JRC RESEARCH
REPORT NO. 90-01

Automatic Programming of Simulation Models

Task 3

Final Report
D.O. 34 Task 3

Bernard J. Schroer
Fan T. Tseng
Shou X. Zhang
Wen S. Dwan
Johnson Research Center
University of Alabama in Huntsville
Huntsville, AL 35899
(205)895-6361

NASA/MSFC contract NAS8-36955
Delivery Order 34

January 1990

TABLE OF CONTENTS

ABSTRACT

This report contains the research results from 1988 research grant NAG8-641 from NASA/MSFC and a follow on 1989 contract NAS8-36955. Therefore, some of the results in this report were documented in the final report, Automatic Programming of Simulation Models, UAH Report 725, September 1988.

1.0 INTRODUCTION

The concepts of software engineering offer an approach to minimizing software development problems and to improving the overall simulation modeling environment. Software engineering encompasses the entire life cycle process by which a program is conceptualized, structured, programmed, verified, validated, and maintained. The goal of software engineering is to develop quality code, on time, and within budget. To meet this goal requires a variety of programming tools such as a good language with a library of reusable modules, a flexible editor, and a potent debugger.

The focus of this research project is on using the concepts of software engineering to improve the simulation modeling environment. Of special interest is to apply an element of rapid prototyping, or automatic programming, to assist the modeler define the problem specification. Then, once the problem specification has been defined, an automatic code generator is used to write the simulation code.

The following two domains were selected for evaluating the concepts of software engineering for discrete event simulation: manufacturing domain and a spacecraft countdown network sequence.

The specific tasks for this follow-on contract were to:

1. Define the software requirements for a graphical user interface to the Automatic Manufacturing Programming System (AMPS) system.
2. Develop a graphical user interface for AMPS.
3. Compare the AMPS graphical interface with the AMPS interactive user interface.

2.0 MODELING LIFE CYCLE

There has been considerable interest in improving the process for developing simulation models. One area of interest is the development of simulation support environments. Henriksen (1983) suggests a simulation software development environment composed of a set of integrated software tools. Standridge (1983) proposes the integration of software tools and database management techniques on each stage of the simulation model development process. Pidd (1984) also outlines a simulation support environment concept for handling one simulation problem at a time.

Overstreet and Nance (1985) emphasize the need of a specification language to assist in analysis of discrete event simulation models. Balci (1986) describes the requirements for general model development environments with focus on discrete event simulation modeling. Balci and Nance (1987) report a simulation support system for prototyping the automation-based paradigm. Rozenblit and Ziegler (1985) set up a conceptual framework for constructing knowledge-based, computer-aided environment for hierarchical, modular discrete-event modeling.

More recently, the Semiconductor Manufacturing Technology Initiative (SEMATECH) is developing a coherent modeling environment called CHIPS (Coherent Intergrated Planning System). CHIPS consists of five major modules: process flow analysis module, queueing network analysis module, system simulation module language, and a cost analysis module (Phillips, et al 1989). SEMATECH is a cooperative project between industry and government with the goal to recover the world leadership in semiconductor manufacturing.

Figure 1 outlines the phases of the model life cycle (Balci 1986 and Nance 1988). Basically, the modeling process is iterative rather than sequential as indicated in Figure 1. That is, the modeler goes back and forth between the various phases during the modeling process.

Figure 1 can be considered as the traditional approach to simulation modeling (Balci 1986 and Nance et al 1988). The same process also applies to general modeling problems. The process consists of six stages described on the left side of the figure. On the right side, different types of models generated at different stages through the process are listed. For example, a conceptual model of a manufacturing system may be the understanding of the system by the modeler and in the mind of the modeler. A communicative model of the manufacturing system may be a graphic representation of the system in the form of a block diagram, flowchart, or network diagram. A GPSS simulation program of the manufacturing system is a programmed model. The model results are generated by executing the program.

3.0 SOFTWARE ENGINEERING IN THE MODELING LIFE CYCLE

Rapid prototyping is a technique used in software engineering for capturing system requirements early in the modeling life cycle so that these requirements can be evaluated, tested, verified and validated early in the process before starting the actual coding. The end result of rapid prototyping is the potential for large increases in productivity.

An element of rapid prototyping is the automatic conversion of the communicative model into executable code. Automatic Programming (AP) is defined as the automation of some aspects of the computer programming process (Barr 1982). This automation is accomplished by developing another

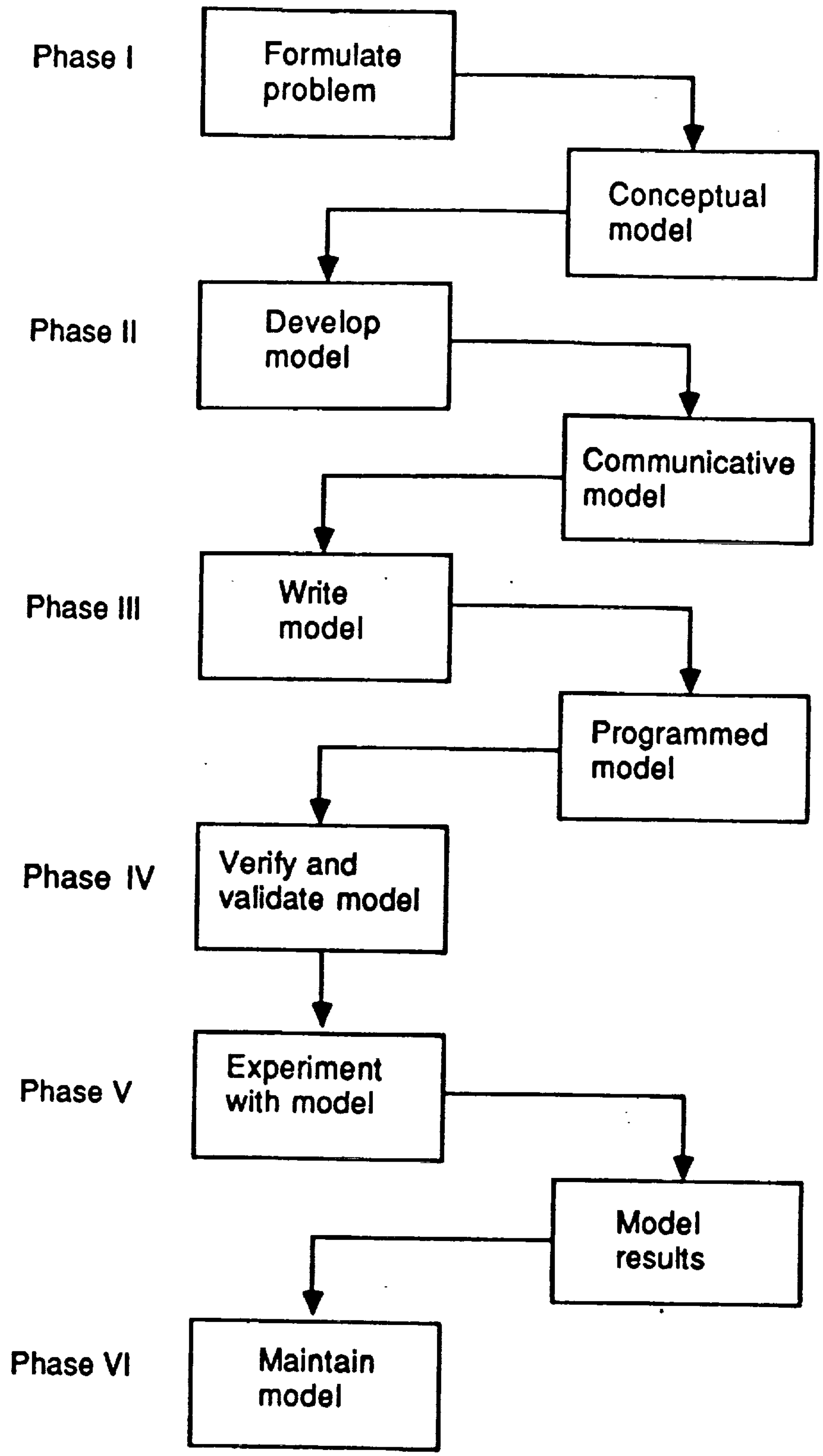


Figure 1. Phases in the modeling life cycle

program, an automation programming system, that raises the level of satisfying computer program instructions. In other words, an AP system helps programmers write programs. AP systems improve the overall environment for defining and writing programs (Brazier and Shannon 1987). Consequently, there should be a reduction in the amount of detail that the programmer needs to know.

To write simulation programs automatically, two phases in the simulation modeling process are usually automated. The first phase is the automation of the process of specifying the problem. The second phase, and the more difficult phase, is the automatic generation of executable code in the target simulation language.

3.1 Problem Specification

Figure 2 shows the overlaying of automatic programming onto the modeling life cycle in Figure 1. Phase II, model development, has been replaced by a user interface program that assists the modeler in defining the problem specification.

The automatic problem specification can be considered as an intelligent assistant to the user in defining the simulation model. Some authors call this approach the specification acquisition element of the simulation model construction (Murray and Sheppard 1988).

Three approaches for assisting the user in defining the simulation model or problem specification are:

°Natural language interface

°Interactive graphical interface

°Interactive dialogue interface

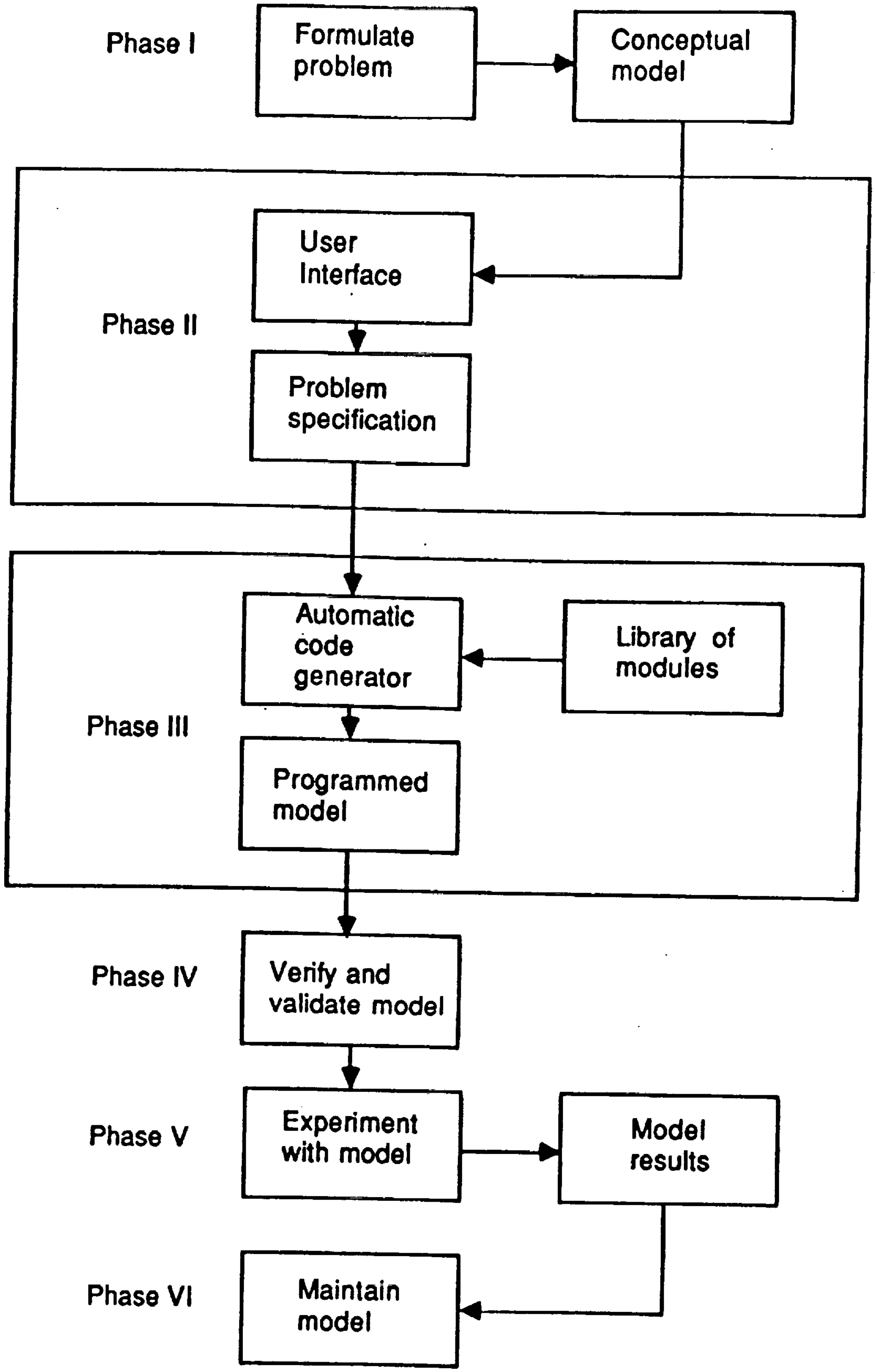


Figure 2. Software engineering imbedded in the modeling life cycle

There is a fourth approach for assisting in the definition of problem specification, which is the use of a high-level specification language. This approach is less domain specific. However, the use of a high-level specification language requires the user to learn another language in order to define a problem.

3.1.1 Natural language interface

The Natural Language Interface (NLI) allows the user to specify the problem in free text format to the computer via a keyboard. The NLI then attempts to parse the text and automatically generate the simulation code in the target language. Most NLI's communicate interactively with the user to identify missing information and possible inconsistencies. The Natural Language Programming for Queueing Simulations (NLPQ) (Heidorn 1974) and the Electronic Manufacturing Simulation System (EMSS) (Ford and Schroer 1987) are two examples of a NLI.

3.1.2 Interactive graphical interface

The second approach to assist the user in specifying the problem is an Interactive Graphical Interface (IGI), which is less difficult than the NLI. An IGI consists of a menu of icons that are mouse selectable by the user in constructing a graphical representation of the system being simulated. Once the system has been constructed, the user inputs the attributes corresponding to the icons through the keyboard.

An example of an IGI is by Khosnevis and Chen (1986) who developed an object-oriented approach for graphically modeling a system. This system is rule-based and written in common LISP on an IBM PC. Once the graphical description of the model is completed, the system automatically generates the equivalent SLAM simulation code.

3.1.3 Interactive dialogue interface

The third approach to assist the user in defining the problem specification is the Interactive Dialogue Interface (IDI). An IDI consists of a series of questions that are asked the user. Among the three approaches for defining the problem specification, the interactive dialogue interface is the one most commonly used by developers.

Several systems have been developed using the interactive dialogue approach. Haddock and Davis (1985) have developed a Flexible Manufacturing System (FMS) simulation generator. Brazier and Shannon (1987) have developed an automatic programming system for modeling Automated Guided Vehicle System (AGVS). Murray and Sheppard (1988) have developed a Knowledge Based Model Construction (KBMC) system to automate model definition and code generation. These last three systems generate SIMAN code.

3.2 Automatic simulation code generation

In Figure 1, Phase III, write model, has been replaced in Figure 2 by an automatic code generation program. Basically, two approaches exist for taking the internal problem specification and then automatically generating executable code in the target simulation language. The first approach is to generate simulation code directly from the internal representation of the problem specification.

A second approach is to use a library of predefined macros to assist in the automatic generation of the simulation code. The advantage of such an approach is the ability to solve more specialized problems than those previously discussed in the literature. The disadvantage is that most macros are domain specific. As a result, additional macros are needed to solve another problem domain.

4.0 AUTOMATIC MANUFACTURING PROGRAMMING SYSTEM (AMPS)

4.1 Introduction

The Automatic Manufacturing Programming System (AMPS) is a software engineering tool for rapidly prototyping selected phases of the simulation process for domain specific manufacturing systems. The AMPS system consists of the following elements:

- °A set of generic manufacturing modules written in GPSS/PC (Minuteman 1986)
- °An interface program for extracting the problem from the user and for creating a problem specification file
- °An automatic code generator program for creating the code in the target simulation language GPSS/PC

The AMPS system domain is those manufacturing systems that can be described as having:

- °Assembly and subassembly lines where parts are being added to an assembly.
- °Manufacturing cells that are providing parts to the assembly and subassembly lines.
- °Inventory of parts being moved between the manufacturing cells and subassembly lines.

4.2 AMPS System Overview

Figure 3 is an overview of the AMPS system operation. Once the user has scoped the problem domain, the user sits at a workstation and responds to the questions from the interface program. Based on the responses, the interface program creates an internal problem specification file. This file includes the manufacturing process network flow and the attributes for all stations, cells and stock points. The problem specification file is then used as input to the automatic code generator program which generates the simulation program in the target language GPSS/PC.

The user then adds the experimental frame, such as the run statements, and the GPSS/PC simulation program is executed. To change the GPSS/PC model, the user recalls the problem specification. The user interface then provides the simulationist with a number of options to change or modify the problem specification. The code generator will then rewrite the GPSS program.

