經營科學 第11卷 第1號
1994年 3月

무인운반차시스템의 설계를 위한 시뮬레이션 코-드 작성시스템†

김갑환* · 김판수** · 배종욱*

A Simulation Code Generator for AGVS Design†

Kap Hwan Kim* · Pan Soo Kim* · Jong Uk Bae*

ABSTRACT

We usually use the simulation technique in the evaluation and the test of a design alternative of Automated Guided Vehicle System. In this paper, we introduce a simulation code generator which can assist simulation programmer in model development and programming. It consists of user interface, program editor, and program sorter. SIMAN is used as the target language.

1. 서 론

무인운반차시스템(automated guided vehicle system: AGVS)이란 무인운반차, 안내통로(guide path), 제어기기및 인터페이스장치 등으로 구성되어 작업자의 개입없이 물자를 운송하는 자동화 물류시스템이다.

대표적으로 AGVS가 사용되고 있는 분야를 살펴보면 유통분야에서 자동창고와 함께 다양한 제품을 출납, 배포하는 데 활용되고 있고 가공분야에서는 유연생산시스템(FMS)에서 가공셀(cell) 간의 가공물이나 공구의 이동에 주로 활용되고 있으며 조립반제품의 조립공정 내의 이동이나 조립부품을 공급하는 데 이용되고 있다.

AGVS의 설치에는 많은투자가 필요하기 때문에 시스템의 설계에 있어서 신중한 분석과 평가가 필요하다. 또한 AGVS의 설계작업은 물류의 흐름, 생산시설, 생산계획, 경제성 등을 고려하여 몇 대의 AGV가 필요한지, 배차는 어떻게 하며, 운행경로는 어떻게 결정할 것인지 하는 등의 서로 연관이 있는 많은 물음들을 동시에 해결해야 하는 문제이기 때문에 매우 복잡하다고 할 수 있다.

† 이 논문은 1992년도 교육부 지원 한국 학술진흥재단의 자유공모과제 학술조성비에 의하여 연구되었음.
* 부산대학교 산업공학과
** 주식회사 에스·티·엠

따라서 이 문제는 몇 개의 분석적인 기법으로 완전히 그 해답을 얻을 수 없으며 궁극적으로는 시뮬레이션을 통하여 AGVS 설계안을 분석, 평가할 수 밖에 없는 것이다.

그러나, 시뮬레이션 기법의 적용은 사용자의 입장에서 몇가지 어려운 점이 있다.

첫째, 일반적으로 시뮬레이션언어(FORTRAN, C, SIMAN, SLAM, SIMSCRIPT 등)가 특정문제를 위하여 설계되어 있지 않기 때문에 모델링하는 데 많은 노력과 시간이 든다.

둘째, 시뮬레이션의 통계적인 설계와 결과의 분석이 복잡하다.

이상의 어려운 점들은 시뮬레이션이 실용적인 기법이라고 인정되면서도 국내의 현장에서 많이 사용되지 못하는 이유 중의 하나라고 할 수 있다.

본 논문의 대상이 되고 있는 첫번째 문제를 해결하기 위한 접근방법으로서 입력자료를 대화식(interactive)으로 받아들여 새 변수값으로 시뮬레이션을 수행하는 data-driven 시뮬레이션 방식과 특정 목표(target)언어로 된 시뮬레이션 코드 자체를 작성해 가는 시뮬레이션 코드 작성시스템 방식이 있다. 두번째 방식이 모델링에 있어서 더 많은 융통성을 가지기 때문에 모델링 대상범위가 훨씬 넓다고 할 수 있다.

본 논문에서는 AGVS의 시뮬레이션 모델링을 위한 시뮬레이션 코드 작성시스템(code generator)의 개발을 내용으로 하며 이는 AGVS의 설계시 사용할 통합적 소프트웨어 환경을 구축하기 위한 기초연구작업이라 할 수 있다.

타 분야에서의 코드 작성시스템에 대한 연구는 오래전부터 이루어져 왔으나 [7,9,10] AGVS에 대한 연구는 주로 AGVS 를 연구하는 연구자에 의해서 simulator의 형태로 만들어져 사용되어 왔다.

1982년 Egbelu[3]는 운영문제에 대한 연구를 하기 위하여 Fortran언어로 된 simulator(AGVSim)를 만들었는데 이는 일정한 형식으로 입력자료 화일을 만들어서 입력시키는 data-driven simulator였다.

또, Gaskin과 Tanchoco[4]는 AGVS의 controller를 설계하기위한 수단으로써 AGVSim2라는 C언어로 된 simulator를 만들었다.

이것들 보다 다소 상위의 시뮬레이션 전용언어를 사용한 것이 SIMAN언어를 사용한 D.A. Davis [1]와 J.T. Lin [8]의 simulator로서 Davis는 재공품의 운행통로정책을 비교하기 위한 것이었고 Lin의 것은 animation모듈과 graphic입력 모듈을 추가한 것이 큰 기여라고 할 수 있다.

이상의 것들은 대부분 특수한 연구과제를 위한 것이었기 때문에 범용성이 적었으며 이외에도 사용언어와 사용목적만 다르지 유사한 여러 simulator가 만들어져 사용되어 왔다.

범용성을 갖추기 위하여 최초로 시도된것이 D. C. Gong과 L.F. McGinnis [6]에 의한 코드 작성시스템으로서 이는 원래 AGVS 설계를 위한 engineering workstation의 구축의 한 모듈로서 만들어진 것이다. 그러나 작성되는 시뮬레이션 프로그램의 기능면에서 기존의 data-driven simulator와 큰 차이가 없고 확장을 위한 배려가 충분치 못하였다.

본 논문에서 제시하는 시스템은 프로그램 베이스의 구축과 활용을 통하여 시스템 사양의 범위를 대폭 확장하고 특히 운영규칙의 선택에 있어서 성능연구의 가치가 있는 것을 포함할 뿐 아니라 사용자가 설계한 규칙을 손쉽게 접할 수 있도록 개발하였다.

다음장에서는 AGVS를 묘사하는 데 필요한 여러가지 구성요소를 나열하고 설명하였다. 제 3장에서는 대상으로 하고자하는 시뮬레이션 코드를 이용하여 어떤 시스템 설계문제를 다룰것인가를

설정하여 그런 문제해결이 도움이 되도록 대상 시뮬레이션 코드의 사양을 정하였다.

제 4장에서는 개발한 코드 작성시스템을 모듈별로 설명하였다.

부록에서는 코드 작성시스템의 수행과정을 예시하였다.

2. 대상시스템의 정의

AGVS의 특성과 활동을 결정할 주요구성요소를 살펴보면 운반대상물(product), 운반차량(vehicle), 운반출발지와 목적지(workstation), 운반통로(guide path network), 운반운영규칙(control logic)으로 구분할 수 있다.

2.1 운반대상물(product)

운반요구를 표현하는 방법에는 상세한 정도에 따라서 여러가지가 있다. 개략적인 형태로는 품목의 구분없이 from-to chart형식과 같이 작업장간의 흐름량 만을 나타낼 수도 있으나 본 연구에서는 정보를 변환하지 않고서도 현장에서 쉽게 구할 수 있는 품목별 공정설계정보의 형태 그대로 이용할 수 있게 한다.

즉, 품목간 생산구성비와 품목별 lot-size의 분포, 그리고 품목별 공정순서와 공정별 준비시간, 개당 가공시간 등으로 운반요구의 특성을 표현하였다.

