담율을 줄여 주게되며 생산의 자동화에 대한 가능성을 제시 입증할 수 있게 된다.

4-2 에뮬레이션

실제 시간 시스템에서 가장 큰 문제의 하나는 오류를 추적하는데 이때문에 에뮬레이션 기능의 사용이 제안되었다.(12) 에뮬레이션을 자료로 구동되는데 그것이 충분히 빨리 수행되도록 만들어 진다면 실제 시간 제어에 적용될 수가 있다.

4-3 그래픽스

미믹 도식화는 복잡한 공장에 대한 작업자 기계를 연결해 주는 도구로 오래전부터 사용되었다. 이는 작업자에게 결정적 정보를 빨리 이해하도록 해준다.

대부분의 새로운 FMS 시뮬레이션과 제어 시스템은 칼러 그래픽 터미날을 사용해서 시스템의 동작상태를 볼수 있도록 하고 있다. 이러한 화면화를 설계하는 것은 때로는 매우 복잡한데, 특히 에니메이션이 요구되는 경우는 더욱 복잡하다. 이러한 이유는 시뮬레이션 혹은 실제 시간 제어에 사용되는 언어는 특수한 목적에 맞도록 되어 있으나 저단계의 화면화 기능을 나중에 서로 연결하려면 상당히 힘이 들게 된다.

시중의 그래픽 디자인 패키지중의 하나를 사용하여 도면화를 전담하게 하는 별개의 작업을 설정하면 나머지 작업과는 독립되어 상당히 효과적이 된다.

4-4 통합적 접근 방법

앞의 사실로 미루어 보아, 앞으로의 장기적인 FMS 소프트웨어는 DBMS 와 그래픽스와 연관된 자료로 구동되는 단계적 소프트웨어로 된다.

필요한 것은 이러한 다양한 기능을 통합적 시무시스템과 유사한 방법으로 연관 운용하는 일이다. FMS 설계자는 데이터 베이스, 그래픽스, 시뮬레이션 시스템을 설계해서 실제 제작 시스템을 평가할 수 있어야 한다. 같은 데이터, 그래픽스 자료, 시뮬레이션의 일부를 가지고 제어 시스템의 개발 및 검토를 수행하고, 그후에 실제 시스템을 제어하는 소프트웨어를 만들어 내어야 한다.

5. 결 론

FMS 에 대한 실제 시간 제어를 위하여 먼저 작업량 발주와 작업량 진척 제어 시스템을 개발하였다. 물적 및 인적 자원의 제한으로 인해 상당히 단순화 시켰음에도 불구하고 구조적 설계 기법의 장점과 다 사용자 컴퓨터 시스템에서 쓰여지는 실제 시간 소프트웨어를 사용하는데 생기는 몇몇의 문제점을 보이기에는 충분히 복잡했다.

이 논문을 통하여 FMS 소프트웨어 설계자들에게 많은 도움이 되기를 바라며 간단하고 동시에 좀더 다양한 기능의 소프트웨어를 만드는 기본 지침을 파악하리라 기대된다.

참 고 문 헌

1. 'Software tools necessary for the guidance of Flexible Manufacturing Systems' Haurat, A., Components and Instruments for Distributed Control System, IFAC, pp 189-192, Paris, France, 1982.
2. 'Flexible Manufacturing Systems - State of Art, degree of Automation and Areas of Application', Feldman K. and Willinger R., Siemens Power Engineering, pp 140-143, Vol.1.,
3. 'The real-time control of a robot-based Flexible Manufacturing System', Lee, Seok-Hee, Ph. D. thesis - UMIST, July 1985.
4. 'Using structured design', Stevens, W.P., John Wiley and Sons Inc, USA, 1981.
5. 'Introduction to real-time system design' Allworth, S.T., The Macmillan Press Ltd., UK, 1981.
6. 'System services - system service reference manual', VAX/VMS Documentation, Vol. 4, Digital Equipment Corporation, Maynard, Massachusetts 01754.
7. 'System programming - Real - time user's guide', VAX/VMS Documentation, Chapter 3, Vol. 9, Ibid.
8. 'Introduction to the RMX86 Operating System', Doc. No. 9803124, INTEL Corp., Santa Clara, California.
9. 'User guide for the operating system OSC245', Ferranti Computer Systems Ltd., Wythenshawe, Manchester.
10. 'A generalised approach to the problem of FMS