4.3 Library of GPSS Macros

In analyzing most manufacturing systems at the macro level, the following function are generally similar in nature:

- Assembly - adding part X to part Y resulting in part Z
- Fabrication - making of part X from part Y
- Inspection - inspecting part X
- Inventory transfer - moving part X or a cart of part X from stock point A to stock point B
- Simple operation - performing an operation on part X resulting in a modified part X

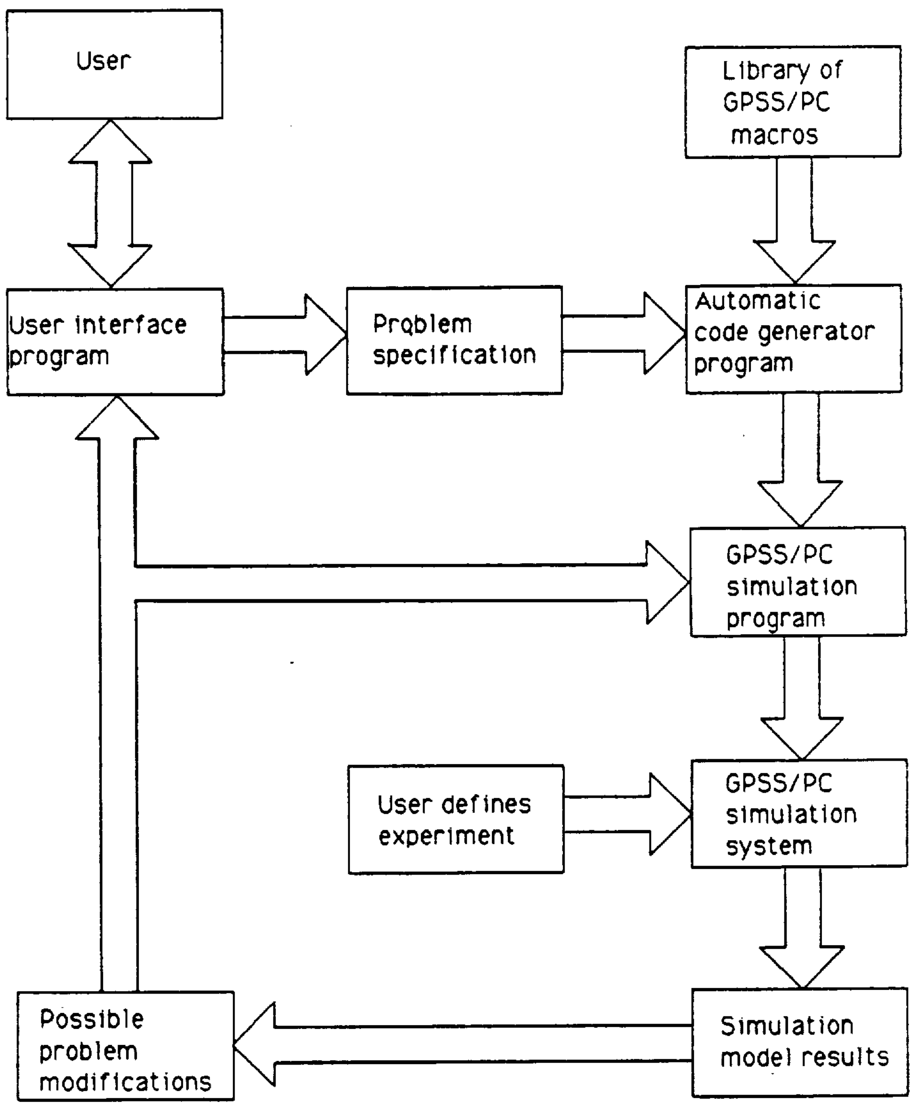


Figure 3. AMPS system overview

These five functions represent the current domain of manufacturing functions within the AMPS system. Once the manufacturing functions have been defined, GPSS subroutines are written for the functions (see Appendix A). These routines constitute a library of predefined GPSS subroutines, or macros. This library of macros is then called, when needed, in the construction of the GPSS simulation model. Currently, the AMPS system has the following five GPSS subroutines:

° Assembly

° Manufacturing

° Inventory transfer

° Inspection

° Task

In a recent article on SEMATECH (Phillips, et al 1989), researchers have identified ten machine modules for the semiconductor manufacturing domain. Furthermore, the SEMATECH group has indicated that possibly no more than 16-20 generic machine modules may be required to completely represent the semiconductor manufacturing environment.

Figure 4 briefly describes each of these macros. For example, the assembly station macro has the capability of simulating the adding of a variety of different items to the incoming part resulting in a modified part that is then transferred to the next destination, a station or stock point. For example, in Figure 4, station STA1 may assemble two part C's and three part D's to the incoming part A resulting in Part B.

The manufacturing cell makes a cart of specified parts when an order is received. The cell can make multiple part types. For example, in Figure 4, cell MC1 may make one part A from two part C's and three part D's

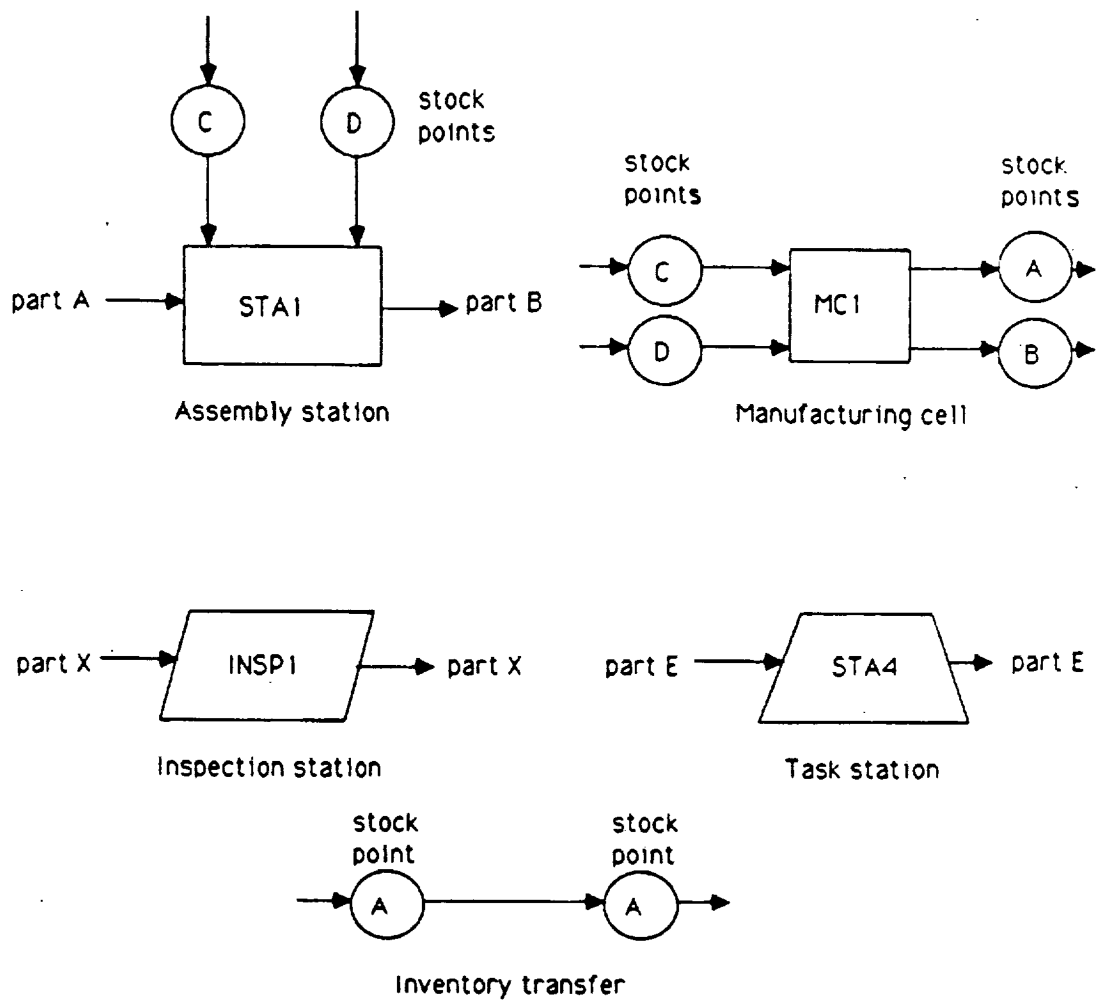


Figure 4. GPSS manufacturing macros

and one part B from one part D. The task station performs an operation on a part. For example, in Figure 4 an operation is performed at station STA4 on part E resulting in a modified part E. The inspection station inspects a defined percentage of parts. Of those inspected, a defined percentage is defective. Of those defective, a defined percentage is scrapped.

The inventory transfer macro grants part requests from an assembly station or a manufacturing cell and checks if the inventory system is a push or pull. For a pull system the macro orders a cart of parts by sending an empty cart back to the source and sends a full cart of parts to the demand stock point from the source stock point.

4.4 Sample Problem

Figure 5 is an example of a typical manufacturing system that can be modeled by the AMPS system. The manufacturing system consists of one assembly line, two subassembly lines, and two manufacturing cells. The assembly line consists of two assembly stations, one task station and one inspection station. Subassembly line 2 consists of one assembly station and one task station while line 3 consists of two assembly stations. Manufacturing cell MC1 provides part type C for assembly station ASSY1 and part type H for assembly station ASSY8. Manufacturing cell MC2 provides part type E for assembly station ASSY5 and part types F and G for assembly station ASSY7. There are a variety of stock points, labeled A through L, located throughout the manufacturing system.

The GPSS program for the manufacturing system in Figure 5 that was generated by AMPS consists of 344 blocks, of which 110 blocks are for the five macros, 25 blocks are for the main program and 209 blocks are matrix savevalues for defining the system attributes.

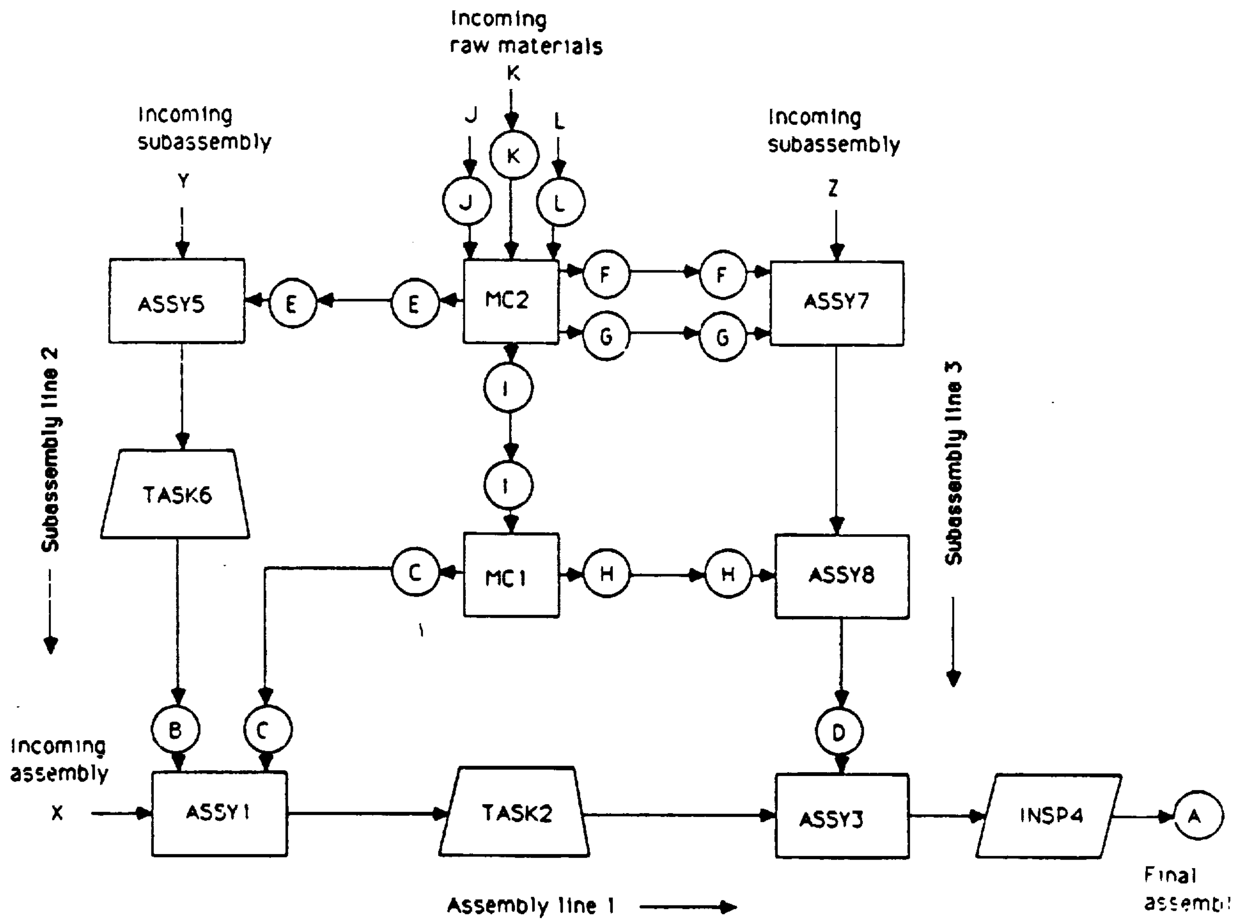


Figure 5. Manufacturing system

5.0 AUTOMATIC NETWORK PROGRAMMING SYSTEM (ANPS)

5.1 Introduction

Large simulation projects have been undertaken for the space program. One of the projects involve simulating the countdown sequence prior to spacecraft liftoff. A countdown has a number of constraints. For example, on a lunar mission, these constraints may include allowable launch azimuth, required earth orbit inclination, daylight at the lunar landing area, and daylight at the primary recovery area. As a result of these constraints, a launch window of only several hours could exist during three consecutive days in a month.

Another constraint is the cryogenic propellents. The handling of the cryogenic propellents prevent a launch hold from one day to the next. For example, a launch that is scrubbed after the cryogenics have been loaded is generally delayed at least until the third day within the launch opportunity. In addition, a typical prelaunch consists of thousands of events, both on the launch vehicle, as well as the ground support equipment, that must be successfully completed to launch within a given launch window.

The Automatic Network Programming System (ANPS) is a tool to assist the modeler of prelaunch countdown sequences define the problem, and to then automatically generate the program code in the target simulation language GPSS/PC. The domain of problems that can be solved by ANPS is the prelaunch activities of space vehicles and the operation of supporting ground support equipment. A broader domain is reliability network models of hardware systems and subsystems.

5.2 Previous Research

Snyder et al. (1967) have developed a simulation model of the Saturn V prelaunch activities beginning at T-24 hours and continuing through T-0 hours, or lift-off. This model was used to predict the probability of launching the spacecraft within a given launch window. A second objective of the model was to identify locations in the countdown for placing holds and to determine the length of these holds. The model consisted of over 1100 vehicle subsystems and 400 ground support subsystems. A detailed time line was developed showing the interrelationships of these subsystems. In additions to the time line, the model input included operational data, reliability data, and maintenance data. The model was written in GPSS-II and ran on an IBM 360 computer.

The Synder model was expanded to include multiple launch windows and the operational sequence when a launch window was missed and the spacecraft had to be recycled to the next launch window (Schroer 1969). The model was used to predict the probability of launching a spacecraft within a given set of back-to-back launch windows. A second objective was to predict the probability of launching in a subsequent window, given a window had been missed and a recycle sequence and a possible hold had to be executed before resuming the countdown.

5.3 ANPS System Overview

The three AP elements in ANPS are an Interactive Dialogue Interface (IDI), a library of software modules, and an automatic simulation code generator.

The actual operation of ANPS is similar to AMPS (see Figure 3). The ANPS system uses an interactive dialogue interface to assist the user define the problem specification. Using this interface, the user sits at a personal computer and enters into a dialogue with the ANPS system. Based on the user's responses, the interactive interface creates an internal problem specification file. This file includes the time line for the countdown sequence, the attributes for the activities, and the dependent relationships between the activities. The specification file is used by the code generator program to create the simulation program in the target simulation language GPSS/PC.

5.4 Library of GPSS Macros

Since the ANPS system is domain specific to prelaunch countdown sequences, the number of needed software modules is minimal. Consequently, ANPS consists of the following four GPSS modules (see Appendix B):

- ° Fixed activity operation function (VENT _A)
- ° Continuous activity operation function (VENT _B)
- ° Activity failure function (FAIL)
- ° Activity interrupt function (XACT _ DELAY)

These modules were selected based on a detailed evaluation of the two previously discussed models by Synder (1967) and Schroer (1969). Interestingly, several of these previously developed modules were written as Fortran HELP routines using the old GPSS-II.

The fixed activity operation function (VENT_A) simulates the operation of each fixed time activity and its time to failure. If the activity fails during its operation, the transaction is fowarded to the activity failure function (FAIL).

The continuous activity operation function (VENT_B) simulates the operation of each continuous time activity and its time to failure. This activity is not completed until all other related activities are completed. For example, system power is a continuous time function that will be on until all activities requiring power are completed. If the activity fails, the transaction is fowarded to the activity failure function. (FAIL).

The activity failure function (FAIL) simulates the failure of an activity as indicated by functions VENT _ A and VENT _ B. When an activity fails, all the dependent activities enter a hold state. The function then simulates the time to repair the activity. If another activity fails during the delay of a dependent activity and the dependent activity is dependent on the first failed activity, the additional time to repair, if any, is added to the delay of the dependent activity. The failure function assumes that a dependent activity that has been delayed cannot fail during the delay. The activity interrrupt function XACT _DELAY contains the logic to add any additional time to an activity on hold if another activity fails during the hold and the held activity is dependent on the failed activity.

The ANPS macros impose the following constraints:

- An activity failure will cause that activity to be delayed until the failure has been repaired.
- All dependent activities will also be delayed for the same time until the failure has been repaired.
- If another activity fails during the delay of a dependent activity and the dependent activity is also dependent on the just failed activity, the additional time to repair, if any, is added to the delay of the dependent activity.

- A dependent activity that has been delayed cannot fail during the delay time and will not cause other dependent activities to be delayed.
- No two continuous activities can end on the same node.
- No two activities can start from the same node and terminate on the same node.
- No two activities can start from the same node and terminate on the same node.

5.5 Sample Problem

Figure 6 is a time line for a simplified prelaunch countdown sequence consisting of 16 fixed activities and two continuous activities. Figure 7 is the time line redrawn in the form of a network diagram and structured for input to the ANPS system. The dotted lines in these figures indicate time line constraints. For example, activities ACT11 and ACT15 must be completed before starting activity ACT12. ACT21 is a dummy activity with zero time that is used to impose the activity ACT15 constraint.

Several other dummy activities were also required to construct the network diagram. For example, dummy activity ACT23 was added to simulate the termination of the second continuous activity ACT2, since no more than one continuous activity can end at a node. Also, dummy activity ACT19 was added at the completion of activity ACT5 since no two activities can start from the same node, node 2, and end at the same node, node 4.