단, 조립공정을 나타내는 공정이나 품목은 현재로서는 대상으로 하지 않고 있다.

2.2 운반차량(vehicle)

무인운반차라고 불리우면서 생산현장에 사용되고 있는 차량들로서 towing vehicle, unit-load vehicle, pallet truck, fork truck, light-load vehicle, assembly-line vehicle 등으로 그 기능에 따라 구분할 수 있다.

그리고, 시스템의 특성이 영향을 미치는 요소로는 차량의 댓수, 차량속도, 차량의 길이, 적재 load의 수, battery 충전시간, 고장특성 등이 있으나 본 연구에서는 우선 앞의 세 요소만 고려하였다.

2.3 운반출발지와 목적지(workstation)

운반의 목적은 궁극적으로 가공을 하기 위한 것이니 작업장에 대한 정보가 필요하다. 각 작업장에 관한 정보로는 작업장내 동일기계의 댓수, delivery point 와 pickup point 의 위치(node 번호), input buffer와 output buffer의 갯수 등을 들 수 있으며 본 연구에서 모두 고려 하였다.

2.4 운반통로(guide path network)

무인운반차의 guidance방법으로서 일반적으로는 설치하는데 비용이 들기때문에 쉽게 변경할 수 없는 고정식 guide path(기계식, inductive, optical, chemical, magnetic 등)가 사용된다. 그러나 최근에는 vision 등을 이용하여 일정한 수준 이상의 자율주행이 가능한 차량의 경우에는 시스템 제어컴퓨터의 메모리상에서만 path가 형성되어 순간순간 guide path의 변경이 가능한 시스템이 소개되고 있는데 이를 free-ranging guide path 또는 virtual guide path 등으로 부른다. 그것이 일시적이든 반영구적이든 간에 guide path network는 필요하고 따라서 어떤 형태로든지 시스템을 묘사할 때 포함되어야 한다.

일반적으로 운행통로망(guide path network)을 표현하는 방법으로써 교차로, pickup point,

delivery point를 마디(node)로 나타내고 운행통로를 link로 나타내는 방식이 많이 사용되고 있다.

따라서 어느 마디가 교차로이며 어느 마디가 입구(entering node)를 나타내고 어느 마디가 출구(outgoing node)를 나타내는지를 정해줄 필요가 있으며, 각 link에 대해서 양 끝의 마디를 정의하고 link의 종류(unidirectional, bidirectional, spur)와 길이 등이 설정되어야 한다.

또한 유휴차량의 운영정책에 따라 주차장의 위치나 순환 루-프가 하나의 마디나 여러개의 마디에 의해서 정의 된다.

2.5 운반작업운영규칙(control logic)

시스템의 성능에 영향을 주는 운영상의 문제로서 출발지와 목적지 사이를 어떤 운행경로를 따라 갈 것인가(routing), 차량과 운반요구를 어떻게 짝지울 것인가(dispatching), 통행방해(blocking)의 경우 어떻게 해결할 것인가(traffic control), dead-lock의 해결방법은 무엇인가 하는 등의 문제들이 있는데 이들 하나하나에 대한 운영논리가 시스템의 운영에 필요하다.

전체적으로 말해서 시뮬레이션 코드 발생기의 성능은 이상에서 묘사한 다양한 AGVS의 특성을 어떻게 시뮬레이션 프로그램에 잘 반영하느냐에 달려있다.

3. AGVS에 대한 여러가지 설계문제들

편집되는 시뮬레이션 프로그램이 설계대안을 평가하는데 쉽게 사용될 수 있기 위해서는 AGVS의 여러가지 설계대안들이 user-interface를 통해서 충분히 묘사될 수 있어야 한다. 그와 같은 요구사항을 만족시키기 위해서는 코드 작성 시스템의 제작단계에서 시뮬레이션에 의해서 지원할 AGVS의 주요한 설계문제에 어떤 것들이 있는지 조사해 볼 필요가 있다.

아래에는 시스템 설계시에 결정해야 하는 주요한 문제들을 열거하여 본다.

3.1 운행통로(guide path)의 설계와 작업장 배치문제

운행통로를 설계한다는 것은 주어진 운반요구를 고려하여 총 운행시간이 최소화될 수 있도록 운행통로를 설정하고 각 단위통로(segment)의 운행방향을 결정하는 것이다.

뿐만 아니라 각 작업장의 pickup point와 delivery point의 위치를 정하거나 작업장 자체의 배치를 하는 것도 중요한 의사결정사항 이라고 할 수 있다. 유휴차량을 위한 주차지역의 설정이나 순환경로의 결정도 시스템의 성능에 큰 영향을 미칠 수 있고 battery charger의 위치결정도 중요한 문제이다.

작업장의 buffer space의 갯수도 시스템의 성능에 큰 영향을 미치는 설계요소이다.

본 연구의 시스템에서는 사용자가 직접 자세한 운행통로와 작업장배치, buffer 갯수 등을 입력시킬 수 있게 함으로써 여러가지 대안을 쉽게 평가할 수 있게 하였다.

또, user interface에 의해서 입력된 사양들이 일단 데이타화일의 형태로 저장되기 때문에 최적화 모듈이 개발되어 설계안을 자동적으로 계산해 내는 경우 데이타화일을 직접 access하여 수정할 수 있게 하였다.

3.2 차량의 댓수결정문제

차량자체의 가격이 비싸기 때문에 적은 수의

차량으로 운반수요를 감당해야 되겠지만 그렇게 될 경우 운반지연으로 인해서 작업장의 가동률이 저하될 가능성이 많기 때문에 차량의 댓수를 결정하는 문제는 중요하다.

기본적으로 차량의 댓수와 시스템의 성능간의 관계가 분석적인 방법으로는 규명되지 않았기 때문에 시뮬레이션에 의존하고 있다.

본 논문의 코드작성시스템에서는 사용자가 데이타 입력시 그 댓수를 지정해 주기만 하면 시뮬레이션이 되도록 되어 있어 여러가지 대안들을 평가해 볼 수 있다.

3.3 시스템 콘트롤러의 논리설계

시스템 콘트롤러가 수행하는 기능들로는 운반작업할당(dispatching), 운행경로결정, 유휴차량 운영, blocking 해결, dead-lock 해결 등이 있다.