on-line management', Manara, R., Giuffrc, O. and Covagnaro, F., 1st Int. Conf. on Flexible Manufacturing Systems, Brighton, UK, Oct. 1982.

11. 'Report on software development methodologies and structural design technigues', Snodgrass, B.N., CAM-i Factory Management Project Meeting, Rochester, NY, June 1979.

12. 'The Automated Manufacturing Research Facility of the National Bureau of Standards', Furlani, C.T., Kent, E.W., Bloom, H.M. and McLean, C.R., Summer - Computer Simulation Conference, Vanoouver B.C., Canada, July 11-13, 1983.

13. 'Experimental F.M.S. on-line at NBS', Hatschek, R.L., American Machinist,pp75-77, Jan., 1984.

14. 'A Description of the HOCUS Simulation System', Szymankiewicz, J., P.E. Information S Systems Ltd., Park House, Egham, Surrey, Issue 10, 1984.

15. 'Simulation of F.M.S. with MAST', Lonz, J.E., S.M.E. C.A.S.A. 82, Ref. MS82-165.

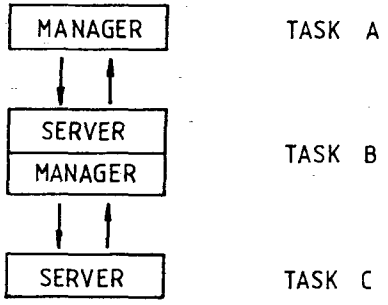

Fig. 2 The Manager-Server relationship in a vertical hierarchy

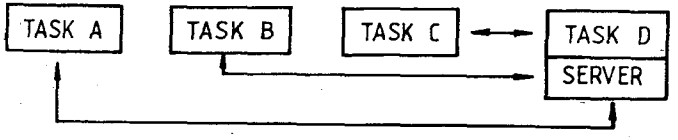

Fig. 3 The Manager-Server relationship for a general service

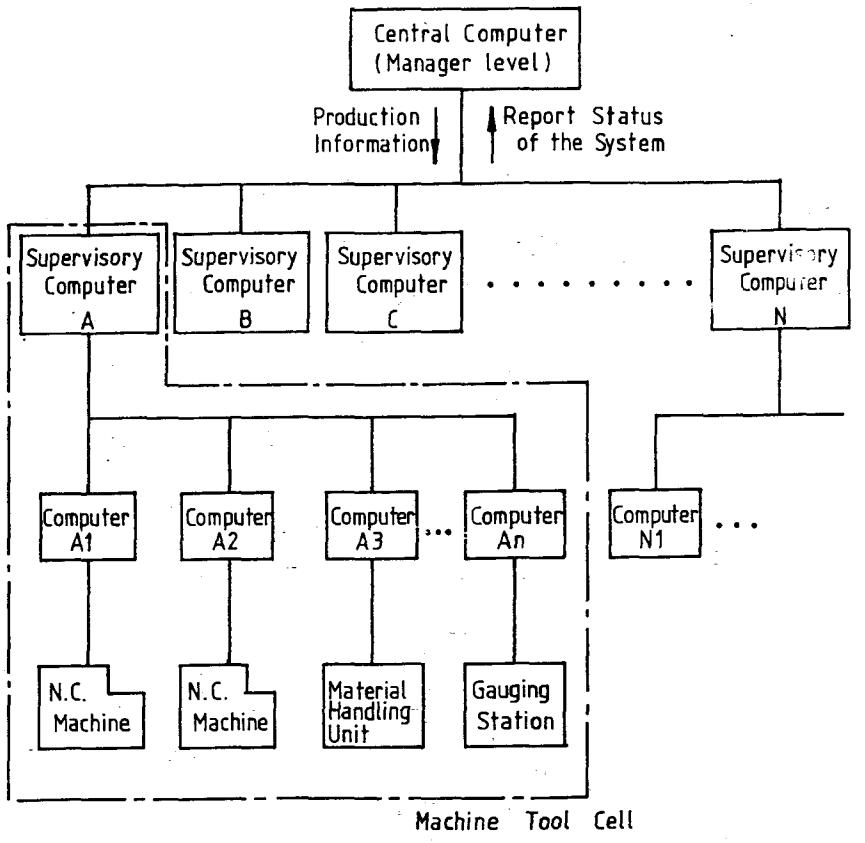

Fig. 1 Basic Concept of a Flexible Manufacturing System

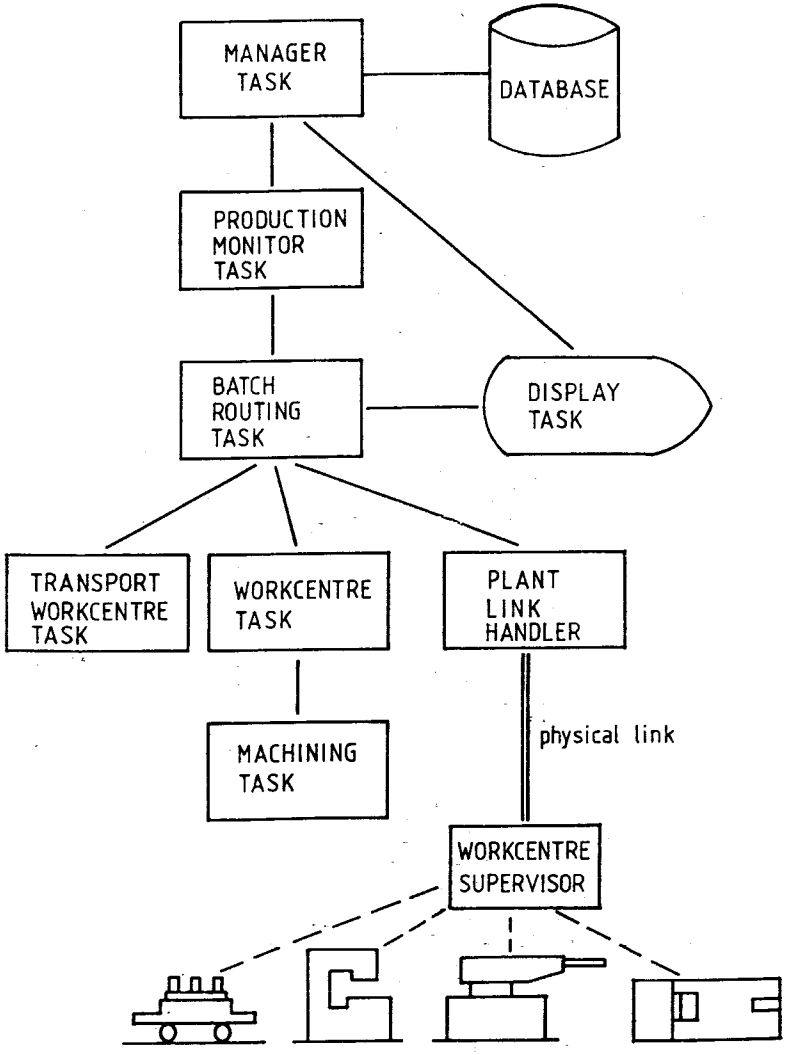