Table I contains the time attributes for the activites in the prelaunch countdown. These attributes include activity duration, activity time to failures, and activity time to repairs. Note that activities ACT1 and

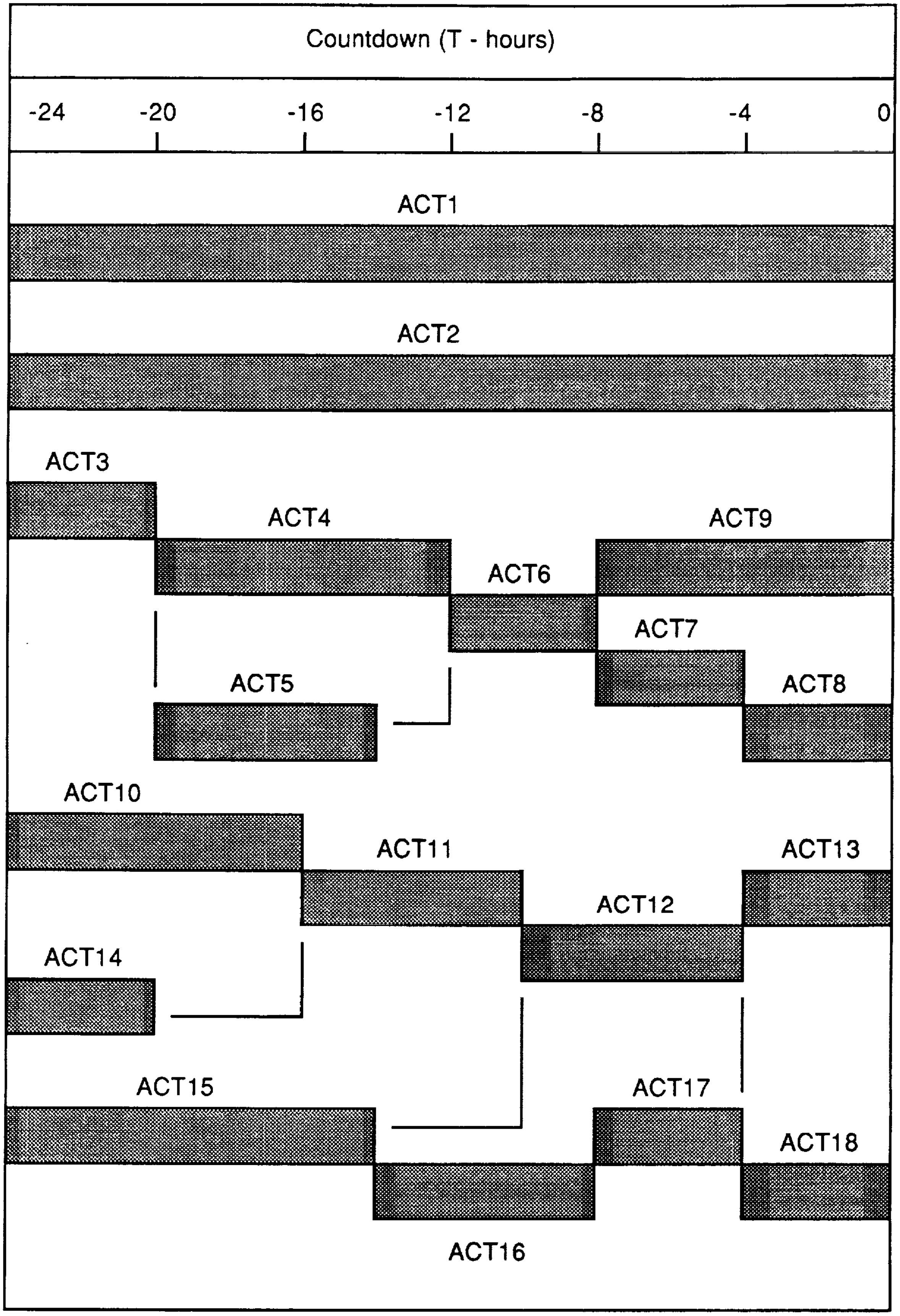


Figure 6. Prelaunch countdown for sample problem

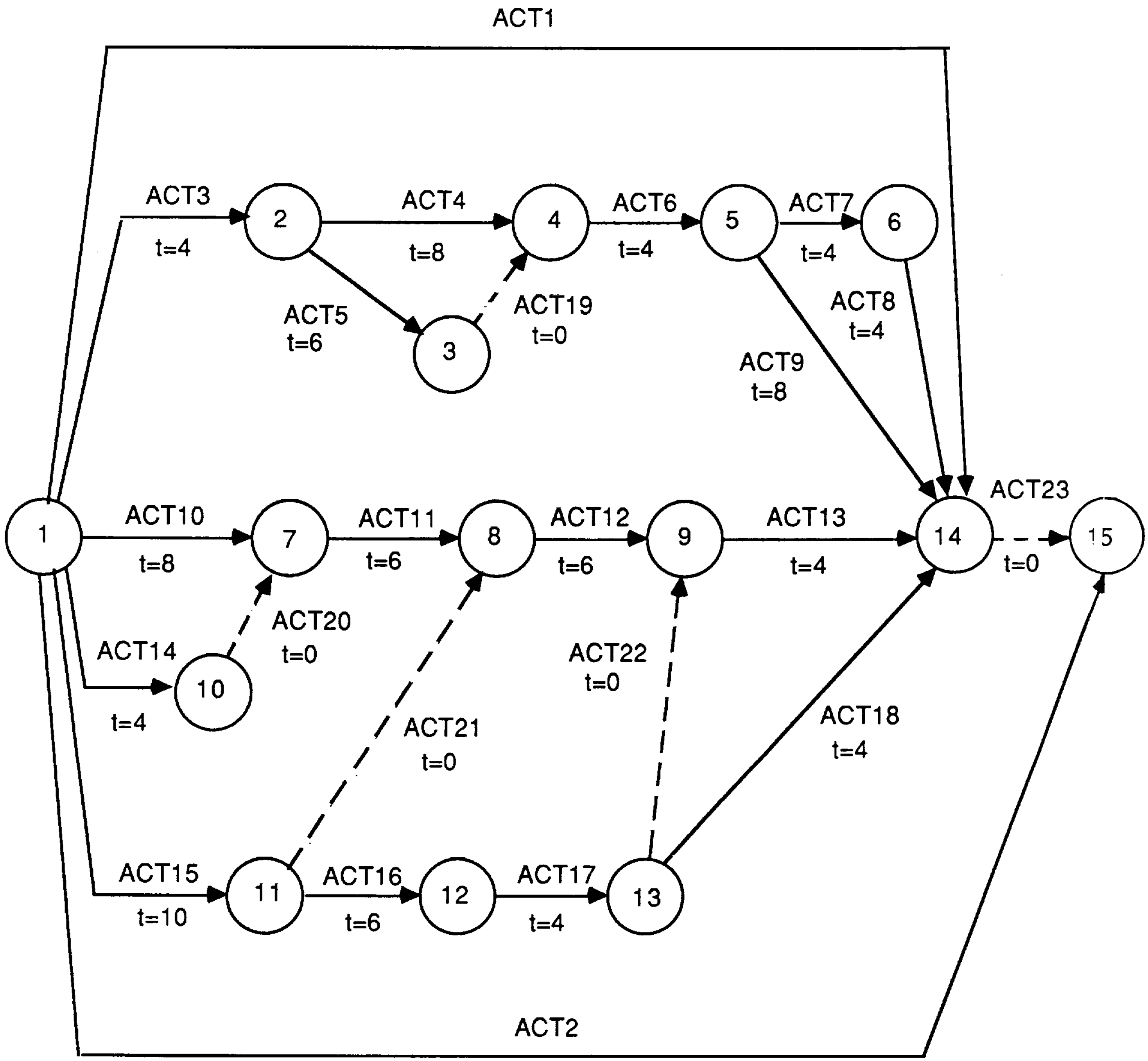


Figure 7. Network representation of prelaunch sequence

Table I. Countdown sequence attributes

Activity	Duration (hours)	Failure time (hours)	Repair time (minutes)
1	Continuous	E(33)	N(60,6)
2	Continuous	E(33)	N(60,6)
3	4	E(12)	N(30,3)
4	8	E(12)	N(30,3)
5	6	E(12)	N(60,6)
6	4	E(12)	N(45,5)
7	4	E(12)	N(45,5)
8	4	E(12)	N(60,6)
9	8	E(12)	N(60,6)
10	8	E(12)	N(60,6)
11	6	E(12)	N(45,5)
12	6	E(12)	N(30,3)
13	4	E(12)	N(60,6)
14	4	E(12)	N(90,9)
15	10	E(12)	N(60,6)
16	6	E(12)	N(120,2)
17	4	E(12)	N(60,6)
18	4	E(12)	N(45,5)
19	Dummy	-	-
20	Dummy	-	-
21	Dummy	-	-
22	Dummy	-	-
23	Dummy	-	-

Table II. Operational dependencies between activities

	Dependent Activity																						
Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	x		x	x	x							x	x					x					
2		x					x	x	x					x	x								
3			x																				
4				x	x																		
5					x																		
6						x																	
7							x																
8								x					x										
9									x														
10										x													
11											x												
12						x						x											
13													x										
14														x									
15															x								
16						x					x					x							
17																	x						
18													x					x					
19																			x				
20																				x			
21																					x		
22																						x	
23																							x

ACT2 have continuous operation times. That is, these activities will operate during the entire prelaunch countdown. An example of a continuous activity is electrical power that may be needed to operate a numer of activities.

Table II contains the operational dependencies between the activities. In other words, the table gives the effect of an activity failure on other activities in the prelaunch. For example, a failure of the continuous activity ACT1 will cause a stopping of activities ACT3, ACT4, ACT5, ACT12, ACT13, and ACT18. Likewise, a failure of activity ACT4 will cause a stopping of activity ACT5.

Figure 8 gives the distribution of time to complete the prelaunch sequence in Figure 6. This distribution is based on the simulation of 200 launches. The mean time to complete the countdown is 34.2 hours. Launch vehicle availability (LVA) is defined as the probability of launching within a given launch window. The LVA for up to six hour window is given in Figure 9. The LVA for a two hour window is 0.015 and increases to 0.596 for a six hour window.

6.0 DEVELOPMENTAL AP SYSTEMS

Table III contains a comparison of the six platforms that have been developed for the AMPS and ANPS systems. Two programmers were used to develop the systems. Programmer A was Mr. W.S. Dwan who was a graduate student in computer science at the University of Alabama in Huntsville (UAH). Mr. Dwan was experienced in LISP on a Symbolics workstation. Programmer B was Mr. S.X. Zhang who was a visiting scholar at UAH from Northwestern Polytechnical University in Xian, China.

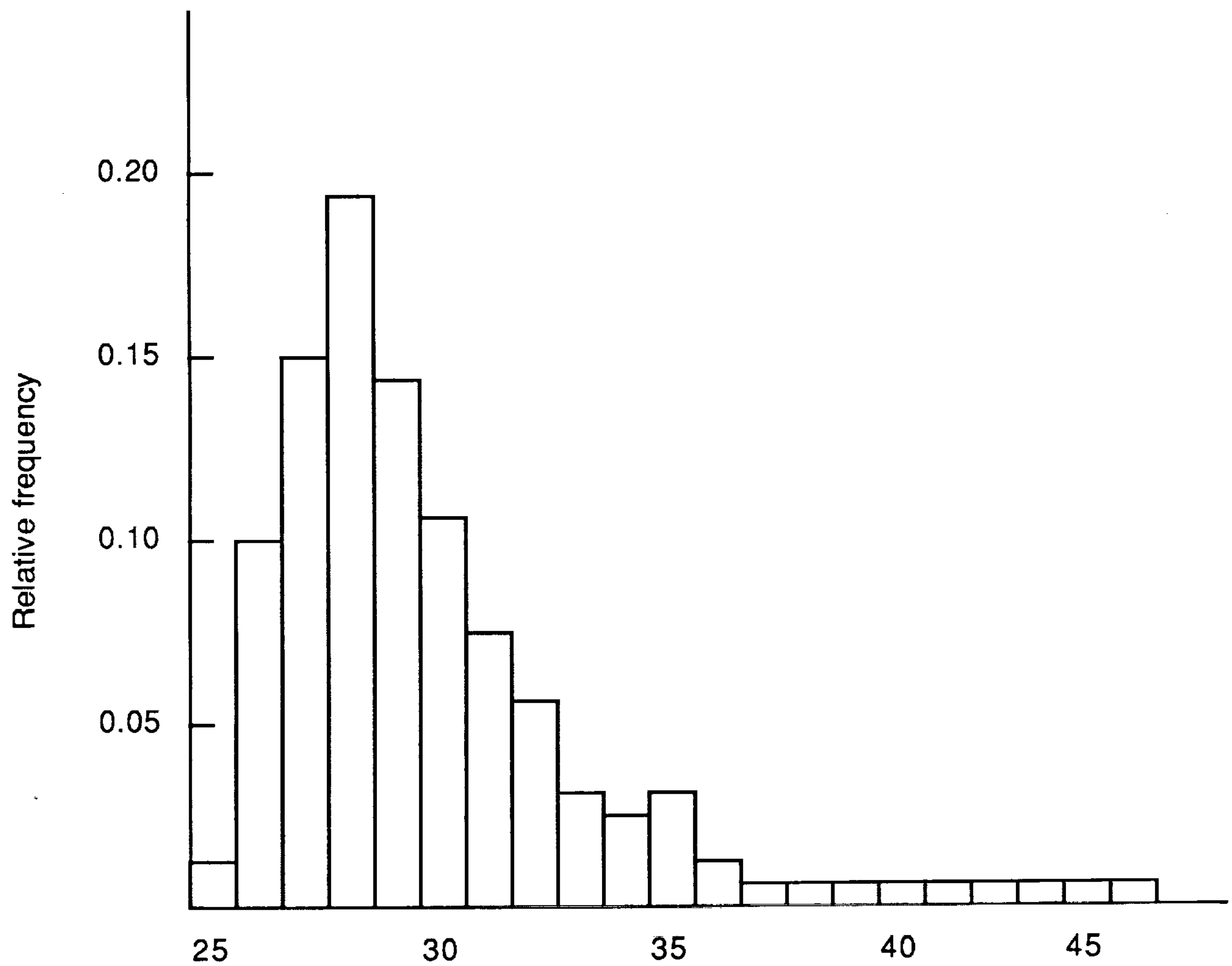


Figure 8. Time to complete prelaunch countdown

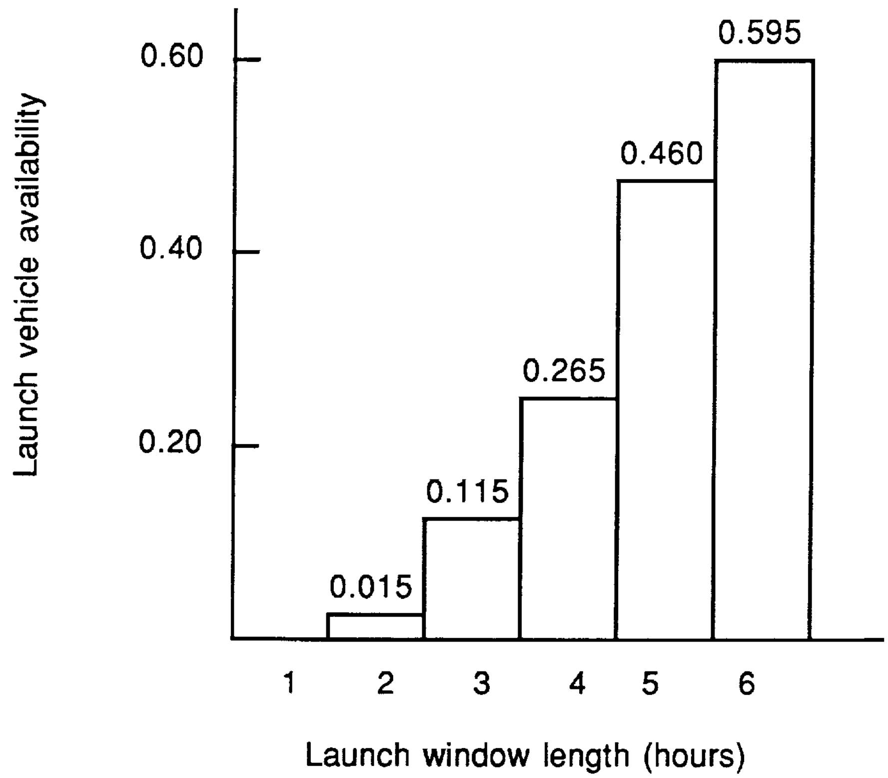


Figure 9. Launch vehicle availability

Table III Comparison of platforms

System	User interface	Order of development	Programmer	Platform	Language	Target language	Man-months	Lines of code	Lines of code per month
AMPS	IDI	1	A	Symbolics	Lisp	GPSS/PC	6	1,500	250
ANPS/PC	IDI	2	B	IBM/PC	Prolog	GPSS/PC	4	1,300	325
AMPS/PC	IDI	3	B	IBM/PC	Pascal	GPSS/PC	3	1,900	633
AMPS/PC	IDI	4	B	IBM/PC	C	GPSS/PC	4	1,300	325
AMPS/Graphics	IGI	5	A	Symbolics	Lisp	GPSS/PC	15	3,500	233
AMPS/PC	IDI	6	B	IBM/PC	C	SIMAN/PC	3	1,600	533

Notes
1. IDI - Interactive Dialogue Interface
2. IGI - Interactive Graphical Interface
3. AMPS - Automatic Manufacturing Programming System
4. ANPS - Automatic Network Programming System

6.1 AMPS/Symbolics

The AMPS system was initially developed for the Symbolics 3620 workstation and used the Interactive Dialogue Interface (IDI). Figure 10 is a portion of a typical IDI dialogue. The AMPS system was written in LISP by programmer A in six man months. The system consisted of 1,500 lines of LISP code. The code production was 250 lines per month.

A detailed description and operation of the AMPS system is given in UAH Report 720, Automated Manufacturing Programming System User's Manual, September 1988. The system has been submitted to NASA COSMIC (reference # 28367). The AMPS/Symbolics system was also ported to the TI Explorer workstation.

6.2 ANPS

ANPS was the second system developed and used with the IDI dialogue. Figure 11 is a portion of a typical IDI dialogue. This system was developed by programmer B using Turbo Prolog on an IBM/PC. The system consisted of 1,300 lines of code. The code production was 325 lines per month.

A detailed description and operation of the ANPS system is given in UAH Report 735, Automatic Network Programming System User's Manual, October 1988. The system has been submitted to NASA COSMIC (reference #26091).