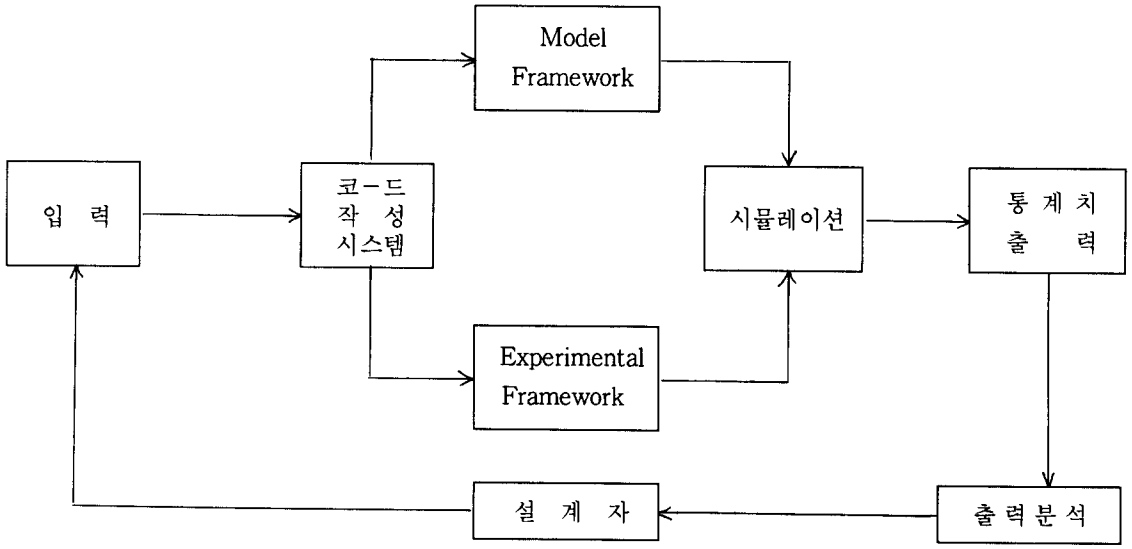
운반작업할당규칙은 SIMAN이 제공하는 기본적인 규칙 이외에도 사용자가 서브루틴기능을 이용하여 할당규칙을 작성할 수 있게 하였으며 운행경로 결정도 SIMAN이 제공하는 최단거리경로 이외에도 사용자가 지정하는 임의의 경로로도 운행이 가능하게 하여 향후 사용자가 고안하는 다양한 경로결정방식을 수용할 수 있게 하였다.

blocking의 해소규칙도 다양한 규칙을 사용자가 선택할 수 있게 하였고, dead-lock해결방식으로는 unit-load를 pickup할 때 마다 도착작업장의 buffer space의 유무를 사전에 확인하게 함으로써 dead-lock이 발생할 수 있는 요인을 제거하였으나 더욱 효율적인 논리의 개발이 필요하다고 생각된다.

4. 시뮬레이션 코드 작성시스템의 구조

4.1 시뮬레이션 코드 작성시스템의 전체적 구조

[그림 1]은 시뮬레이션 코드 발생기를 이용한 시뮬레이션 절차를 나타낸다. 우선 사용자가 시뮬레이션하려는 시스템의 사양을 입력하면 코드



[그림 1] 시뮬레이션 절차

작성시스템이 SIMAN system에 맞추어 모델 frame과 실험frame을 작성해 낸다. 이렇게 작성된 화일들을 SIMAN에 입력시켜 집행시키면 시스템의 성능에 대한 결과치가 출력되고 이것을 시스템설계자가 분석하여 새로운 설계대안을 구성한 후 새로운 사양을 입력하여 사양을 개선해 나가는 과정을 되풀이 하게된다.

일반적으로 시뮬레이션 수행과정에서 많은 노력을 필요로 하는 부분이 시뮬레이션 코드를 작성하는 부분과 통계출력치로 부터 새로운 설계대안을 고안하는 부분인데 본 연구에서는 코드 작성작업을 자동화 하겠다는 것이다.

[그림 2]는 시뮬레이션 코드 작성시스템 구조를 보여주고 있다.

우선 사용자 인터페이스를 이용하여 사용자는 자신이 설계한 AGVS의 사양을 입력한다. 그러면 사용자 인터페이스는 사용자의 선택사양을 데이타화일의 형태로 저장한다. 개발된 사용자 인터페이스는 pull-down메뉴방식으로 작성되어 있고 앞 장의 여러가지 설계대안들이 대화식으로 입력될 수 있게 하였다.

화면의 예가 부록에 나와 있다.

사용자 인터페이스에 의해서 작성된 미가공 데이타 화일의 예는 [그림 3]과 같다. 이 화일은 앞으로 AGVS의 설계에 있어서 여러가지 설계요소에 대한 최적화 모듈들이 개발되어 사용되는 경우 미가공 데이타화일을 직접 접근(access)하여 그 최적화 결과를 설계대안으로서 화일속에 담아 놓거나 수정할 수 있게 될 것이다.

예를들어 운행통로(routing)의 대안을 작성하는 최적화 모듈이 저장되면 그 최적화 결과를 직접 이 미가공데이타화일속의 해당부분과 교체하여 시뮬레이션해 보게 할 수 있다. 또 운행통로 방법중에서 최단거리방식이 아닌 사용자 자신의 운행통로 앨고리즘을 개발하여 테스트해 보는 경우에도 해당부분의 자료만 교체하여 시뮬레이션해 볼 수 있을 것이다. 따라서 이런 최적화 모듈을 개발할 때 미가공 데이타화일을 직접 접근하고 수정할 수 있게 하는 것이 필요하다.

목표언어인 SIMAN은 시뮬레이션 논리를 표현하기 위한 모델 화일과 실험조건을 나타내기 위한 실험 화일로 나누어서 작성하게 된다. 그리고 이 두가지 화일은 그 syntex의 성격상에도 차이가 많아 실험 화일의 경우는 그 특성상 비슷한 레코드의 반복횟수에는 변화가 많으나(예를들어, 작업장의 수나 제품의 수에 따른 데이타 반복횟수의 증가) 사용되는 레코드의 종류에는 차이가 적은 반면 모델 화일은 선택사양의 변화에 따라 프로그램 논리자체의 변화가 크므로 사용되는 프로그램 문장자체에 차이가 많아지게 된다.

따라서, 미가공 데이타화일을 보고 제일 먼저 검토하여야 하는 것은 어떤 선택사양이 모델 화일 상의 변화를 주는 것인지, 아니면 실험 화일상의 변경을 요구하는 것인지를 구분하는 것이다.