Fig. 5 General diagram of the software system tasks

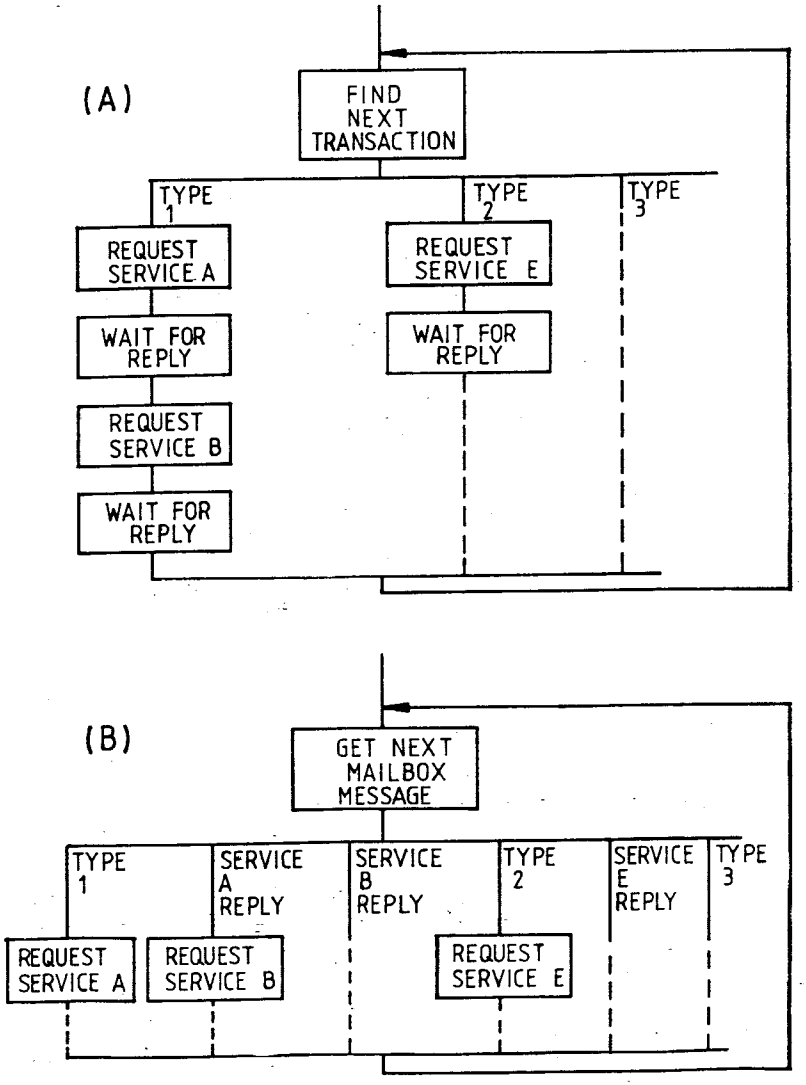

Fig. 4 Comparison of two types of transaction-processing program structure

EVENT	CONDITION	ACTION
BATCH ISSUABLE QUERY	VALID BATCH	SEND POSITIVE ACKNOWLEDGEMENT
BATCH ISSUABLE QUERY	INVALID BATCH	SEND NEGATIVE ACKNOWLEDGEMENT
NEW BATCH PRIORITY	——	UPDATE LIVE-SCHEDULE FILE INFORM BATCH ROUTING TASK
BATCH COMPLETED	——	UPDATE LIVE-SCHEDULE FILE INFORM DISPLAY TASK

Fig. 6 Event-Condition-Action file structure

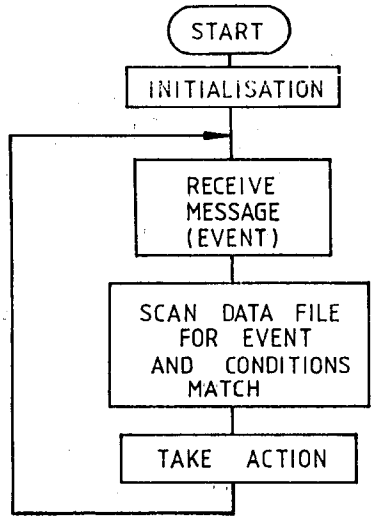

Fig. 7 Data driven task structure