6.3 AMPS/PC

This version of AMP was developed in Turbo Pascal on an IBM/PC and uses an IDI dialogue. The system was developed by programmer B using Turbo Pascal (Borland 1987) on an IBM PC. The system consists of 1,900 lines of code. The code production was 633 lines per month.

```
How many types of final products to be made in the manufacturing system: 2
   Name of the product 1: A
   Name of the product 2: B

Do you want to modify the input above? (Y or N) No.

**************************************
*    Specification of product A
**************************************

Type of the facility used to produce product A at the final stage: Assembly line

Name of the line to produce product A: MAIN
Number of stations in line MAIN: 2
Capacity and initial inventory at the stock points:
   Maximum number of parts at stock point: 2000
   Initial number of parts at stock point: 0

Do you want to modify the input above? (Y or N) No.

*********************************************
*    Description of line MAIN
*********************************************

Input process (Interarrival time of orders):
   Time:
   Distribution: Exponential
    Mean: 100

Do you want to modify the input above? (Y or N) No.

station  1
(1) Station id: 1
(2) Type of station: Assembly station
(3) Station name: ONE
(4) Part required:
    Number of part types required: 2
      Name of part: C
      Number of part: 1
      Name of part: D
      Number of part: 2
(5) Time:
    Distribution: Normal
    Mean: 100
    Standard deviation: 2

Do you want to modify the input above? (Y or N) No.
```

Figure 10. Typical IDI dialogue for AMPS/Symbolics

```
    Name of GPSS program file                 : EXAMPLE1
    Name of GPSS problem specification file: SPEC1
1.  Number of activities             : 7
2.  Activity attributes
    Activity name                    : $ACT1
    Activity type (fixed/variable)   : FIXED
    Duration distribution type       : CONSTANT
             mean time               : 20

    Starting node number             : 1
    Ending   node number             : 5
    MTTF     distribution type       : CONSTANT
             mean time               : 110

    MTTR     distribution type       : CONSTANT
             mean time               : 0

    Number of dependent activities : 0

   Do you want to modify the above input (Y/N): N
```

Figure 11. Typical IDI dialogue for ANPS

A detail description and operation of the system is given in UAH Report 723, Automatic Manufacturing Programming System User's Manual, October 1988. The system has been accepted for publication by NASA COSMIC (reference #28398).

6.4 AMPS/PC

This version of AMP is identical to the AMPS/PC in Section 6.3. The only difference is this version is written in Turbo C (Borland 1988) for the IBM/PC. The system consists of 1,300 lines of code. The code production was 325 lines per month. This system has not been submitted to NASA COSMIC.

6.5 AMPS/Graphics

This version of AMPS was developed for the Symbolics 3620 workstation and uses the Interactive Graphical Interface (IGI). The system was written in LISP by programmer A in fifteen months. The system consists of 3,500 lines of code. The code production was 233 lines per month.

The AMP/Graphics system is documented in UAH Report 788, Automatic Manufacturing Programming/Graphics, August 1989. The system is being submitted to NASA COSMIC. Since the AMPS/Graphics has been developed under the followon contract, a more detailed discussion of the system follows.

6.5.1 AMPS/Graphics Overview

An overview of the AMPS/Graphics system is given in Figure 12. The user sits at a Symbolics 3620 workstation to create or modify the model. The output of the Interactive Graphical Interface (IGI) program is the

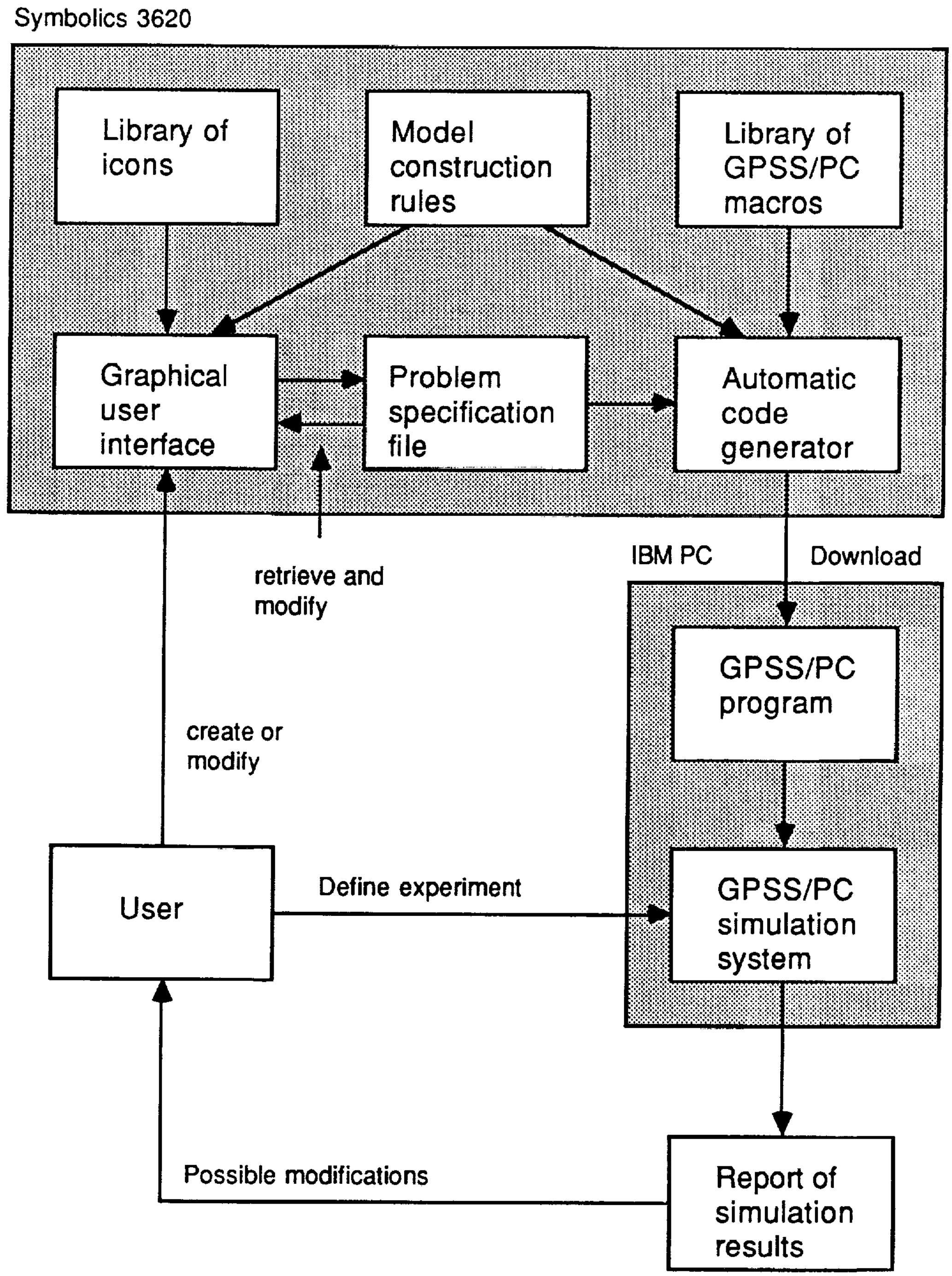


Figure 12. AMPS/Graphics system overview

problem specification file. The automatic code generator program combines the specification file with the selected GPSS/PC macros and writes the simulation program. The program is then downloaded to the IBM PC and executed by the GPSS/PC system. To modify the program, the user recalls the problem specification file and the cycle repeated.

The tree structure of the AMPS commands is given in Figure 13. The system consists of five menus: Main, Model, Layout, Specification and GPSS. In summary, the Main Menu contains the master control commands. The Model Menu contains the commands for creating, editing, saving, and reading models. The Layout Menu contains the commands for constructing the model. The Specification Menu includes the commands for defining the model parameters. The GPSS Menu contains the commands for writing the simulation code.

Figure 14 is a list of the icons available in AMPS. These icons serve as the construction blocks in defining a manufacturing system. To define a manufacturing system the user selects these icons and develops a process flow showing the various stations and the flow between the stations. Figure 15 gives all the feasible connections between the icons. For example, it is not feasible to connect an inspection station to a manufacturing cell.

The function and connection rules for each of these icons are documented within the system. The user can click on an icon to learn the function and the rules of the icon. All the connection rules are implemented in the system as construction rules of the models. As the user creates a model, the AMPS checks the partially completed model immediately for possible local violations of the rules. For example, Figure 16 shows the rules for an assembly station.

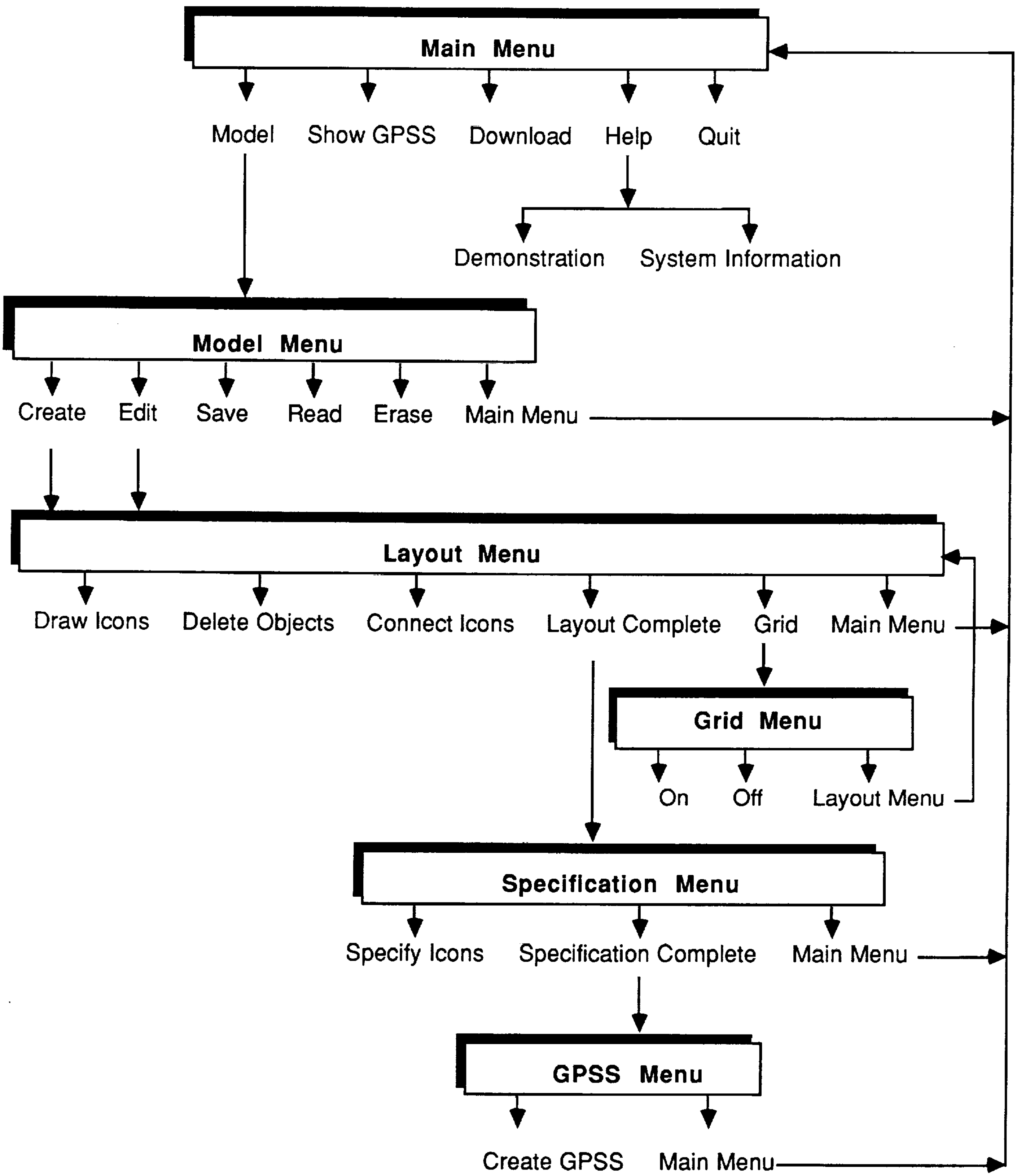


Figure 13. AMPS/Graphics commands

Icons	Function
	Assembly station
	Starting point of an assembly line
	Demand stock point of pull inventory system
	Ending stock point for final product
	Inspection station
	Manufacturing cell
	Stock point for part ordered from outside
	stock point for Push inventory system
	Supply stock point of pull inventory system
	Task station