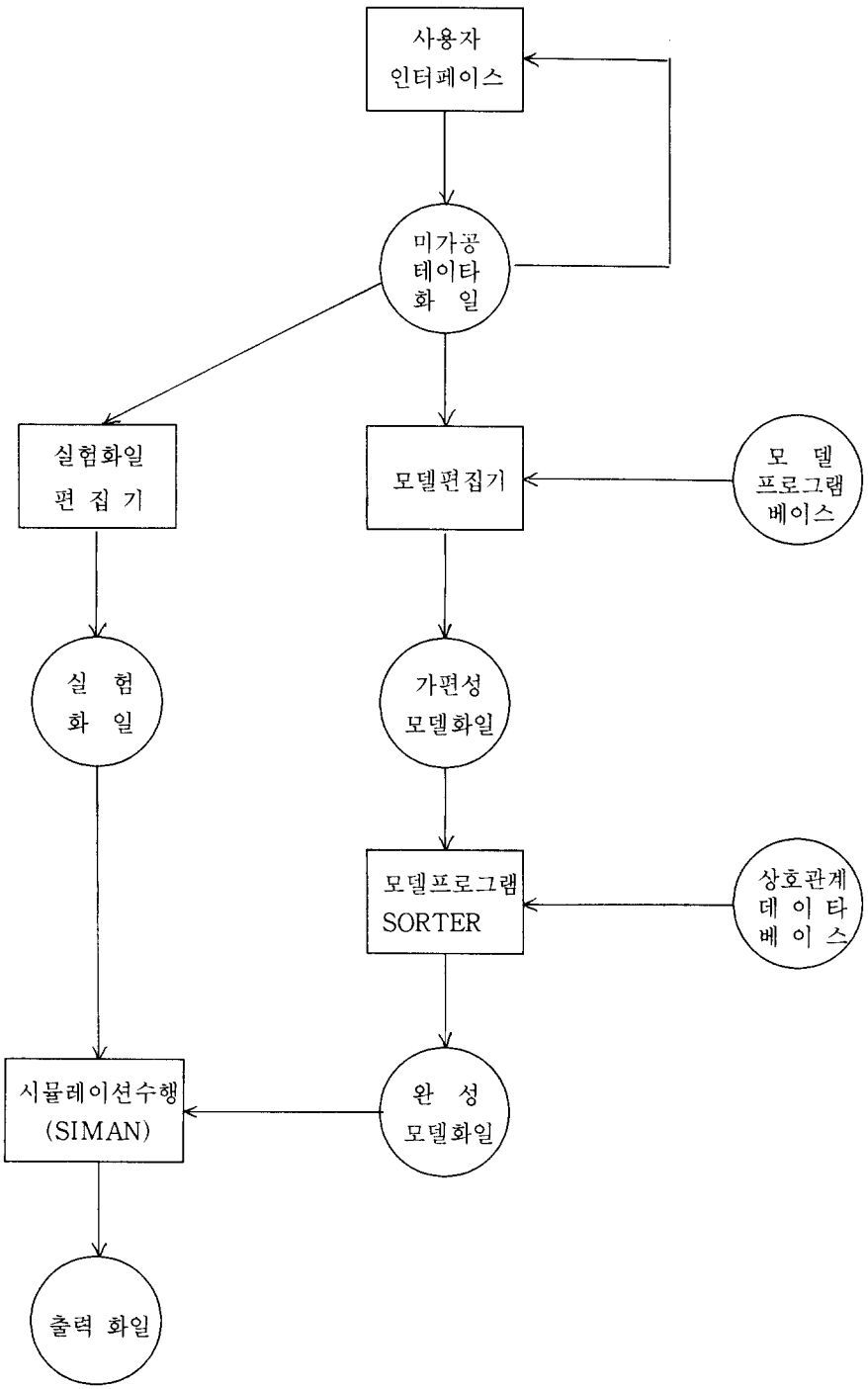
4.2 실험화일 편집기의 구조

실험화일은 실험화일편집기에 의해서 직접 작성하게 하였으며 작성된 실험화일의 예가 [그림 4]에 나와있다.

실험화일 편집기는 실험화일의 성격상 그 요소간의 상대적위치가 중요하지 않으므로 편집이나 순서정리와 같은 기능이 필요없고 반복해야 하는 데이타 갯수나 수치데이타의 갯수 등만을 고려하고 입력자료를 편집하여 출력한다.

4.3 모델화일 편집기의 구조

모델화일은 엔터티(entity ; 본 시스템에서는 가공부품)의 흐름을 묘사하기 때문에 선택사양에



[그림 2] 시뮬레이션 코-드 작성시스템의 구조

〈표 1〉 프로그램 베이스의 형식

A	B	C	문장이름 (block label)	문장 (block)
0	0	1	z	begin, no ;
0	0	2	z	create : ed(1):mark(timein) ;
1	1	1	z	request : agv(Agv-Asnd) : next(h1) ;
2	2	7	n6	assign : Netcl = 1 : next(n2) ;

따라 프로그램문장 자체가 변화한다. 그러므로, 문장의 선택이나 문장 사이의 순서가 중요하다.

본 연구에서의 접근방법은 요구되는 선택항목 상의 여러 선택옵션에 대해서 사전에 프로그램 베이스에 필요한 프로그램 레코드들을 작성하여 둔 다음, 사용자가 특정옵션을 선택하게 되면 필요한 프로그램 레코드를 프로그램 베이스에서 선택적으로 추출하여, 순서를 정리함으로써 모델화일을 완성한다.

그러는 중에 SIMAN이 직접적으로는 제공하지 못하지만, AGVS 운영에 필요한 운영규칙들이 있는 경우에는 이를 사용자에게 제공하기 위해서 사용자 정의함수나 서브루틴을 작성하여 추가하였다.

프로그램 베이스의 예가 [그림 5]에 나와 있고, 모델 프로그램 편집기에 의해서 작성된 가편성 모델화일의 예가 [그림 6]에 나와있다.

프로그램 베이스에서 A는 모델화일상에 변화를 주는 사용자가 입력할 선택항목의 번호(예를 들어, 경로정책 선택항목이면 1, 주차정책 선택항목이면 2)를 나타내며, B는 A의 선택항목에 대해서 사용자가 선택한 선택사양번호를 나타내고, C는 선택사양별 문장의 일련번호를 나타내는데 A, B, C가 합쳐져서 각 문장의 고유번호 구실을 한다.

예를들어 사용자가 경로정책이란 선택항목(A=1)에 대해서 최단거리규칙이란 선택사양을(B=1)을 선정하였다면 프로그램 베이스에서 A=1, B=1인 값을 가지는 모든 문장이 가편성 대상 문장이 된다. 즉, 모델화일 편집기는 A의 선택항목에 대해, B를 보고 그 선택항목에 필요한 문장인지 확인한 후 필요한 문장이면 선택해서 가편성 화일을 작성한다. 문장이름중 Z는 문장이름이 필요없는 문장의 가상(dummy) 문장이름이다.

가편성 모델화일을 작성하는 방식은 다음과 같다.

1. 사용자 인터페이스 프로그램에 의해 만들어진 데이타 화일을 읽는다. 이 중 일부는 프로그램 관련 데이타이고, 일부는 수치 관련 데이타이다. 프로그램 관련 데이타의 예로는 운행방법과 유휴차량의 운영방법 등 이고 수치관련 데이타의 예로는 작업장의 수, 혹은 운반제품의 수 등 이다.
2. 가편성 모델화일 제작의 처리절차는 아래와 같다. 프로그램 선택사양에서 i 번째 선택항목에서 j 번째 선택사항을 선택하였을 때 A = i, B = j라고 나타내자.

   단계 0 : k = 0. 모든 필수 문장들을 읽어들여 프린트한다. 이들은 프로그램 베이스에

서 항목번호가 0인 문장들로 선택사양에 관계없이 꼭 필요한 문장이다.

단계 1 : k = k + 1. 만약 k가 최대선택항목의 수보다 크면 멈추어라. 선택항목 k에 대해서 선택사항이 j라면 A = k, B = j인 문장들을 프로그램 베이스에서 찾아서 가편성 모델화일에 프린트한다. 단계 1을 반복한다.

가편성 모델화일은 프로그램 베이스에서 필요한 문장만 뽑아서 모아놓은 것이기 때문에 모델화일을 완성하기 위해서는 이를 순서대로 정리하는 것(sorting)이 필요하다. 이와 같은 작업을 하는 것이 모델 프로그램 순서정리기의 기능이다.

프로그램 순서정리기는 프로그램 문장 간의 상호순위에 대한 정보를 가지고 있는 상호관계 데이타베이스를 참조하여 프로그램 문장을 순서대로 정리하게 된다.

상호관계 데이타베이스 형식은 〈표2〉와 같은데 첫번째 A, B, C가 나타내는 문장(모두 S1이라 하자)과 두번째 A, B, C(모두 S2라 하자)가 나타내는 문장의 상호위치관계를 R이 표시하고 있다. 상호위치관계에서 R의 값이 1이면 S1의 문장이 프로그램의 가장 처음에 위치해야 한다는 뜻이고, 2이면 S1은 S2의 바로 뒤에 와야한다는 뜻이며, 3이면 S1은 S2의 바로 앞에 와야 하는 것을 의미한다. 또, R의 값이 4이면 S1문장이 프로그램의 가장 마지막에 위치하여야 한다는 것을 의미한다.