Figure 14. Library of AMPS/Graphics icons

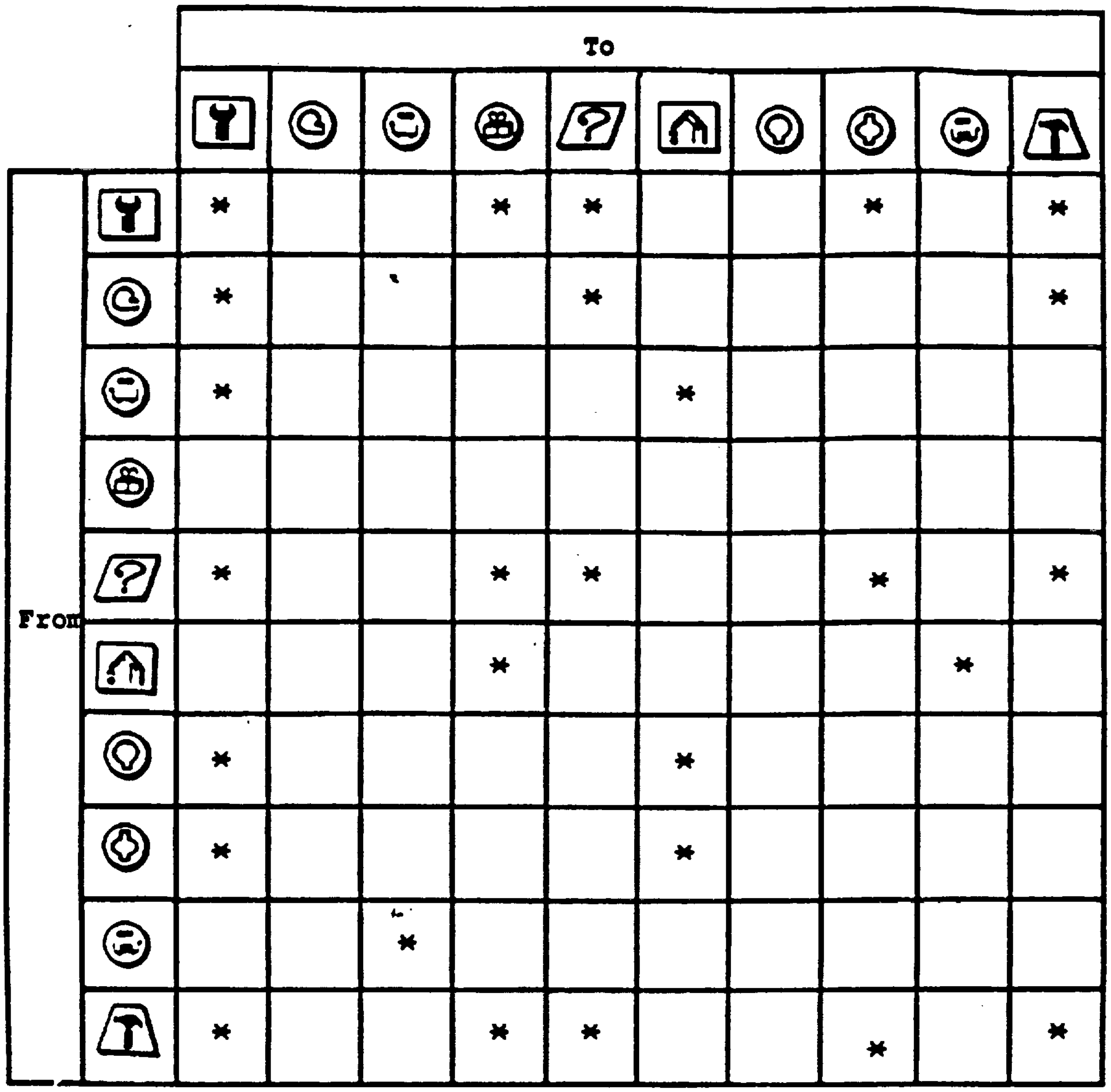


Figure 15. Valid AMPS/Graphics icon connections

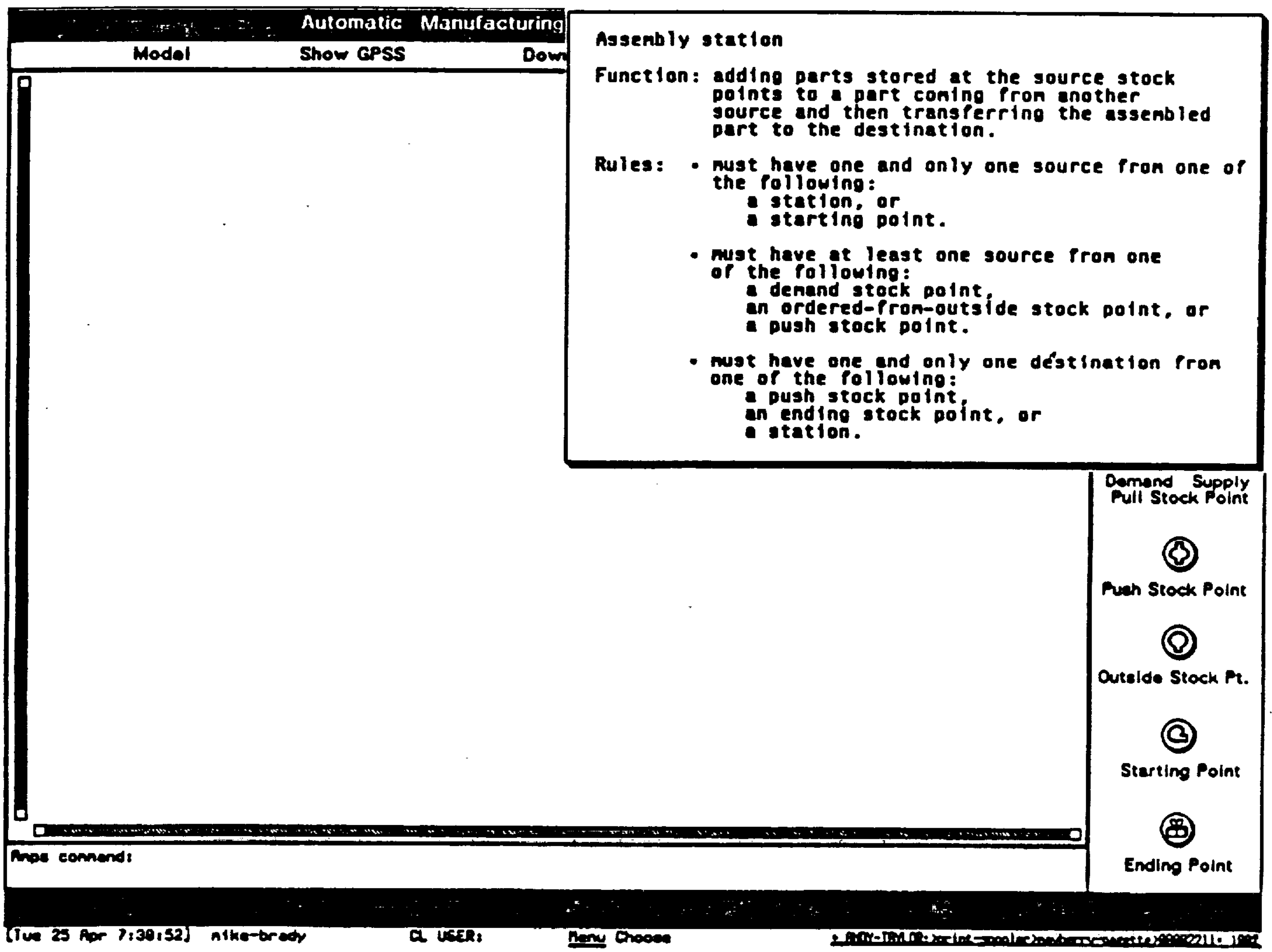


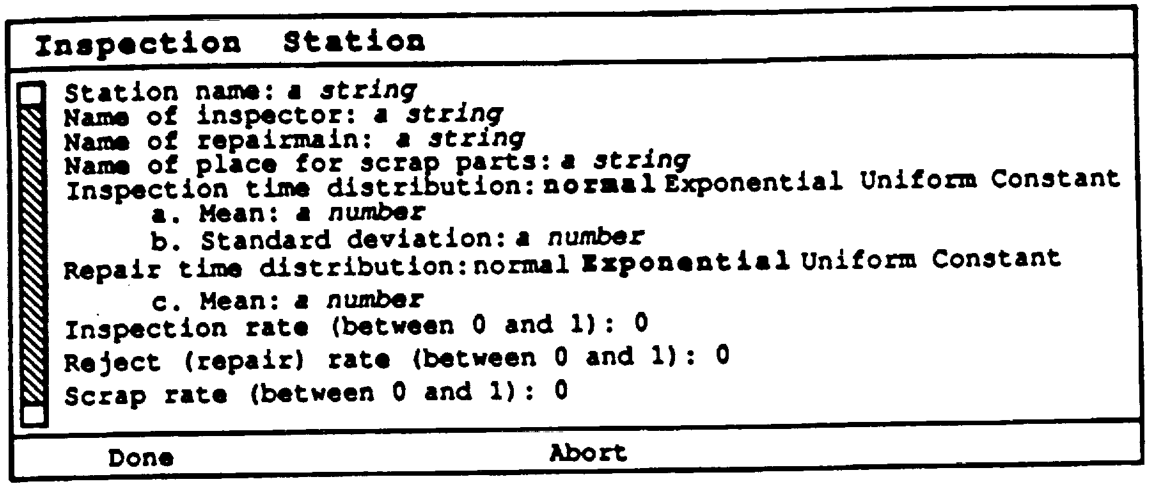
Figure 16. Assembly station rules

When the process flow has been completely drawn, the AMPS/Graphics will check the completeness of the structure. After the layout has passed the check for completeness, the user enters the parameters of the manufacturing system. The user then clicks on each icon to input the specification. A parameter menu will pop up on the screen. Figure 17 shows the parameter menu of an inspection station. The user can move the cursor to each field to enter the data. The system then performs additional checking. For example, the AMPS will check whether the data are the right types for the fields. The AMPS will make certain that an initial inventory level is not larger than the capacity.

6.5.2 Sample Problem

Figure 18 is an example of a simple manufacturing system formulated using the AMPS/Graphics system. The manufacturing system consists of an assembly line, MAIN and two assembly stations, STA1 and STA2. The assembly line produces part A. Station STA1 assembles part C to the incoming part and passes it to station STA2. Station STA2 then assembles part B to the incoming part from station STA1 and produces part A. Part C is supplied through a pull inventory control system from manufacturing cell MC. A part C is made of parts D and E at the manufacturing cell MC. Parts B, D, and E are supplied from outside sources.

Parts arriving at the assembly line follow the exponential distribution with a mean of 100. The assemble time of each of the two stations is a constant 100. Station STA1 requires one part C and station STA2 requires one part B for an assembly. The stock point to hold the final product, part A, has a capacity of 1000 units.



Inspection Station

Station name: a *string*
Name of inspector: a *string*
Name of repairmain: a *string*
Name of place for scrap parts: a *string*
Inspection time distribution: **normal** Exponential Uniform Constant
a. Mean: a *number*
b. Standard deviation: a *number*
Repair time distribution: normal **Exponential** Uniform Constant
c. Mean: a *number*
Inspection rate (between 0 and 1): 0
Reject (repair) rate (between 0 and 1): 0
Scrap rate (between 0 and 1): 0

Done Abort

Figure 17. Typical parameter input

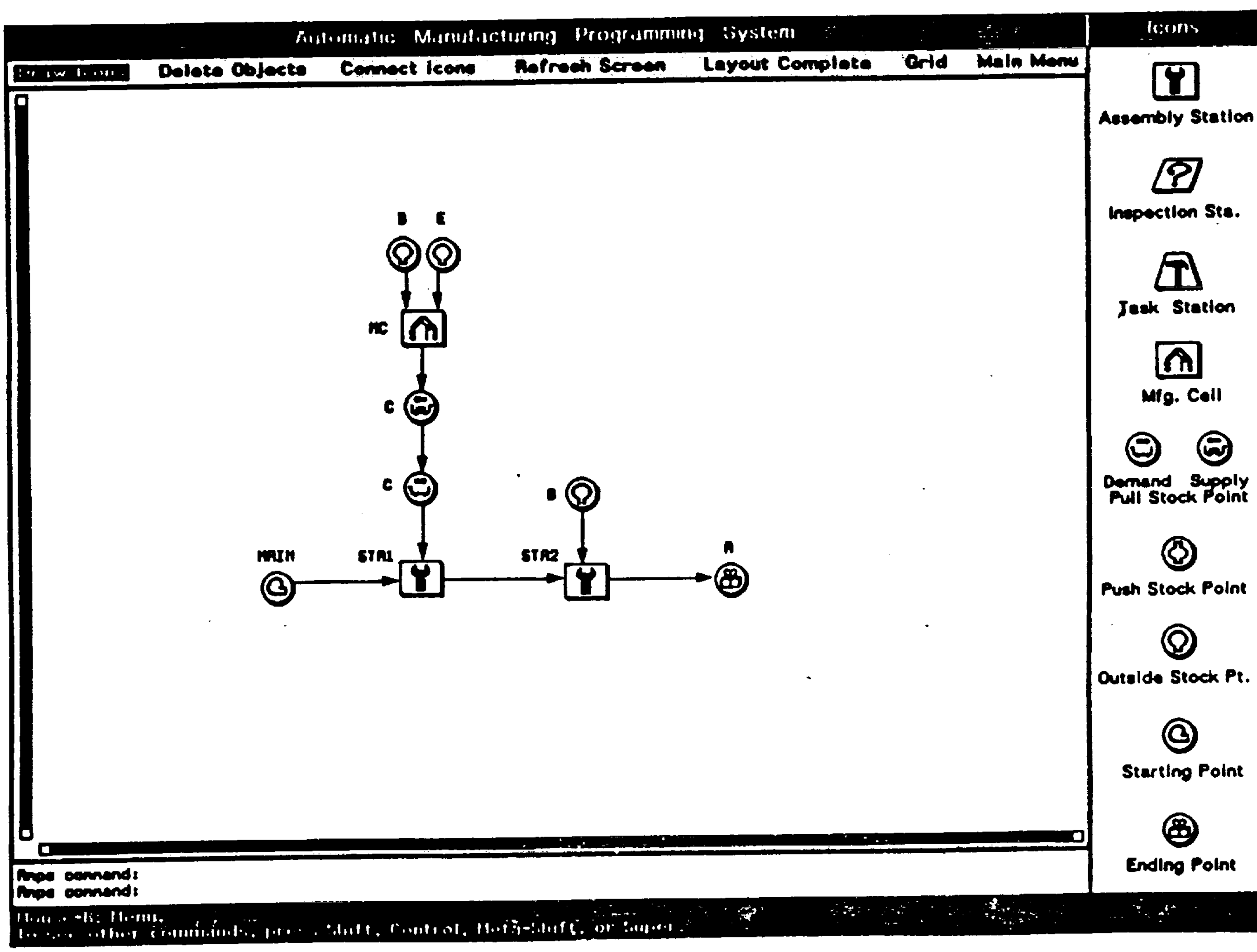


Figure 18. Sample manufacturing system

Part C is used at station STA1 and is manufactured at manufacturing cell MC. A pull inventory system controls the production and shipment of part C, which is represented by a pair of supply and demand stock points. A vehicle WGIG is used to move the carts between the stock points. The time to move the carts is 10. Each cart has a capacity of 4 parts C. Initially there is a cart of parts C at each of the supply and demand stock points. Parts B, D, and E are supplied from outside sources. Initially there are 1000 units for each part type. Manufacturing cell MC makes part C. One part D and two parts E are used to make one part C. The manufacturing time is 100 and there is no setup time.

The model is created by selecting the Model command from the Main Menu and the Create command from the Model Menu (See Figure 13). The actual layout of the model is created by using the commands Draw Icons and Connect Icons in the Layout Menu.

After the model has been completely drawn, the Layout Complete option is selected to start specifying the model parameters. Figure 19 shows a portion of the model parameters. To specify an icon the user simply clicks on the icon when the AMPS is in the Specification Menu.

Both the layout and the parameters can be saved for future use through the Save command in the Model Menu. At the completion of the problem specification, the user selects the Specification Complete command to end the model specification. The system then leads the user to the GPSS Menu command to create the corresponding simulation code in the target language GPSS/PC.

Appendix D contains another sample problem. Included in this appendix are layout of the manufacturing system, a listing of the input parameters, and a complete listing of the GPSS simulation model.

Starting Point of Line
  Name of line: MAIN
  Interarrival time distribution: Constant
    Constant: 100

Final Product from Assembly Line
  Part name: A
  Capacity and initial inventory at the stock point:
    Maximum number of parts at stock point: 1000
    Initial number of parts at stock point: 0

Demand Stock Point
  Part name: C
    In a pull system, parts are assumed to be ordered, made, and shipped by carts. Two stockpoints: supply and demand are created.
  Capacity and initial inventory at the stock point:
    Current cart capacity (number of parts per cart): 4
    Initial number of carts at demand stock point: 1
    Initial number of carts at supply stock point: 1
  Vehicle used to move carts between stock points: wgig
  Moving time distribution: Constant
    Constant: 10

Supply Stock Point
  Part name: C
    In a pull system, parts are assumed to be ordered, made, and shipped by carts. Two stockpoints: supply and demand are created.
  Capacity and initial inventory at the stock point:
    Current cart capacity (number of parts per cart): 4
    Initial number of carts at demand stock point: 1
    Initial number of carts at supply stock point: 1
  Vehicle used to move carts between stock points: wgig
  Moving time distribution: Constant
    Constant: 10

Ordered from outside
  Part name: D
  Capacity and initial inventory at the stock point:
    Maximum number of parts at stock point: 1000
    Initial number of parts at stock point: 1000
  Will Part D be replenished during the simulation? No

Figure 19. Partial parameter input

6.6 AMPS/PC/SIMAN

The basic AMPS/PC system in Section 6.4 was modified to create SIMAN (Pegden 1985) rather than GPSS/PC code. This system used the identical Interactive Dialogue Interface (IDI) as the AMPS/PC system. However, the automatic code generator program was rewritten to create code in the target simulation language SIMAN. A listing of the SIMAN macros is given in Appendix C.

The system was written by programmer B in Turbo C on an IBM/PC. The system consisted of 1,600 lines of code. The code production was 533 lines per month. This system has not been documented and has not been submitted to NASA COSMIC.

7.0 SYSTEM EVALUATION

The concepts developed in AMPS have been used to model three real world problems. The first system was a Flexible Manufacturing System (FMS) at Rexham Speedring Inc., in Cullman, Alabama. The FMS consisted of 18 stations and nine alien stations. The FMS makes four different parts with each requiring 47, 31, 22 and 22 operations respectively (Schroer 1988).

The second system was a 25 station assembly line at SCI Manufacturing Inc. in Huntsville, Alabama. The line assembles a health monitoring device (Schroer 1988). The third system was a twelve station Unit Production System (UPS) at Camptown Togs, Inc. in Clanton, Alabama (Schroer and Ziemke 1989).

The following observations are made based on the above implementations:

° The problem domains were sufficiently different that the AMPS user interface could not be used in defining the problem specification.

- The library of GPSS modules were used extensively in writing the simulation models. For the FMS model, several additional simulation modules were developed.

- By using the library of GPSS modules, the UPS model was written and validated in less than four hours as compared to forty hours without the use of the modules.

- The use of the GPSS modules caused the resulting simulation code to be structured code and well documented.

8.0 PUBLICATIONS

The following is a list of publications resulting from the research supported by NASA Grant NAG8-641 and NASA contract NAS8-36995.

1. "Use of Simulation Generators in Modeling Manufacturing Systems," F.T. Tseng and B.J. Schroer, Proceedings Southeastern Computer Simulation Conference, October 1987, Huntsville, AL, pp. 149-153.
2. "LISP-Based Simulation Generators for Modeling Complex Space Processes," F.T. Tseng, B.J. Schroer, and W.S. Dwan, Proceedings 3rd Conference on Artificial Intelligence for Space Applications, November 1987, Huntsville, AL, pp. 243-247.
3. "Modeling Complex Manufacturing Systems Using Simulation," B.J. Schroer and F.T. Tseng, Proceedings 1987 Winter Simulation Conference, December 1987, Atlanta, GA, pp. 677-682.

4. A Simulation Assistant for Modeling Manufacturing Systems, B.J. Schroer, F.T. Tseng, and W.S. Dwan, UAH Research Report No. 659, January 1988.

5. "Constructing Discrete Event Models in GPSS Using a Simulation Assistant," F.T. Tseng and B.J. Schroer, ORSA/TIMS, Washington, D.C., May 1988. (presentation only)

6. "Automatic Manufacturing Programming Systems (AMPS), B.J. Schroer, F.T. Tseng, and J.W. Wolfsberger, Proceedings for Conference on Space and Military Applications of Automation and Robotics, Huntsville, AL, June 1988, pp. 451-459.

7. "Automatic Programming of Manufacturing Simulation Models," B.J. Schroer, F.T. Tseng, S.X. Zhang, and J.W. Wolfsberger, Proceedings 1988 Summer Computer Simulation Conference, Seattle, WA, July 1988, pp. 569-574.

8. "Modeling Complex Manufacturing Systems Using Discrete Event Simulation," B.J. Schroer and F.T. Tseng, Computers and Industrial Engineering, Vol. 14, No. 4, 1989.

9. Automatic Network Programming System (ANPS), F.T. Tseng, S.X. Zhang, and B.J. Schroer, UAH Research Report No. 704, June 1988.

10. "Using Automatic Programming for Simulating Reliability Network Models," F.T. Tseng, B.J. Schroer, S.X. Zhang, and J.W. Wolfsberger, Proceedings Fourth Conference on Artificial Intelligence for Space Applications, Huntsville, AL, November 15-16, 1988.

11. "Automatic Programming Assistant for Network Simulation Models," F.T. Tseng, B.J. Schroer, S.X. Zhang, and J.W. Wolfsberger, Proceedings 1988 Winter Simulation Conference, December 12-14, 1988, San Diego, CA, pp. 240-245.

12. Automatic Manufacturing Programming System (AMPS) User's Manual, B.J. Schroer, F.T. Tseng, and W.S. Dwan, UAH Research Report No. 710, September 1988.

13. Automatic Manufacturing Programming System/Graphics User's Manual, F.T. Tseng, B.J. Schroer, and W.S. Dwan, UAH Research Report No. 788, August 1989.

14. "Simulation of Reliability Network Models Using Automatic Programming Techniques", S.X. Zhang, B.J. Schroer, Y.C. Feng, and R.T. Crumbly, Proceedings Beijing International Conference on System Simulation and Scientific Computing, Beijing, China, October, 1989, pp 542-546.

15. "A Simulation Assistant for Modeling Manufacturing Systems", B.J. Schroer, Simulation, Vol. 53, No. 5, November 1989, pp. 201-206.

16. "An Intelligent Assistant for Manufacturing System Simulation", B.J. Schroer and F.T. Tseng, International Journal of Production Research, Vol. 27, No. 10, 1989, pp. 1665-1683.

17. "Combining Software Engineering Principles with Discrete Event Simulation", B.J. Schroer, F.T. Tseng, and S.X. Zhang, Proceding 1989 Winter Simulation Conference, Washington DC, December 1989, pp. 828-833.

18. "Software Engineering and Simulation", S.X. Zhang, B.J. Schroer, S.L. Messimer, and F.T. Tseng, Proceedings Third International Software for Strategic Systems Conference, Huntsville, AL, February 1990.

19. "Improving the Manufacturing Simulation Modeling Environment", B.J. Schroer, Manufacturing Review, Vol. 2, No. 4, pp. 283-289.

20. "Applying Software Engineering to Discrete Event Simulation", Proceedings Eastern Multiconference Computer Simulation, Nashville, TN, April 1990.

9.0 CONCLUSIONS

9.1 Comparison of the AMPS/GRaphics System with the AMPS System

Both AMPS and AMPS/Graphics were written in Lisp on the Symbolics 3620 machine. The AMPS/Graphics used the Symbolics system dependent features such as the flavors (frame) window, and the graphics function. Also, AMPS/Graphics use object oriented programming concepts. The adoption of the above features greatly simplified the programming effort for such a complicated system. However, these system dependent features also make the conversion of the AMPS/Graphics to other types of machines very difficult. On the other hand, the AMPS system used very few system dependent features. Most of the statements in AMPS are Common Lisp compatible. Therefore, it is much easier to convert AMPS to other platforms. For example, the AMPS system has been successfully ported converted to a TI Explorer with only minor modifications.

The AMPS system provides an Interactive Dialogue Interface (IDI) for the user to create the model. In AMPS, the user must follow the preset logic system and answer a series of questions prompted in constructing a model. That is, the user is in a passive role. The AMPS system controls the main logic. The AMPS system allows the user to make only very limited modifications throughout the development process. Also, the user must remember the stage of the development process. Consequently, it is difficult to visualize the development process of the model in AMPS.

The AMPS/Graphics has a an Interactive Graphic Interface (IGI) through which the user builds the model mainly by icons. The user can start building the model from any part of the model. Also, the user can always see the partially completed model on the screen. The AMPS/Graphics system allows the user to modify any part of the model throughout the development process. Consequently, it is much easier to build a model and to trace the logic by the AMPS/Graphics. Furthermore, a graphical model is also a much better communicative model than a descriptive model.

The AMPS/Graphics system provides several help features. For example, on-line documentation of each icon can be obtained by clicking a button. The documentation shows the function of each icon and the connection rules with other icons. In the construction process, if a mistake is made, the system will immediately give the appropriate error message. The AMPS system does not have these help features.

The models created by the AMPS/Graphics can be saved and then modified through the IGI interface. The corresponding simulation program will then be modified automatically. The AMPS system does not have this capability.

It is much slower to design a user friendly system such as AMPS/Graphics at the beginning of the design process because of the many factors to be considered. However, once the basic framework of the system is completed, a system such as AMPS/Graphics is much easier to modify and expand. For example, it is rather easy to add a new facility icon or to change a model construction rule. A carefully designed system should be flexible enough to add or remove a construct from the system with only minor effort. On the other hand, a system like AMPS is easier to initially design, and therefore is ideal to serve as a prototype. However, any change after the initial design requires a major modification to the system.

The AMPS/Symbolics system makes use of some advantages of the Symbolics machine. For example, the system automatically checks for some types of errors, executes much faster, and has a large amount of memory available. However, currently none of the popular commercially available discrete simulation languages, such as GPSS, is available on the Symbolics. The simulation programs must be downloaded to an IBM-PC to run the simulation. On the other hand, the AMPS/PC system is much slower than the AMPS/Symbolics. The small memory of the PC's also limits any reasonably large models to be constructed on the AMPS/PC.

9.2 Summary

In summary, an Automatic Programming (AP) system, such as AMPS and ANPS, offers a number of advantages for improving the simulation modeling environment. These advantages include:

- Rapid prototyping - Once the necessary library of simulation modules has been written, the AP system permits the user to rapidly construct a model. As a result, the AP system produces executable simulation code that is syntax error free.

- Software correctness - Correct simulation software requires the definition of a complete and formal set of model requirements. An AP system forces the user to completely define these requirements.

- Improved clarity of simulation code - The simulation code generated by the AP system is structured code that is easy to read, trace, and modify. An added benefit is embedded code documentation.

- Increased productivity - By using an AP system, the modeler should have an increased productivity in the lines of simulation code written per hour.

- Automatic documentation - Instead of changing program code, the user modifies the problem specification through the AP system's user interface. The AP system then rewrites a new simulation model. Therefore, the problem specification file always reflects the current configuration, or documentation, of the problem.

- Software reusability - Software reusability refers to the ability of new simulation models to use element of other models. Large collections, or libraries, of reusable program modules can be defined, making it possible to develop new models by writing only a small amount of new code. The library of GPSS modules provides the basic building blocks for the simulation model. This library is constantly being updated and expanded as the AP system is used in other domains.

- Software compatibility - Software compatability is the ability of program modules to be interfaced with other simulation code. An AP system designed with expansion in mind and as generic as possible will be easier to modify as additional requirements are defined.

- Extendability - Since an AP system operates in a structural environment, the overall software maintenance is less difficult. Software designed and developed using an AP system will have each data element and related processes grouped into one location, making modifications simpler.

- Reduced simulation knowledge - An advantage of an AP system is a reduction in the modeler's knowledge of the simulation language.

There are also a number of disadvantages of an AP system such as AMPS and ANPS. These disadvantages include:

- Domain specific - Most AP systems are very domain specific. Therefore, the systems can only model a very limited class of problems. To model a slightly different problem in a similar domain may require additional modules and modifications to the user interface.

- Library robustness - A related disadvantage is the robustness of the library of predefined modules. Generally skilled GPSS programmers are needed to write a new modules.

° Memory and execution time - Another disadvantage is that AP systems require more memory and execute slower than a nonstructured equivalent simulation program. However, this disadvantage is not as significant as in prior years because computers are now faster and have more memory.

In comparing the IDI and IGI for the Symbolics systems, the following observations are made:

° The IGI had 3,500 lines of code versus 1,500 lines for the IDI. Interestingly, the code production was similar for both systems.

° The IGI, or object oriented approach, is preferred by the user.

10.0 ACKNOWLEDGEMENTS

In addition to the funds received from NASA/MSFC, a contract was received in 1988 from the Science, Technology, and Energy Division, Alabama Department of Economic and Community Affairs (ADECA) (Reference Contract ADECA-UAH-9001) and a grant in 1989 from the Alabama Industrial Development Training (AIDT).

11.0 REFERENCES

Balci, O. 1986. "Requirements for Model Development Environments," Computers and Operations Research, Vol. 13, No. 1, pp. 53-67,

Balci, O., and R. E. Nance. 1987. "Simulation Support: Prototyping the Automation-based Paradigm," Proceedings of the Winter Simulation Conference, Atlanta, GA, 495-502.

Barr, A. and E.A. Feigenbaum. 1982. The Handbook of Artificial Intelligence, Vol. 2, W. Kaufman, Inc., CA.

Brazier, M.K. and R.E. Shannon. 1987. "Automatic Programming of AGVS Simulation Models," 1987 Winter Simulation Conference, Atlanta, GA, (December) pp. 703-708.

Ford, D.R. and B.J. Schroer. 1987. "An Expert Manufacturing Simulation System." Simulation, Vol. 48,, No. 5, (May) pp. 193-200.

GPSS/PC Reference Manual. 1986. Minuteman Software, Stow, MA.

Haddock, J. and R. P. Davis. 1985. "Building a Simulation Generator for Manufacturing Cell Design and Control." Annual International Industrtial Engineering Spring Conference Proceedings, Los Angelos, CA, (May) pp. 237-244.

Heidorn, G.E. 1974. "English as a Very High Level Language for Simulation Programming." SIGPLAN Notices, Vol. 9, No. 4, pp. 91-100.

Henriksen, J. D. 1983. "The Integrated Simulation Environment (Simulation Software of the 1990s)," Operations Research 31, pp. 1053-1073.

Khoshnevis, B. and A.P. Chen. 1986. "An Expert Simulation Model Builder." Intelligent Simulation Environment, Society for Computer Simulation, Vol. 17,No. 1, pp. 129-132.

Murray, K.J. and S.V. Sheppard. 1988. "Knowledge-based Simulation Model Specification," Simulation, Vol. 50, No. 3, (March) pp. 112-119.

Overstreet, C. M. and R. E. Nance. 1985. "A Specification Language to Assist in Analysis of Discrete Event Simulation Models, Communications ACM 28, pp. 190-201.

Pegden, C. Dennis, 1985. Introduction to SIMAN, Systems Modeling Corp., College Station, PA.

Phillips, D.T., G.L. Curry, B.L. Deuermeyer, M. Wortman, and J.P. Sitarik. 1989. "SEMATECH: IE at work in the trenches to meet world competition." Industrial Engineering, Vol. 21, No, 12, December, pp. 36-44.

Pidd, M. 1984. Computer Simulation in Management Science. Wiley, Chichester.

Schroer, B.J. and M.C. Ziemke. 1989. A Simulation Model of a Unit Production System, UAH Report 789, August.

Schroer, B.J. 1988. A Simulation Model of a Assembly Line, UAH Report 696, June.

Schroer, B.J. 1988. A Simulation Model of a Manufacturing Cell. University of Alabama in Huntsville, Research Report No. 676, April.

Schroer, B.J., F.T. Tseng, and W.S. Dwan. 1988. A Simulation Assistant for Modeling Manufacturing Systems, University of Alabama in Huntsville Research Report No. 659, January.

Schroer, B.J., F.T. Tseng, and W.S. Dwan. 1988. Automatic Manufacturing Programming System (AMPS) User's Manual, UAH Research Report No. 720, September.

Schroer, B.J. 1969. "Saturn V Prelaunch Systems Simulation Model for a Launch Opportunity Containing Multiple Launch Windows," Proceedings Third Conference on Applications of Simulation, Los Angelos, December, pp. 503-511.

Snyder, J.E., E.R. Bennich, and Y.H. Lindsey. 1967. "Implementation of Advanced Simulation Techniques for Predicting the Saturn V Launch Vehicle System Behavior," Journal of Spacecraft and Rockets, Vol. 4, No. 8, pp. 998-1002.

Strandridge, C. R. 1983. "Software Aids for Simulation," Simulation 41, 193.

Turbo Prolog 2.0 Reference Guide. 1986 . Borland International, Scotts Valley, CA.

Turbo Pascal 4.0 Reference Guide, 1988 . Borland International, Scotts Valley, CA.

Turbo C 2.0 Users Guide. 1988 . Borland International, Scotts Valley, CA.

Appendix A

GPSS macros for AMPS

GPSS Assembly station subroutine

```
2370 ***************************************
2380 *         ASSEMBLY STATION            *
2390 ***************************************
2400 ASM     ASSIGN       3,MX$STAN(P2,1)
2410         ASSIGN       7,MX$STAN(P2,2)
2420         ASSIGN       6,MX$STIME(P2,1)
2430         ASSIGN       8,1
2440         ASSIGN       9,2
2450         QUEUE        P3
2460 PAQ     ASSIGN       8+,2
2470         ASSIGN       9+,2
2480         ASSIGN       5,MX$STAN(P2,P8)
2490         ASSIGN       10,MX$PART(P5,1)
2500         ASSIGN       20,MX$STAN(P2,P9)
2510         QUEUE        P10
2520         TRANSFER     SBR,TAKEP,RTRN2
2530         DEPART       P10
2540         LOOP         7,PAQ
2550         SEIZE        P3
2560         DEPART       P3
2570         ADVANCE      V*6
2580         RELEASE      P3
2590         TRANSFER     P,RTRN1,1
```

GPSS Task station subroutine

```
1980 ***************************************
1990 *         TASK STATION                *
2000 ***************************************
2010 TASK    ASSIGN       3,MX$STAN(P2,1)
2020         ASSIGN       6,MX$STIME(P2,1)
2030         QUEUE        P3
2040         SEIZE        P3
2050         DEPART       P3
2060         ADVANCE      V*6
2070         RELEASE      P3
2080         TRANSFER     P,RTRN1,1
```

GPSS Inspection station subroutine

```
**************************************
*      INSPECTION STATION            *
**************************************
INSP    ASSIGN      3,MX$STAN(P2,2)
        ASSIGN      4,MX$IPERC(P3,1)
        ASSIGN      5,MX$ITIME(P3,1)
        ASSIGN      6,MX$ITIME(P3,2)
        QUEUE       MX$STAN(P2,1)
        DEPART      MX$STAN(P2,1)
        TRANSFER    ,FN*4
CHECK   QUEUE       MX$ISTA(P3,1)
        SEIZE       MX$ISTA(P3,1)
        DEPART      MX$ISTA(P3,1)
        ADVANCE     V*5
        RELEASE     MX$ISTA(P3,1)
        ASSIGN      4,MX$IPERC(P3,2)
        TRANSFER    ,FN*4
REPAIR  QUEUE       MX$ISTA(P3,2)
        SEIZE       MX$ISTA(P3,2)
        DEPART      MX$ISTA(P3,2)
        ADVANCE     V*6
        RELEASE     MX$ISTA(P3,2)
        ASSIGN      4,MX$IPERC(P3,3)
        TRANSFER    ,FN*4
PASS    TRANSFER    P,RTRN1,1
SCRAP   QUEUE       MX$ISTA(P3,3)
        DEPART      MX$ISTA(P3,3)
        TERMINATE
```

```
****************************************
*          MANUFACTURING CELL          *
****************************************
MFG      ASSIGN        13,MX$CELL(P12,1)
         ASSIGN        14,MX$CTIME(P12,1)
         ASSIGN        16,MX$CTIME(P12,2)
         QUEUE         P13
         ASSIGN        7,MX$CSIZE(P12,1)
CARTQ    ASSIGN        17,MX$ITEM(P12,1)
         ASSIGN        8,0
         ASSIGN        9,1
PARTQ    ASSIGN        8+,2
         ASSIGN        9+,2
         ASSIGN        5,MX$ITEM(P12,P8)
         ASSIGN        10,MX$PART(P5,1)
         ASSIGN        20,MX$ITEM(P12,P9)
         QUEUE         P10
         TRANSFER      SBR,TAKEP,RTRN2
         DEPART        P10
         LOOP          17,PARTQ
         LOOP          7,CARTQ
FAC      SEIZE         P13
         DEPART        P13
         ADVANCE       V*14
         ADVANCE       V$MTIME
MTIME    FVARIABLE     V*16#MX$CSIZE(P12,1)
         RELEASE       P13
         TRANSFER      P,RTRN3,1
```

GPSS Manufacturing cell subroutine

```
2600 ****************************************
2610 *           INVENTORY CONTROL          *
2620 ****************************************
2630 TAKEP   TEST E       MX$PART(P5,2),1,PULL
2640 PUSH    TEST GE      S*10,P20
2650         LEAVE        *10,P20
2660         TRANSFER     P,RTRN2,1
2670 PULL    ASSIGN       30,MX$CART(P5,1)
2680         TEST GE      S*10,P20,NEEDC
2690 MINUSP  LEAVE        *10,P20
2700         SPLIT        1,USEP
2710         TRANSFER     P,RTRN2,1
2720 NEEDC   ASSIGN       20-,S*10
2730         LEAVE        *10,S*10
2740         SPLIT        1,USEP
2750         TEST GE      S*30,1
2760         LEAVE        *30
2770         ENTER        *10,MX$CSIZE(P5,1)
2780         TEST GE      S*10,P20,NEEDC
2790         LEAVE        *10,P20
2800         SPLIT        1,USEP
2810         TRANSFER     P,RTRN2,1
2820 USEP    TEST G       S*10,0,EMPTYC
2830         TERMINATE
2840 EMPTYC  SPLIT        1,ORDER1
2850         TEST GE      S*30,1
2860         LEAVE        *30,1
2870         ENTER        *10,MX$CSIZE(P5,1)
2880         TERMINATE
2890 ORDER1  ASSIGN       26,MX$FGIG(P5,1)
2900         ASSIGN       16,MX$CTIME(P5,1)
2910         ASSIGN       36,MX$MTIME(P5,1)
2920         QUEUE        P26
2930         SEIZE        P26
2940         DEPART       P26
2950         ADVANCE      V*36
2960         RELEASE      P26
2970         SPLIT        1,GET1F
2980         ASSIGN       12,P5
2990         ASSIGN       15,MX$SCART(P5,1)
3000 GET1C   TRANSFER     SBR,MFG,RTRN3
3010         ENTER        *15,1
3020         TERMINATE
3030 GET1F   ASSIGN       31,MX$SCART(P5,1)
3040         QUEUE        P31
3050         TEST GE      S*31,1
3060         LEAVE        *31,1
3070         DEPART       P31
3080 SEND1F  QUEUE        P26
3090         SEIZE        P26
3100         DEPART       P26
3110         ADVANCE      V*36
3120         RELEASE      P26
3130         ENTER        *30,1
3140         TERMINATE
```

GPSS Inventory transfer subroutine

Appendix B

GPSS macros for ANPS

```
1360 *
1370 *        ACTIVITY TIME SIMULATION GENERATOR
1380 *
1390 VENT_A SEIZE         P2
1395        ASSIGN        99,MX$WORK_TIME(P3,1)
1400        ASSIGN        ETIME,V*99
1405 BACK3  ASSIGN        98,MX$F_TIME(P3,1)
1410        ASSIGN        MTTF,V*98
1420        TEST L        P$MTTF,P$ETIME,NOFAIL
1430        ADVANCE       P$MTTF
1440        ASSIGN        ROW,P3
1450        TRANSFER      SBR,FAIL,RTRN1
1460        ASSIGN        REST_TIME,V$TIME3
1470 TIME3  FVARIABLE     P$ETIME-P$MTTF
1480        ASSIGN        ETIME,P$REST_TIME
1490        TRANSFER      ,BACK3
1500 NOFAIL ADVANCE       P$ETIME
1510        RELEASE       P2
1520        TRANSFER      P,RTRN2,1
1530 *
1830 *
1832 *      CONTINUOUS ACTIVITY TIME SIMULATION GENERATOR
1834 *
1840 VENT_B SEIZE         P2
1842        ASSIGN        98,MX$F_TIME(P3,1)
1843        SAVEVALUE     FTS,V*98
1845 TIME9  FVARIABLE     X$FTS
1850 TIME8  FVARIABLE     X$FTS/100
1855        TEST L        V$TIME9,100,BACK6
1860        ASSIGN        TIM3,1
1865        ASSIGN        BSUM,V$TIME9
1870        TRANSFER      ,BACK5
1875 BACK6  ASSIGN        TIM3,V$TIME8
1880        ASSIGN        BSUM,100
1885 BACK5  ASSIGN        NR_LOOPS,P$BSUM
1890 BACK4  GATE LR       MX$SWITCH1(P3,1),ENDA
1895        ADVANCE       P$TIM3
1900        LOOP          NR_LOOPS,BACK4
1905        ASSIGN        ROW,P3
1910        TRANSFER      SBR,FAIL,RTRN1
1915        TRANSFER      ,BACK5
1920 ENDA   RELEASE       P2
1925        TRANSFER      P,RTRN2,1
```

```
1060 *
1070 *         ACTIVITY FAILURE SIMULATION GENERATOR (DIRECTLY)
1080 *
1090 FAIL      ASSIGN         NR_ACTS,X$ACTS
1100           ASSIGN         COL,1
1102           ASSIGN         100,MX$R_TIME(P$ROW,1)
1105           MSAVEVALUE     R1_TIME,P$ROW,1,V*100
1110 BACK0     TEST NE        MX$ACT_NAME(P$ROW,P$COL),0,AA
1120           GATE U         MX$ACT_NAME(P$ROW,P$COL),AA
1130           SPLIT          1,AB
1140           TRANSFER       ,AA
1150 AB        MARK           DELAY
1160           PREEMPT        MX$ACT_NAME(P$ROW,P$COL)
1170           ASSIGN         ADELAY,MP$DELAY
1180           TEST LE        P$ADELAY,0,XACT_DELAY
1190           MSAVEVALUE     R2_TIME,P$ROW,P$COL,MX$R1_TIME(P$ROW,1)
1200 BACK1     ADVANCE        MX$R2_TIME(P$ROW,P$COL)
1210           BUFFER
1220           RETURN         MX$ACT_NAME(P$ROW,P$COL)
1230           TERMINATE
1240 AA        ASSIGN         COL+,1
1250           LOOP           NR_ACTS,BACK0
1260           TRANSFER       P,RTRN1,1
```

```
1270 *
1280 *         LOGIC WHEN ACTIVITY ALREADY IS INTERRUPTED
1290 *
1300 XACT_DELAY ADVANCE
1310           MSAVEVALUE     R2_TIME,P$ROW,P$COL,V$NEWDELAY
1320 NEWDELAY FVARIABLE       MX$R1_TIME(P$ROW,1)-P$ADELAY
1330           TEST L         MX$R2_TIME(P$ROW,P$COL),0,BACK1
1340           MSAVEVALUE     R2_TIME,P$ROW,P$COL,0
1350           TRANSFER       ,BACK1
```