순서정리기는 상호관계 데이타베이스를 참조하여 가편성 모델화일 내의 문장들에 대한 선후순서를 결정한 후 sorting된 완성모델화일을 작성하는 기능을 한다.

모델프로그램 순서정리기에 의해서 순서가 정리되어 최종적으로 작성된 완성모델화일의 예가 [그림 8]에 나와 있다. [그림 9]는 완성된 모델화일과 실험화일을 실행시켰을 때의 결과를 보여주고 있다. 여기서 각 부품의 작업장내 흐름시간과 각 자원(resource)들의 활용도등의 정보를 알 수 있다. 실험화일과 모델화일의 작성이 완료되면, 일반적인 SIMAN 수행절차에 의해서 프로그램을 수행하면 된다.

사용자 subroutine은 사용자 인터페이스중 기능선택화면에서 선택을 하면 자동으로 접속가능하도록 프로그램되어 있다. 구체적인 접속방법은 SIMAN 사용자 설명서 [11]에 의한 방법을 따랐다.

본 시스템은 사용자 인터페이스의 초기화면 상태(부록의 [그림 10])에서 데이타 입력, 프로그램 코드 작성, 실험, 결과치 출력 등의 일련의 과정이 수행가능하다.

〈표 2〉 상호관계 데이타베이스의 형식

S1			S2			R
A	B	C	A	B	C	
0	0	2	0	0	1	3
0	0	3	0	0	2	3

```
4
1     ws1     1     1     1     2     2
2     ws2     1     2     2     2     2
3     ws3     1     3     3     2     2
4     ws4     1     4     4     2     2
12
1      2      3      4      5      6      7      8
9     10     11     12
6
5
17
1     1      7     8     1
2     7      2     4     1
3     7      4     4     2
4     4      8     8     1
5     2      8     6     1
6     8      3     7     3
7     3      5     25    1
   .     .     .     .
   .     .     .     .
   .     .     .     .
```

[그림 3] 미 가공 데이타 화일

```
BEGIN,no,yes;
PROJECT, Simulation Code for AGVS,USER;
ATTRIBUTES:Agv#:
                Timein:
                Agv-Asnd:
                NetSt:
                NextOper:
                NetCl:
                ProTime;
QUEUES:ws1-q,FIFO:
        ws1-in-bf-q,FIFO:
        ws1-ou-bf-q,FIFO:
        ws2-q,FIFO:
        ws2-in-bf-q,FIFO:
        ws2-ou-bf-q,FIFO:
        ws3-q,FIFO:
        ws3-in-bf-q,FIFO:
        ws3-ou-bf-q,FIFO:
        ws4-q,FIFO:
        ws4-in-bf-q,FIFO:
        ws4-ou-bf-q,FIFO:
        agvque1:
        agvque2;
RESOURCES:machine(4),1,1,1,1:
            buffer-in(4),2,2,2,2:
    .           .           .
    .           .           .
    .           .           .
```

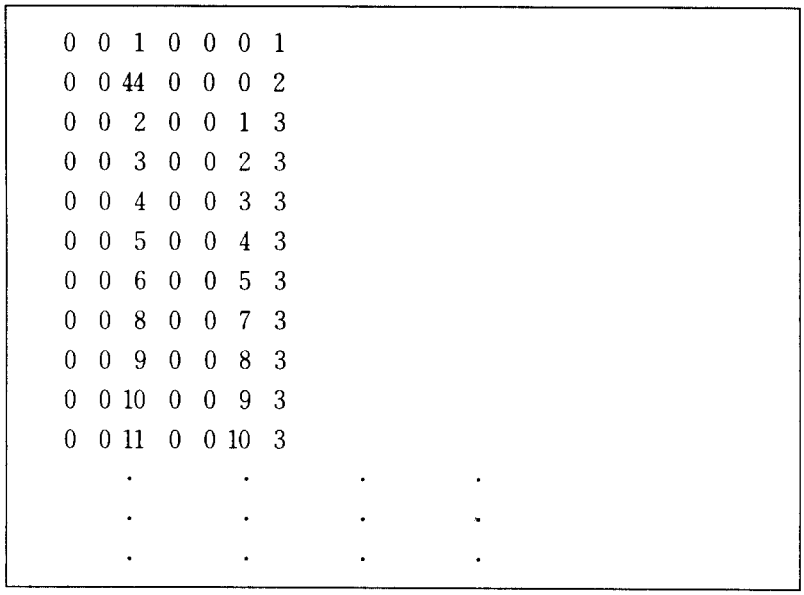
[그림 4] 작성된 실험화일

```
0 0  1 z    begin,no;
0 0  2 z    create:ed(1):mark(timein);
0 0  3 z    branch,1:if,nument.gt.maxent,a1:else,a2;
0 0  4 a1   assign:m=m+1:dispose;
0 0  5 a2   assign:m=enter:ns=ed(2):nextoper=1;
0 0  6 z    count:enter:next(b1);
0 0  7 b1   assign:NetSt=msq(ns,NextOper);
0 0  8 z    branch,1:if,NetSt.eq.exitsystem,c2:else,c1;
0 0  9 c1   queue,NetSt*3-1;
0 0 10 z    seize:buffer_in(NetSt);
0 0 11 c2   assign:Agv_Asnd=UF(1);
0 0 12 z    assign:Agv_Asnd=d(1);
0 0 13 z    branch,1:if,Agv_Asnd.eq.0,e1:else,d1;
0 0 14 d1   assign:S(Agv_Asnd)=1;
0 0 15 z    queue,agvque1;
1 1  1 z    request:agv(Agv_Asnd):next(h1);
1 2  1 z    allocate:agv(Agv_Asnd);
    .       .       .       .
    .       .       .       .
    .       .       .       .
```

[그림 5] 프로그램 베이스

```
1 z    begin,no;
2 z    create:ed(1):mark(timein);
3 z    branch,1:if,nument.gt.maxent,a1:else,a2;
4 a1   assign:m=m+1:dispose;
5 a2   assign:m=enter:ns=ed(2):NextOper=1;
6 z    count:enter:next(b1);
5 4z   end;
7 b1   assign:NetSt=msq(ns,NextOper);
8 z    branch,1:if,NetSt.eq.exitsystem,c2:else,c1;
9 c1   queue,NetSt*3-1;
10 z   seize:buffer_in(NetSt);
   .       .       .       .
   .       .       .       .
   .       .       .       .
```

[그림 6] 가편성 모델화일



[그림 7] 상호관계 데이타베이스

```
    begin,no;
    create:ed(1):mark(timein);
    branch,1:if,nument.gt.maxent,a1:else,a2;
a1  assign:m=m+1:dispose;
a2  assign:m=enter:ns=ed(2):NextOper=1;
    count:enter:next(b1);
b1  assign:NetSt=msq(ns,NextOper);
    branch,1:if,NetSt.eq.exitsystem,c2:else,c1;
c1  queue,NetSt*3-1;
    seize:buffer-in(NetSt);
c2  assign:Agv-Asnd=UF(1)
    assign:Agv-Asnd=d(1);
    branch,1:if,Agv-Asnd.eq.0,e1:else,d1;
d1  assign:S(Agv-Asnd)=1;
    queue,agvque1;
    allocate:agv(Agv-Asnd);
    assign:A(1)=Agv-Asnd;
f1
move:agv,intx(nextx(1,lt(agv,A(1)),ROUT(lt(agv,A(1)),INSC(m))))
;
    branch,1:if,INSC(m).eq.lt(agv,A(1)),h1:else,f1;
e1  queue,agvque2;
    request:agv(sds,1);
    assign:Agv-Asnd=A(1);
h1  assign:S(Agv-Asnd)=2;
    delay:TRFTIME;
    branch,1:if,m.eq.enter,j1:else,i1;
i1  release:buffer-out(m-NUMWC);
j1
move:agv,intx(nextx(1,lt(agv,A(1)),ROUT(lt(agv,A(1)),INSC(NetSt
))));
k1  branch,1:if,INSC(NetSt).eq.lt(agv,A(1)),l1:else,j1;
l1  route:0,seq;
    count:m;
    .       .       .       .
    .       .       .       .
    .       .       .       .
```

[그림 8] 완성 모델화일

SIMAN IV-License #9150420
Pusan National University
Summary for Replication 1 of 1

Project: Simulation Code for AGVS
Analyst: KIM P.S.
Replication ended at time : 5500.0

Run execution date : 12 / 9 /1993
Model revision date : 12 / 9 /1993

TALLY VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Observations
1-flowtime	481.43	.36556	134.01	711.37	103
2-flowtime	417.91	.44673	108.26	650.94	40
ALL-flowtime	463.66	.38964	108.26	711.37	143

DISCRETE-CHANGE VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Final Value
Loaded AGV util.	21.771	1.3644	.00000	100.00	33.333
UnLoaded AGV uti.	22.153	1.2733	.00000	100.00	33.333
Ratio of idle AGV	55.417	.62635	.00000	100.00	33.333
ws1 in buff util.	90.033	.27543	.00000	100.00	100.00
ws1 util.	98.437	.12601	.00000	100.00	100.00
ws1 out buff util.	13.541	1.7793	.00000	100.00	.00000
ws2 in buff util.	5.4852	2.8763	.00000	100.00	.00000
ws2 util.	23.953	1.7818	.00000	100.00	100.00
ws2 out buff util.	3.0012	3.9573	.00000	50.000	.00000
ws3 in buff util.	21.861	1.3726	.00000	100.00	50.000
ws3 util.	61.055	.79866	.00000	100.00	.00000
ws3 out buff util.	10.046	2.0115	.00000	100.00	50.000
ws4 in buff util.	17.562	1.4713	.00000	100.00	.00000
ws4 util.	60.421	.80935	.00000	100.00	.00000
ws4 out buff util.	11.737	2.0591	.00000	100.00	50.000
Que. length for AGV	.55471	1.3506	.00000	4.0000	1.0000
AGV utilization	69.667	.28826	.00000	100.00	66.667

COUNTERS

Identifier	count	Limit
NO LOTS OUT OF ST 1	148	Infinite
NO LOTS OUT OF ST 2	41	Infinite
NO LOTS OUT OF ST 3	104	Infinite
NO LOTS OUT OF ST 4	105	Infinite
NO LOTS OUT OF ST 9	143	Infinite
NO LOTS OUT OF ST 10	161	Infinite

Run Time: 0 min(s) 53 sec(s))
Simulation run complete.

[그림 9] 시뮬레이션 프로그램 수행결과

5. 결 론

본 연구에서 개발한 코-드 작성시스템은 다음과 같은 장점이 있다.

첫째, AGVS 설계자가 시뮬레이션 목표언어에 대한 지식이 전혀 없어도 본 시스템을 이용하여 자기가 설계한 AGVS 설계대안의 성능을 평가해 볼 수 있다.

둘째, AGVS 운영소프트웨어를 개발하는 연구자의 경우 본 시스템에서 제공되지 않은 운영논리를 테스트하려 할 때 범용언어로 프로그래밍하여 쉽게 연결할 수 있게 되어 있다.

셋째, AGVS 설계나 운영논리에 있어서 새로운 선택사양을 추가로 제공하고자 하는 경우 기존의 프로그램을 변경시키지 않고 모델프로그램 베이스와 상관관계 데이타베이스에 그 데이타를 추가시킴으로써 기능을 쉽게 확장할 수 있다.

향후 이 연구에 추가하여 보완 또는 수행되어야 할 연구내용으로는 다음과 같은 것들이 있다.

우선 본 시스템은 프로그램 베이스의 내용을 더욱 풍부하게 하여 더욱 다양한 시스템에 적용가능하게 범위를 확장하는 것이 필요하다. 예를 들어, 시스템에 운반을 위해 들어오는 장소가 본 시스템은 한 개뿐으로 제약적인데 두 개 이상이 된다든가, 어떤 작업을 하기 위한 이동이 아니고 창고출납과 같은 단순한 운반만을 목적으로 하는 등 다양한 시스템의 흐름형태에도 적용가능하게 하는 적용분야를 확장하기 위한 연구가 수행되어야 한다.

그 다음으로, 에니메이션 기능 추가나 결과해석을 위한 전문가 시스템 개발, 최적화 모듈개발 등 AGVS 설계작업 지원을 위한 통합 소프트웨어 환경구축의 작업이 필요하다.

참 고 문 헌

[1] Davis, D. A., "Comparison of In-transit Routing Policies for Automated Guided Vehicle Systems", *Ph.D. Dissertation*, The Pennsylvania State University, May, 1988.

[2] Davis, D. A., "Modeling AGV Systems", *Proceedings of the 1986 Simulation Conference*, pp.568-574, 1986.

[3] Egbelu, P. J., "A Design Methodology for Operational Control Elements for Automated Guided Vehicle Based Material Handling Systems", *Ph.D. Dissertation*, Virginia Polytechnic Institute and State University, 1982.

[4] Gaskin, R. J., J. M. A. Tanchoco, "AGVSim2-A Development Tool for AGVS Controller Design", *International Journal of Production Research*, Vol.127, No.6, pp.915-926, 1989.

[5] Goetschalchx, M., L. F. McGinnis, "Engineering Workstation is Design Tool for Computer-Aided Engineering of Material Flow Systems", *IE*, *June*, pp.34-38, 1989.

[6] Gong, D. C., L. F. McGinnis, "An AGVS Simulation Code Generator for Manufacturing Application", *Working paper*, Georgia Tech Research Corporation, 1990.

[7] Haddock, J., "An Expert System Framework Based on a Simulation Generator", *SIMULATION*, Vol.48, No.2, pp.45-53, 1987.