Appendix C

SIMAN macros for AMPS

```
;       ***   Assembly station model    ***
 ASM          ASSIGN:A(1)=M;
              ASSIGN:A(2)=0;
 BACKASM      ASSIGN:A(2)=A(2)+1;
              BRANCH,1:
                IF,A(2).GT.A(4),OUT:
                ELSE,DOWN1;
 DOWN1        BRANCH,2:
                ALWAYS,BACKASM:
                ALWAYS,DOWN2;
 DOWN2        ASSIGN:M=3+2*A(2);
              ASSIGN:A(5)=A(M);
              ASSIGN:A(6)=A(M+1);
              ASSIGN:A(11)=A(5)+3;
              BRANCH,2:
                 IF,P(A(11),2).EQ.2,PULL1:
                 ALWAYS,PICKPT;
;
 PICKPT       ASSIGN:M=A(5);
              QUEUE,M+1;
              SEIZE:PART(M),A(6);
 DOWN3        ASSIGN:M=A(1);
              QUEUE,M;
              SEIZE:STATIONN(M);
              DELAY:ED(A(3));
              RELEASE:STATIONN(M):NEXT(LOOP);
;
 PULL1        ASSIGN:A(11)=A(5)+3;
              ASSIGN:A(12)=A(5)+1;
              BRANCH,1:
                IF,NR(A(12)).GE.P(A(11),4),PICKCAR:
                ELSE,OUT;
;
 PICKCAR      ASSIGN:M=P(A(11),5);
              QUEUE,M+4;
              SEIZE:CAR(M);
              QUEUE,M+5;
              SEIZE:SCAR(M);
              BRANCH,2:
                ALWAYS,VECHIC:
                ALWAYS,MAKE;
;
 VECHIC       ASSIGN:M=P(A(11),8);
              QUEUE,M+7;
              SEIZE:ROBOT(M);
              ASSIGN:A(12)=P(A(11),9);
              DELAY:ED(A(12));
              RELEASE:ROBOT(M);
              ASSIGN:M=P(A(11),5);
              RELEASE:CAR(M);
              ASSIGN:M=A(5);
              RELEASE:PART(M),P(A(11),4):DISPOSE;
```

```
MAKE       ASSIGN:A(1)=A(5);
           ASSIGN:A(2)=0;
           ASSIGN:A(4)=P(A(11),13);
BACKMAK    ASSIGN:A(2)=A(2)+1;
           BRANCH,1:
             IF,A(2).GT.A(4),OUT:
             ELSE,DOWNM1;
DOWNM1     BRANCH,2:
             ALWAYS,BACKMAK:
             ALWAYS,DOWNM2;
DOWNM2     ASSIGN:A(3)=2*A(2)+12;
           ASSIGN:A(5)=P(A(11),A(3));
           ASSIGN:A(3)=A(3)+1;
           ASSIGN:A(6)=P(A(11),A(3));
           ASSIGN:M=A(5);
           BRANCH,2:
             ALWAYS,MAKEPT:
             ALWAYS,CHKPUL;
MAKEPT     ASSIGN:A(12)=A(6)*P(A(11),4);
           QUEUE,M+12;
           SEIZE:PART(M),A(12);
DOWNM3      ASSIGN:M=P(A(11),10);
           QUEUE,M+6;
           SEIZE:MCELL(M);
           ASSIGN:A(12)=P(A(11),12);
           DELAY:ED((A(12))*P(A(11),4);
           ASSIGN:M=P(A(11),10);
           RELEASE:MCELL(M);
           ASSIGN:A(12)=P(A(11),11);
           DELAY:ED((A(12));
           ASSIGN:M=P(A(11),5);
           RELEASE:SCAR(M):DISPOSE;
;
CHKPUL     ASSIGN:A(5)=M;
           ASSIGN:A(11)=A(5)+3;
           BRANCH,1:
             IF,P(A(11),2).EQ.2,PULL1:
             ELSE,OUT;
OUT        DELAY:0.0:DISPOSE;
```

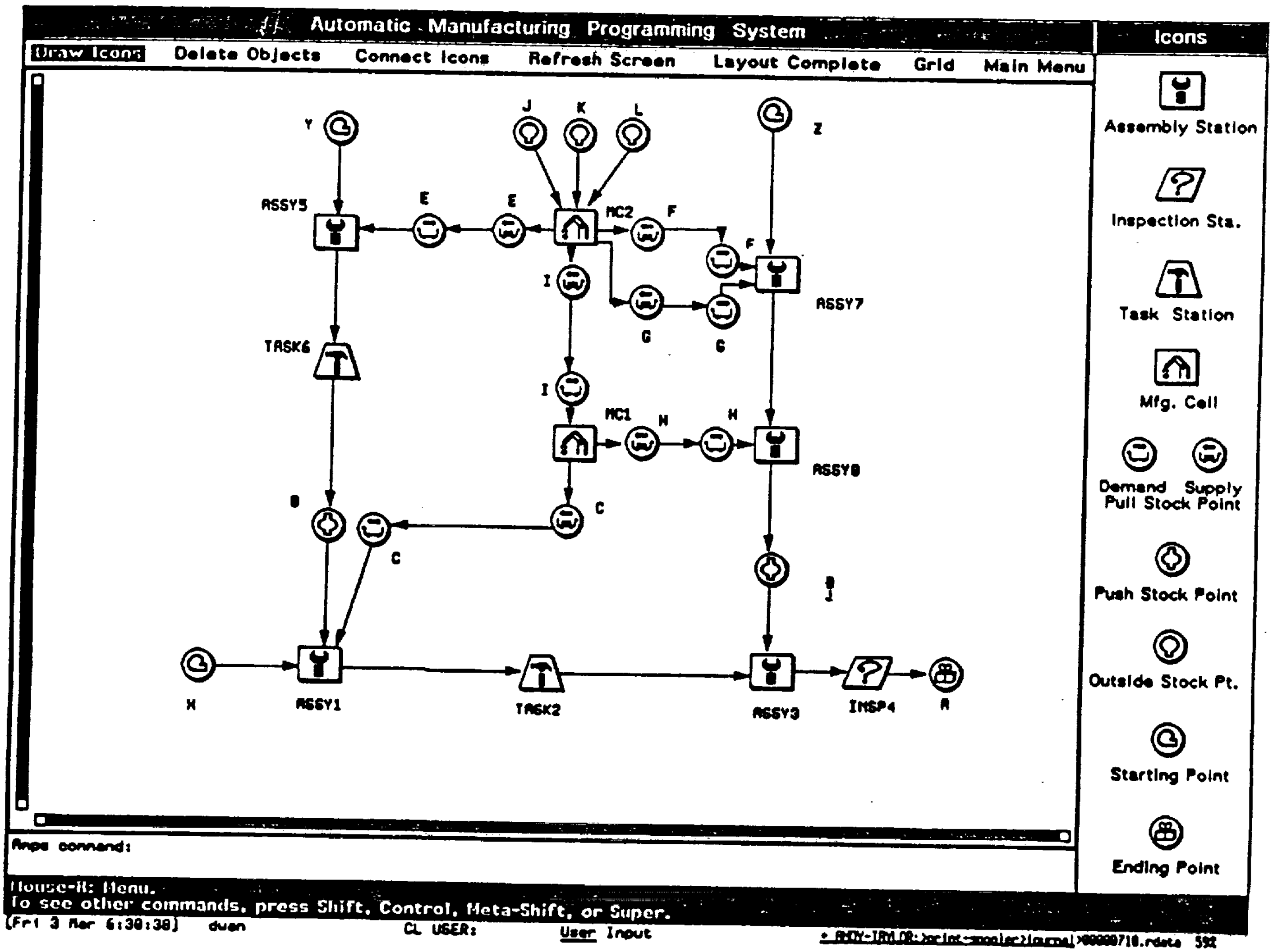
PRECEDING PAGE BLANK NOT FILMED

```
;
;        ***    Inspection station model    ***
 INSP           BRANCH,1:
                  WITH,A(4),CHECK:
                  ELSE,PASS;
 CHECK          QUEUE,M;
                SEIZE:STATIONN(M);
                DELAY:ED(A(3));
                RELEASE:STATIONN(M);
                BRANCH,1:
                  WITH,A(5),REPAIR:
                  ELSE,PASS;
 REPAIR         ASSIGN:M=A(8);
                QUEUE,M+8;
                SEIZE:REPAIR(M);
                DELAY:ED(A(7));
                RELEASE:REPAIR(M);
                BRANCH,1:
                  WITH,A(6),SCRAP:
                  ELSE,PASS;
 SCRAP          DELAY:0:DISPOSE;
 PASS           DELAY:0:NEXT(LOOP);



;        ***    Task station model    ***
 TASK           QUEUE,M;
                SEIZE:STATIONN(M);
                DELAY:ED(A(3));
                RELEASE:STATIONN(M):NEXT(LOOP);
```

PRECEDING PAGE BLANK NOT FILMED



Model-8-stations

PRECEDING PAGE BLANK NOT FILMED

Parameters of Example Model Model-8-Stations

```
Starting Point of Line
   Name of line: Y
   Interarrival time distribution: NORMAL
       Mean: 100
       Standard deviation: 5


Starting Point of Line
   Name of line: X
   Interarrival time distribution: EXPONENTIAL
       Mean: 300


Starting Point of Line
   Name of line: Z
   Interarrival time distribution: NORMAL
       Mean: 75
       Standard deviation: 5


Final Product from Assembly Line
   Part name: A
   Capacity and initial inventory at the stock point:
       Maximum number of parts at stock point: 2000
       Initial number of parts at stock point: 0


Demand Stock Point
   Part name: F
    In a pull system, parts are assumed to be ordered,
    made, and shipped by carts. Two stockpoints: supply
    and demand are created.
   Capacity and initial inventory at the stock point:
       Current cart capacity (number of parts per cart): 4
       Initial number of carts at demand stock point: 4
       Initial number of carts at supply stock point: 4
   Vehicle used to move carts between stock points: TRUCK2
   Moving time distribution: UNIFORM
       Minimum: 6
       Maximum: 14


Supply Stock Point
   Part name: F
    In a pull system, parts are assumed to be ordered,
    made, and shipped by carts. Two stockpoints: supply
    and demand are created.
   Capacity and initial inventory at the stock point:
       Current cart capacity (number of parts per cart): 4
       Initial number of carts at demand stock point: 4
       Initial number of carts at supply stock point: 4
   Vehicle used to move carts between stock points: TRUCK2
   Moving time distribution: UNIFORM
       Minimum: 6
       Maximum: 14
```

```
Supply Stock Point
   Part name: H
     In a pull system, parts are assumed to be ordered,
     made, and shipped by carts. Two stockpoints: supply
     and demand are created.
   Capacity and initial inventory at the stock point:
       Current cart capacity (number of parts per cart): 4
       Initial number of carts at demand stock point: 4
       Initial number of carts at supply stock point: 4
   Vehicle used to move carts between stock points: TRUCK3
   Moving time distribution: UNIFORM
       Minimum: 8
       Maximum: 12


Demand Stock Point
   Part name: H
     In a pull system, parts are assumed to be ordered,
     made, and shipped by carts. Two stockpoints: supply
     and demand are created.
   Capacity and initial inventory at the stock point:
       Current cart capacity (number of parts per cart): 4
       Initial number of carts at demand stock point: 4
       Initial number of carts at supply stock point: 4
   Vehicle used to move carts between stock points: TRUCK3
   Moving time distribution: UNIFORM
       Minimum: 8
       Maximum: 12


Supply Stock Point
   Part name: E
     In a pull system, parts are assumed to be ordered,
     made, and shipped by carts. Two stockpoints: supply
     and demand are created.
   Capacity and initial inventory at the stock point:
       Current cart capacity (number of parts per cart): 4
       Initial number of carts at demand stock point: 4
       Initial number of carts at supply stock point: 4
   Vehicle used to move carts between stock points: TRUCK1
   Moving time distribution: UNIFORM
       Minimum: 8
       Maximum: 12


Demand Stock Point
   Part name: E
     In a pull system, parts are assumed to be ordered,
     made, and shipped by carts. Two stockpoints: supply
     and demand are created.
   Capacity and initial inventory at the stock point:
       Current cart capacity (number of parts per cart): 4
       Initial number of carts at demand stock point: 4
       Initial number of carts at supply stock point: 4
   Vehicle used to move carts between stock points: TRUCK1
   Moving time distribution: UNIFORM
       Minimum: 8
       Maximum: 12


Supply Stock Point
   Part name: I
     In a pull system, parts are assumed to be ordered,
     made, and shipped by carts. Two stockpoints: supply
     and demand are created.
```

```
    Capacity and initial inventory at the stock point:
        Current cart capacity (nunber of parts per cart): 4
        Initial nunber of carts at denand stock point: 4
        Initial nunber of carts at supply stock point: 4
    Vehicle used to nove carts between stock points: TRUCK3
    Moving tine distribution: UNIFORM
        Mininun: 8
        Maxinun: 12


Denand Stock Point
    Part nane: I
      In a pull systen, parts are assuned to be ordered,
      nade, and shipped by carts. Two stockpoints: supply
      and denand are created.
    Capacity and initial inventory at the stock point:
        Current cart capacity (nunber of parts per cart): 4
        Initial nunber of carts at denand stock point: 4
        Initial nunber of carts at supply stock point: 4
    Vehicle used to nove carts between stock points: TRUCK3
    Moving tine distribution: UNIFORM
        Mininun: 8
        Maxinun: 12


Supply Stock Point
    Part nane: G
      In a pull systen, parts are assuned to be ordered,
      nade, and shipped by carts. Two stockpoints: supply
      and denand are created.
    Capacity and initial inventory at the stock point:
        Current cart capacity (nunber of parts per cart): 4
        Initial nunber of carts at denand stock point: 4
        Initial nunber of carts at supply stock point: 4
    Vehicle used to nove carts between stock points: TRUCK2
    Moving tine distribution: UNIFORM
        Mininun: 6
        Maxinun: 14


Denand Stock Point
    Part nane: G
      In a pull systen, parts are assuned to be ordered,
      nade, and shipped by carts. Two stockpoints: supply
      and denand are created.
    Capacity and initial inventory at the stock point:
        Current cart capacity (nunber of parts per cart): 4
        Initial nunber of carts at denand stock point: 4
        Initial nunber of carts at supply stock point: 4
    Vehicle used to nove carts between stock points: TRUCK2
    Moving tine distribution: UNIFORM
        Mininun: 6
        Maxinun: 14


Supply Stock Point
    Part nane: C
      In a pull systen, parts are assuned to be ordered,
      nade, and shipped by carts. Two stockpoints: supply
      and denand are created.
    Capacity and initial inventory at the stock point:
        Current cart capacity (nunber of parts per cart): 4
        Initial nunber of carts at denand stock point: 4
        Initial nunber of carts at supply stock point: 4
    Vehicle used to nove carts between stock points: TRUCK1
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
  Moving time distribution: UNIFORM
      Minimum: 8
      Maximum: 12



Demand Stock Point
   Part name: C
     In a pull system, parts are assumed to be ordered,
     made, and shipped by carts. Two stockpoints: supply
     and demand are created.
   Capacity and initial inventory at the stock point:
       Current cart capacity (number of parts per cart): 4
       Initial number of carts at demand stock point: 4
       Initial number of carts at supply stock point: 4
   Vehicle used to move carts between stock points: TRUCK1
   Moving time distribution: UNIFORM
       Minimum: 8
       Maximum: 12



Push stock point
   Part name: B
   Capacity and initial inventory at the stock point:
       Maximum number of parts at stock point: 2000
       Initial number of parts at stock point: 120



Push stock point
   Part name: D
   Capacity and initial inventory at the stock point:
       Maximum number of parts at stock point: 2000
       Initial number of parts at stock point: 120



Ordered from outside
   Part name: J
   Capacity and initial inventory at the stock point:
       Maximum number of parts at stock point: 10000
       Initial number of parts at stock point: 10000
   Will Part J be replenished during the simulation? No



Ordered from outside
   Part name: K
   Capacity and initial inventory at the stock point:
       Maximum number of parts at stock point: 10000
       Initial number of parts at stock point: 10000
   Will Part K be replenished during the simulation? No



Ordered from outside
   Part name: L
   Capacity and initial inventory at the stock point:
       Maximum number of parts at stock point: 10000
       Initial number of parts at stock point: 10000
   Will Part L be replenished during the simulation? No



Inspection Station
```

```
  Station name: INSP4
  Name of inspector: INSPECTOR
  Name of repairman: REPAIRMAN
  Name of place for scrap parts: SCRAP4
  Inspection time distribution NORMAL
      Mean: 50
      Standard deviation: 5
  Repair time distribution: NORMAL
      Mean: 400
      Standard deviation: 10
  Inspection rate (between 0 and 1): 1
  Reject (repair) rate (between 0 and 1): 0.2
  Scrap rate (between 0 and 1): 0.5


Task Station
  Station name: TASK6
  Task time distribution: NORMAL
      Mean: 100
      Standard deviation: 5


Task Station
  Station name: TASK2
  Task time distribution: NORMAL
      Mean: 300
      Standard deviation: 10


Assembly Station
  Station name: ASSY5
  Parts required for assembly:
    Name of part #1: E
    Number of part #1: 2
  Assembly time distribution: NORMAL
      Mean: 100
      Standard deviation: 5


Assembly Station
  Station name: ASSY1
  Parts required for assembly:
    Name of part #1: B
    Number of part #1: 3
    Name of part #2: C
    Number of part #2: 2
  Assembly time distribution: NORMAL
      Mean: 300
      Standard deviation: 10


Assembly Station
  Station name: ASSY3
  Parts required for assembly:
    Name of part #1: D
    Number of part #1: 4
  Assembly time distribution: NORMAL
      Mean: 300
      Standard deviation: 10
```

```
Assembly Station
  Station name: ASSY8
  Parts required for assembly:
    Name of part #1: H
    Number of part #1: 1
  Assembly time distribution: NORMAL
      Mean: 75
      Standard deviation: 5


Assembly Station
  Station name: ASSY7
  Parts required for assembly:
    Name of part #1: F
    Number of part #1: 2
    Name of part #2: G
    Number of part #2: 1
  Assembly time distribution: NORMAL
      Mean: 75
      Standard deviation: 5


Manufacturing Cell -- F
  Cell name: MC2
  Items required to make the part F:
  Number of item types required: 1
    Name of item #1: L
    Number of item #1: 2
  Setup time for a cart of parts: CONSTANT
      Constant: 0
  Manufacturing time for a part:  NORMAL
      Mean: 10
      Standard deviation: 1


Manufacturing Cell -- E
  Cell name: MC2
  Items required to make the part E:
  Number of item types required: 2
    Name of item #1: J
    Number of item #1: 2
    Name of item #2: K
    Number of item #2: 1
  Setup time for a cart of parts: CONSTANT
      Constant: 0
  Manufacturing time for a part:  NORMAL
      Mean: 10
      Standard deviation: 1


Manufacturing Cell -- I
  Cell name: MC2
  Items required to make the part I:
  Number of item types required: 2
    Name of item #1: J
    Number of item #1: 2
    Name of item #2: K
    Number of item #2: 1
  Setup time for a cart of parts: CONSTANT
      Constant: 0
  Manufacturing time for a part:  NORMAL
      Mean: 5
      Standard deviation: 1
```

```
Manufacturing Cell -- G
   Cell name: MC2
   Items required to make the part G:
   Number of item types required: 1
     Name of item #1: K
     Number of item #1: 2
   Setup time for a cart of parts: CONSTANT
        Constant: 0
   Manufacturing time for a part:  NORMAL
        Mean: 10
        Standard deviation: 1


Manufacturing Cell -- H
   Cell name: MC1
   Items required to make the part H:
   Number of item types required: 1
     Name of item #1: I
     Number of item #1: 1
   Setup time for a cart of parts: CONSTANT
        Constant: 0
   Manufacturing time for a part:  NORMAL
        Mean: 30
        Standard deviation: 3


Manufacturing Cell -- C
   Cell name: MC1
   Items required to make the part C:
   Number of item types required: 1
     Name of item #1: I
     Number of item #1: 2
   Setup time for a cart of parts: CONSTANT
        Constant: 0
   Manufacturing time for a part:  NORMAL
        Mean: 30
        Standard deviation: 3
```