[8] Lin, J. T., "Development of a Graphic

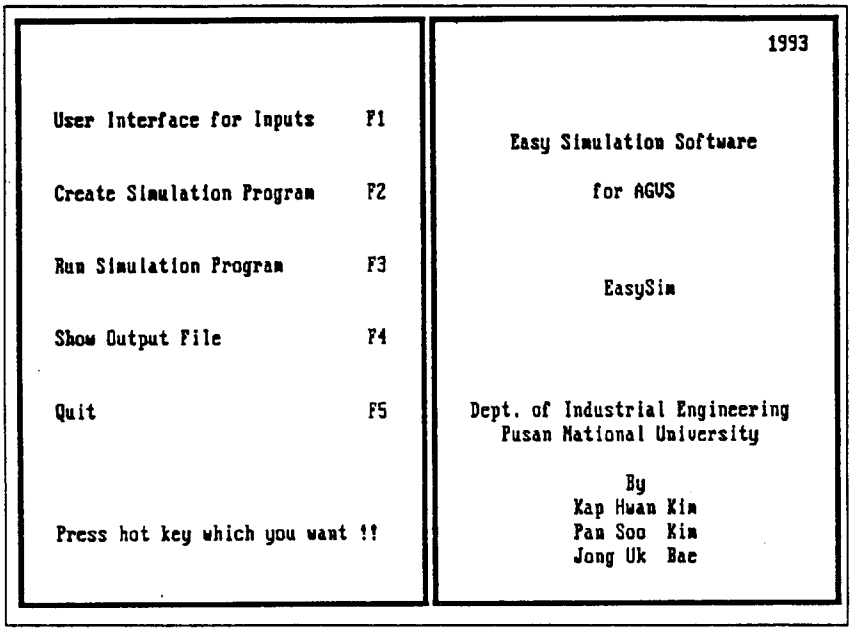
Simulation Model for Design of Automated Vehicle Systems", *Ph.D. Dissertation*, Lehigh University, 1986.

[9] Mathewson, S. C. "The Application of Program Generator Software and Its Extensions to Discrete Event Simulation Modeling", *IIE Transaction*, Vol.16, No. 1, pp.3-18, 1984.

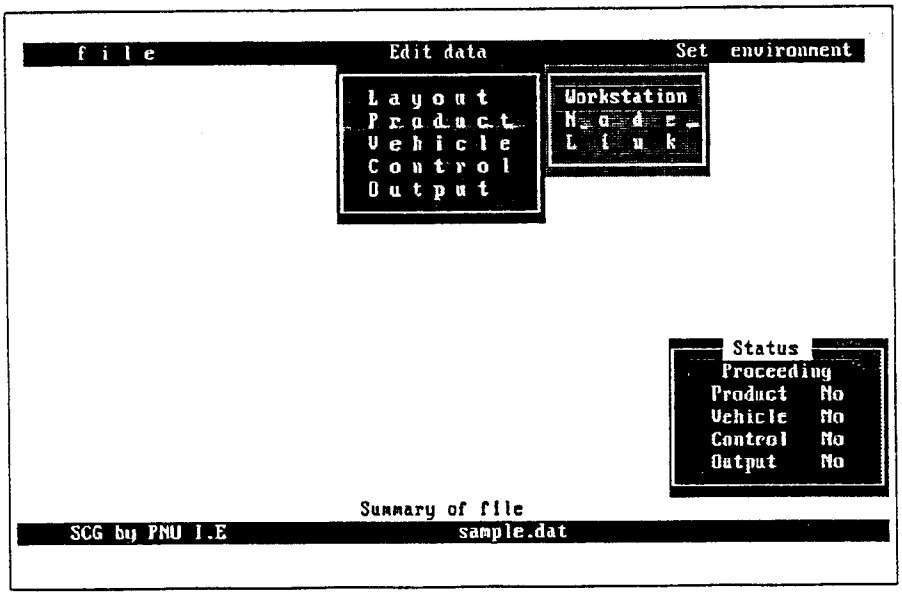
[10] Mathewson, S. C., "Simulation Program Generators : Code and Animation on a P. C.", *Journal of the Operational Research Society*, Vol.36, No.7, pp.583-586, 1985.

[11] Pegden, C. D., R. E. Shannon, R. P. Sadowski, Introduction to Simulation Using SIMAN, McGraw-Hill, 1990.

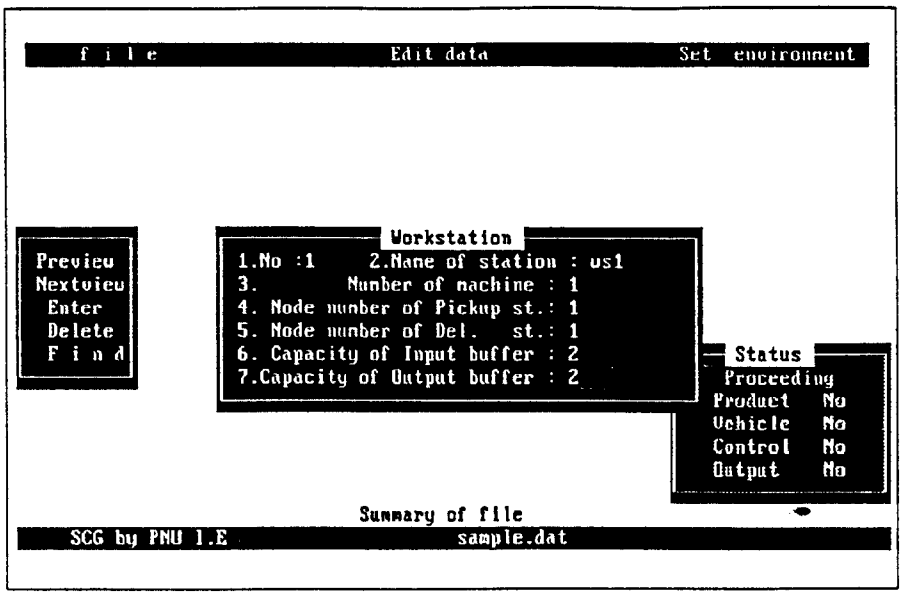
[부록] 코드 작성시스템의 중요화면



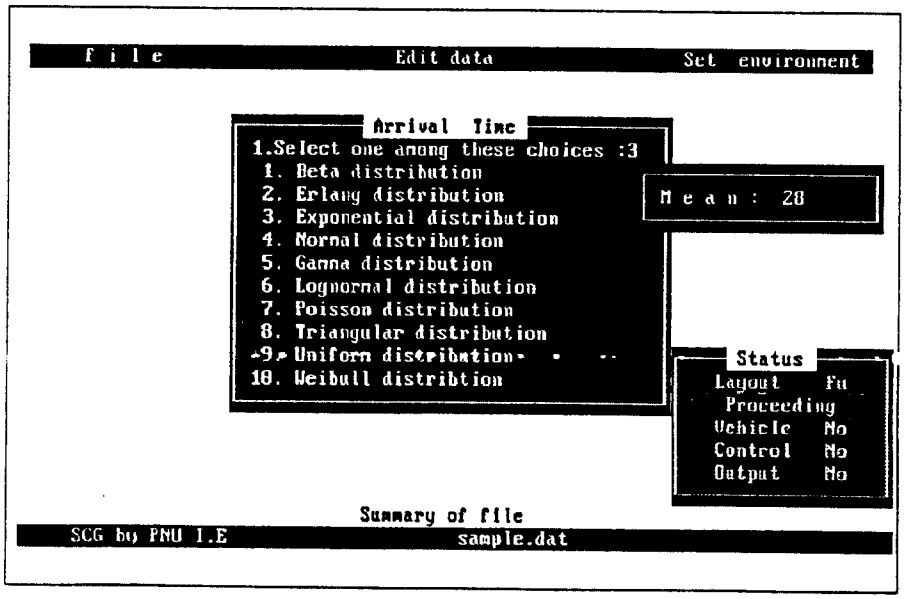
[그림 10] 본 시스템의 초기화면



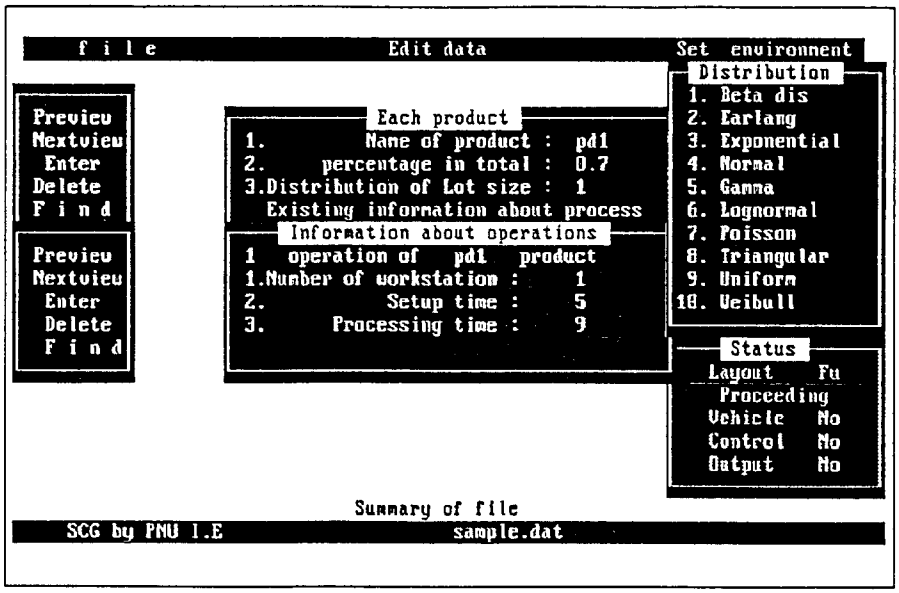
[그림 11] 배치도에 관한 정보 입력초기화면



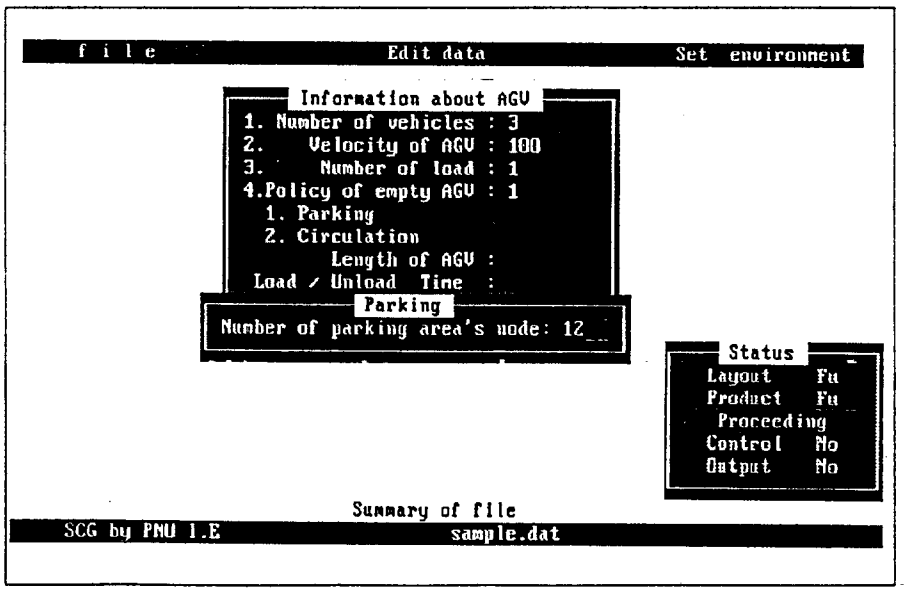
[그림 12] 작업장에 관한 정보 입력화면



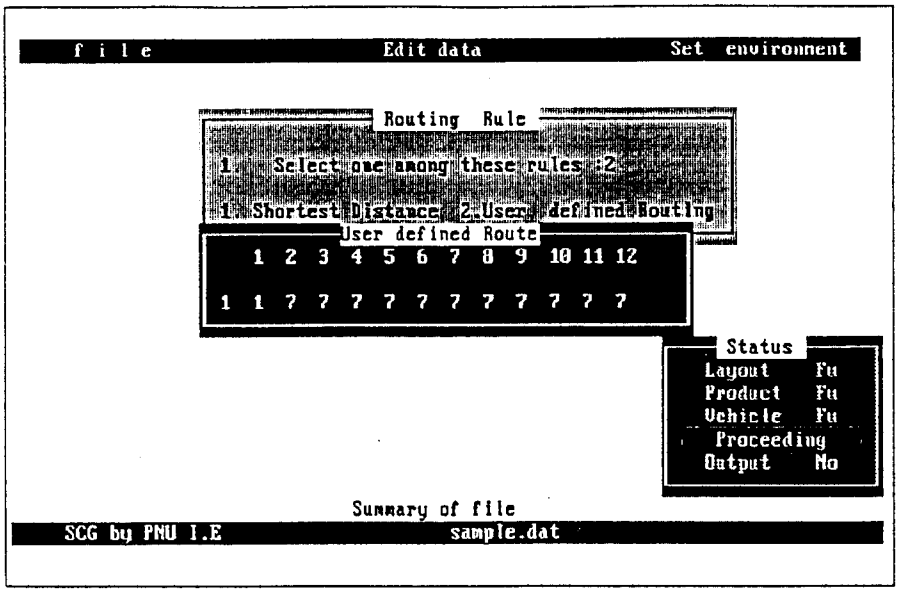
[그림 13] 가공할 부품의 도착시간 간격분포 입력화면



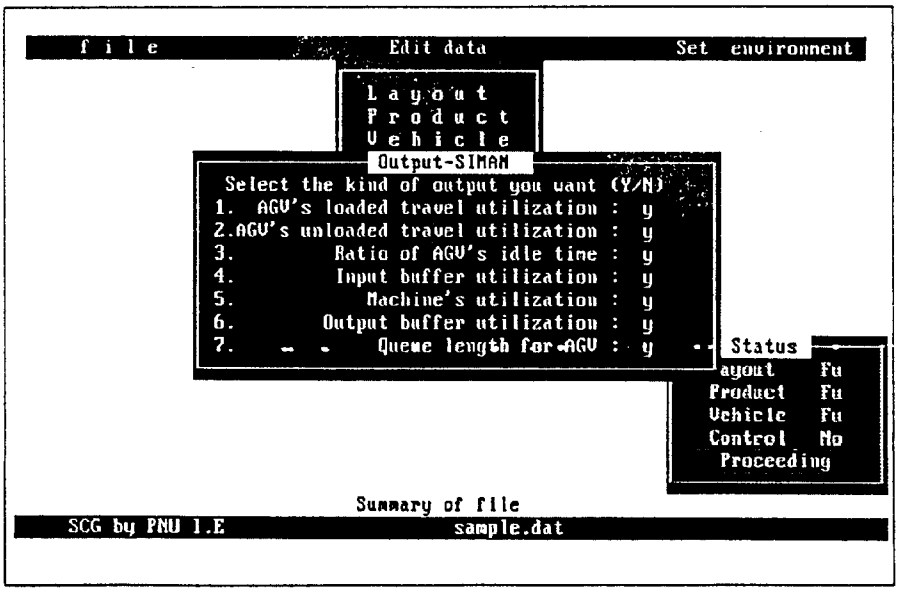
[그림 14] 부품의 각 공정에 관련된 정보 입력화면



[그림 15] 차량에 관한 정보 입력화면



[그림 16] 운행경로 결정규칙중 사용자정의경로 입력화면



[그림 17] 원하는 결과 선택화면