GPSS Code of Example Model Model-8-Stations


```
100 **************************************************************
110 *
120 *      This is a 8-stations model.           3/15/89
130 *
140 *-------------------------------------------------------
150 *
160 * This is a GPSS program automatically created from
170 *
180 * AMPS - (Automatic Manufacturing Programming System)
190 *
200 * developed at
210 *
220 * The University of Alabama in Huntsville, 1988.
230 *
240 **************************************************************
250           SIMULATE
260 UNIFM    FUNCTION     RN1,C2
0,0/1,1
270 SNORM    FUNCTION     RN1,C25
0,-5/.00003,-4/.00135,-3/.00621,-2.5/.02275,-2
.06681,-1.5/.11507,-1.2/.15866,-1/.21186,-.8/.27425,-.6
.34458,-.4/.42074,-.2/.5,0/.57926,.2/.65442,.4
.72575,.6/.78814,.8/.84134,1/.88493,1.2/.93319,1.5
.97725,2/.99379,2.5/.99865,3/.99997,4/1,5
280 XPDIS    FUNCTION     RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7.0/.9998,8.0
290 *** MAIN PARAMETERS    ***
300 PER11,    FUNCTION     RN1,D2
1,CHECK/1,PASS
310 PER12,    FUNCTION     RN1,D2
0.2,REPAIR/1,PASS
320 PER13,    FUNCTION     RN1,D2
0.5,SCRAP/1,PASS
330 TIME1    FVARIABLE    6+8*FN$UNIFM
340 TIME2    FVARIABLE    0
350 TIME3    FVARIABLE    10+1*FN$SNORM
360 TIME4    FVARIABLE    8+4*FN$UNIFM
370 TIME5    FVARIABLE    5+1*FN$SNORM
380 TIME6    FVARIABLE    30+3*FN$SNORM
390 TIME7    FVARIABLE    300*FN$XPDIS
400 TIME8    FVARIABLE    300+10*FN$SNORM
410 TIME9    FVARIABLE    50+5*FN$SNORM
420 TIME10    FVARIABLE    400+10*FN$SNORM
430 TIME11    FVARIABLE    100+5*FN$SNORM
440 TIME12    FVARIABLE    75+5*FN$SNORM
450 *** DEFINITION OF MATRIX  ***
460 PART     MATRIX       ,12,2
470 STIME    MATRIX       ,8,1
480 STAM     MATRIX       ,8,6
490 ITIME    MATRIX       ,1,2
500 ISTA     MATRIX       ,1,3
510 IPERC    MATRIX       ,1,3
520 MTIME    MATRIX       ,7,1
530 FGIG     MATRIX       ,7,1
540 SCART    MATRIX       ,7,1
550 CART     MATRIX       ,7,1
560 CTIME    MATRIX       ,7,2
570 CELL     MATRIX       ,7,1
580 CSIZE    MATRIX       ,7,1
590 ITEM     MATRIX       ,7,5
600 *** CAPACITY OF PART & CART COUNTERS ***
610 PA_A     STORAGE      2000
620 PA_F     STORAGE      4
630 CART_F   STORAGE      8
640 SCART_F  STORAGE      8
650 PA_H     STORAGE      4
660 CART_H   STORAGE      8
670 SCART_H  STORAGE      8
680 PA_E     STORAGE      4
690 CART_E   STORAGE      8
700 SCART_E  STORAGE      8
```


ORIGINAL PAGE IS
OF POOR QUALITY

```
710 PA_I     STORAGE     4
720 CART_I   STORAGE     8
730 SCART_I  STORAGE     8
740 PA_G     STORAGE     4
750 CART_G   STORAGE     8
760 SCART_G  STORAGE     8
770 PA_C     STORAGE     4
780 CART_C   STORAGE     8
790 SCART_C  STORAGE     8
800 PA_B     STORAGE     2000
810 PA_D     STORAGE     2000
820 PA_J     STORAGE     10000
830 PA_K     STORAGE     10000
840 PA_L     STORAGE     10000
850 ***************************************
860 *        INITIAL VALUES               *
870 ***************************************
880          GENERATE    ,,,1
890 *** PART ID  ***
900          MSAVEVALUE  PART,10,1,$PA_A  ;the id of part A is 10
910          MSAVEVALUE  PART,2,1,$PA_F   ;the id of part F is 2
920          MSAVEVALUE  PART,5,1,$PA_H   ;the id of part H is 5
930          MSAVEVALUE  PART,3,1,$PA_E   ;the id of part E is 3
940          MSAVEVALUE  PART,4,1,$PA_I   ;the id of part I is 4
950          MSAVEVALUE  PART,7,1,$PA_G   ;the id of part G is 7
960          MSAVEVALUE  PART,6,1,$PA_C   ;the id of part C is 6
970          MSAVEVALUE  PART,11,1,$PA_B  ;the id of part B is 11
980          MSAVEVALUE  PART,12,1,$PA_D  ;the id of part D is 12
990          MSAVEVALUE  PART,9,1,$PA_J   ;the id of part J is 9
1000         MSAVEVALUE  PART,8,1,$PA_K   ;the id of part K is 8
1010         MSAVEVALUE  PART,1,1,$PA_L   ;the id of part L is 1
1020 *** THE SIZE OF EACH CART  ***
1030         MSAVEVALUE  CSIZE,2,1,4
1040         MSAVEVALUE  CSIZE,5,1,4
1050         MSAVEVALUE  CSIZE,3,1,4
1060         MSAVEVALUE  CSIZE,4,1,4
1070         MSAVEVALUE  CSIZE,7,1,4
1080         MSAVEVALUE  CSIZE,6,1,4
1090 *** INITIAL INVENTORY LEVEL AT EACH STOCK POINT ***
1100         ENTER       CART_F,4
1110         ENTER       SCART_F,4
1120         ENTER       CART_H,4
1130         ENTER       SCART_H,4
1140         ENTER       CART_E,4
1150         ENTER       SCART_E,4
1160         ENTER       CART_I,4
1170         ENTER       SCART_I,4
1180         ENTER       CART_G,4
1190         ENTER       SCART_G,4
1200         ENTER       CART_C,4
1210         ENTER       SCART_C,4
1220         ENTER       PA_B,120
1230         ENTER       PA_D,120
1240         ENTER       PA_J,10000
1250         ENTER       PA_K,10000
1260         ENTER       PA_L,10000
1270 *** MAKE ONE CART READY AT EACH DEMAND STOCK POINT  ***
1280         LEAVE       CART_F,1
1290         ENTER       PA_F,MX$CSIZE(2,1)
1300         LEAVE       CART_H,1
1310         ENTER       PA_H,MX$CSIZE(5,1)
1320         LEAVE       CART_E,1
1330         ENTER       PA_E,MX$CSIZE(3,1)
1340         LEAVE       CART_I,1
1350         ENTER       PA_I,MX$CSIZE(4,1)
1360         LEAVE       CART_G,1
1370         ENTER       PA_G,MX$CSIZE(7,1)
1380         LEAVE       CART_C,1
1390         ENTER       PA_C,MX$CSIZE(6,1)
1400 *** ITEMS REQUIRED TO MAKE EACH PART  ***
1410         MSAVEVALUE  ITEM,2,1,1   ;part F requires 1 part type(s).
1420         MSAVEVALUE  ITEM,2,2,1   ;   part L.
1430         MSAVEVALUE  ITEM,2,3,2   ;   2 unit(s).
```

```
         MSAVEVALUE   ITEM,5,1,1   ;part H requires 1 part type(s).
         MSAVEVALUE   ITEM,5,2,4   ;   part I.
         MSAVEVALUE   ITEM,5,3,1   ;   1 unit(s).
         MSAVEVALUE   ITEM,3,1,2   ;part E requires 2 part type(s).
         MSAVEVALUE   ITEM,3,2,9   ;   part J.
         MSAVEVALUE   ITEM,3,3,2   ;   2 unit(s).
         MSAVEVALUE   ITEM,3,4,8   ;   part K.
         MSAVEVALUE   ITEM,3,5,1   ;   1 unit(s).
         MSAVEVALUE   ITEM,4,1,2   ;part I requires 2 part type(s).
         MSAVEVALUE   ITEM,4,2,9   ;   part J.
         MSAVEVALUE   ITEM,4,3,2   ;   2 unit(s).
         MSAVEVALUE   ITEM,4,4,8   ;   part K.
         MSAVEVALUE   ITEM,4,5,1   ;   1 unit(s).
         MSAVEVALUE   ITEM,7,1,1   ;part G requires 1 part type(s).
         MSAVEVALUE   ITEM,7,2,8   ;   part K.
         MSAVEVALUE   ITEM,7,3,2   ;   2 unit(s).
         MSAVEVALUE   ITEM,6,1,1   ;part C requires 1 part type(s).
         MSAVEVALUE   ITEM,6,2,4   ;   part I.
         MSAVEVALUE   ITEM,6,3,2   ;   2 unit(s).
*** STATION SERVICE TIME   ***
         MSAVEVALUE   STIME,6,1,$TIME11   ;time of TASK6 is TIME11
         MSAVEVALUE   STIME,2,1,$TIME8   ;time of TASK2 is TIME8
         MSAVEVALUE   STIME,5,1,$TIME11   ;time of ASSY5 is TIME11
         MSAVEVALUE   STIME,1,1,$TIME8   ;time of ASSY1 is TIME8
         MSAVEVALUE   STIME,3,1,$TIME8   ;time of ASSY3 is TIME8
         MSAVEVALUE   STIME,8,1,$TIME12   ;time of ASSY8 is TIME12
         MSAVEVALUE   STIME,7,1,$TIME12   ;time of ASSY7 is TIME12
*** TIME TO MOVE A CART BETWEEN SUPPLY AND DEMAND POINTS  ***
         MSAVEVALUE   MTIME,2,1,$TIME1  ;moving time of a cart of part F is TIME1
         MSAVEVALUE   MTIME,5,1,$TIME4  ;moving time of a cart of part H is TIME4
         MSAVEVALUE   MTIME,3,1,$TIME4  ;moving time of a cart of part E is TIME4
         MSAVEVALUE   MTIME,4,1,$TIME4  ;moving time of a cart of part I is TIME4
         MSAVEVALUE   MTIME,7,1,$TIME1  ;moving time of a cart of part G is TIME1
         MSAVEVALUE   MTIME,6,1,$TIME4  ;moving time of a cart of part C is TIME4
*** SETUP TIME FOR A CART OF PARTS AND  ***
*** MANUFACTURING TIME FOR A PART ***
         MSAVEVALUE   CTIME,2,1,$TIME2 ;setup time for a cart of part F is TIME2
         MSAVEVALUE   CTIME,2,2,$TIME3 ;manufacturing time for part F is TIME3
         MSAVEVALUE   CTIME,5,1,$TIME2 ;setup time for a cart of part H is TIME2
         MSAVEVALUE   CTIME,5,2,$TIME6 ;manufacturing time for part H is TIME6
         MSAVEVALUE   CTIME,3,1,$TIME2 ;setup time for a cart of part E is TIME2
         MSAVEVALUE   CTIME,3,2,$TIME3 ;manufacturing time for part E is TIME3
         MSAVEVALUE   CTIME,4,1,$TIME2 ;setup time for a cart of part I is TIME2
         MSAVEVALUE   CTIME,4,2,$TIME5 ;manufacturing time for part I is TIME5
         MSAVEVALUE   CTIME,7,1,$TIME2 ;setup time for a cart of part G is TIME2
         MSAVEVALUE   CTIME,7,2,$TIME3 ;manufacturing time for part G is TIME3
         MSAVEVALUE   CTIME,6,1,$TIME2 ;setup time for a cart of part C is TIME2
         MSAVEVALUE   CTIME,6,2,$TIME6 ;manufacturing time for part C is TIME6
*** CELL WHERE EACH PART IS MADE  ***
         MSAVEVALUE   CELL,2,1,$MC2  ;part F is made on machine MC2
         MSAVEVALUE   CELL,5,1,$MC1  ;part H is made on machine MC1
         MSAVEVALUE   CELL,3,1,$MC2  ;part E is made on machine MC2
         MSAVEVALUE   CELL,4,1,$MC2  ;part I is made on machine MC2
         MSAVEVALUE   CELL,7,1,$MC2  ;part G is made on machine MC2
         MSAVEVALUE   CELL,6,1,$MC1  ;part C is made on machine MC1
*** NAME OF EACH STATION   ***
         MSAVEVALUE   STAN,4,1,$INSP4  ;the id of station INSP4 is 4
         MSAVEVALUE   STAN,6,1,$TASK6  ;the id of station TASK6 is 6
         MSAVEVALUE   STAN,2,1,$TASK2  ;the id of station TASK2 is 2
         MSAVEVALUE   STAN,5,1,$ASSY5  ;the id of station ASSY5 is 5
         MSAVEVALUE   STAN,1,1,$ASSY1  ;the id of station ASSY1 is 1
         MSAVEVALUE   STAN,3,1,$ASSY3  ;the id of station ASSY3 is 3
         MSAVEVALUE   STAN,8,1,$ASSY8  ;the id of station ASSY8 is 8
         MSAVEVALUE   STAN,7,1,$ASSY7  ;the id of station ASSY7 is 7
*** INSPECTION STATION INDEX   ***
         MSAVEVALUE   STAN,4,2,1  ;the index of inspection station INSP4 is 1
*** INSPECTION STATION   ***
         MSAVEVALUE   IPERC,1,1,$PER11  ;inpsection rate of INSP4 is 1
         MSAVEVALUE   IPERC,1,2,$PER12  ;repair rate of INSP4 is 0.2
         MSAVEVALUE   IPERC,1,3,$PER13  ;scrap rate of INSP4 is 0.5
         MSAVEVALUE   ISTA,1,1,$INSPECTOR  ;inspector of INSP4 is INSPECTOR
         MSAVEVALUE   ISTA,1,2,$REPAIRMAN  ;repairman of INSP4 is REPAIRMAN
         MSAVEVALUE   ISTA,1,3,$SCRAP4  ;scrapped items of INSP4 are sent to SCRAP4
```

```
2170          MSAVEVALUE   ITIME,1,1,$TIME9  ;inspection time of INSP4 is TIME9
2180          MSAVEVALUE   ITIME,1,2,$TIME10  ;repair time of INSP4 is TIME10
2190 *** PART (ID) REQUIRED AT EACH STATION   ***
2200          MSAVEVALUE   STAN,5,2,1     ;station ASSY5 requires 1 part type(s).
2210          MSAVEVALUE   STAN,5,3,3     ;    part E.
2220          MSAVEVALUE   STAN,5,4,2     ;    2 unit(s).
2230          MSAVEVALUE   STAN,1,2,2     ;station ASSY1 requires 2 part type(s).
2240          MSAVEVALUE   STAN,1,3,11    ;    part B.
2250          MSAVEVALUE   STAN,1,4,3     ;    3 unit(s).
2260          MSAVEVALUE   STAN,1,5,6     ;    part C.
2270          MSAVEVALUE   STAN,1,6,2     ;    2 unit(s).
2280          MSAVEVALUE   STAN,3,2,1     ;station ASSY3 requires 1 part type(s).
2290          MSAVEVALUE   STAN,3,3,12    ;    part D.
2300          MSAVEVALUE   STAN,3,4,4     ;    4 unit(s).
2310          MSAVEVALUE   STAN,8,2,1     ;station ASSY8 requires 1 part type(s).
2320          MSAVEVALUE   STAN,8,3,5     ;    part H.
2330          MSAVEVALUE   STAN,8,4,1     ;    1 unit(s).
2340          MSAVEVALUE   STAN,7,2,2     ;station ASSY7 requires 2 part type(s).
2350          MSAVEVALUE   STAN,7,3,2     ;    part F.
2360          MSAVEVALUE   STAN,7,4,2     ;    2 unit(s).
2370          MSAVEVALUE   STAN,7,5,7     ;    part G.
2380          MSAVEVALUE   STAN,7,6,1     ;    1 unit(s).
2390 *** SUPPLY SYSTEM OF EACH PART  ***
2400          MSAVEVALUE   PART,10,2,1  ;part A is in push mode
2410          MSAVEVALUE   PART,2,2,0  ;part F is in pull mode
2420          MSAVEVALUE   PART,5,2,0  ;part H is in pull mode
2430          MSAVEVALUE   PART,3,2,0  ;part E is in pull mode
2440          MSAVEVALUE   PART,4,2,0  ;part I is in pull mode
2450          MSAVEVALUE   PART,7,2,0  ;part G is in pull mode
2460          MSAVEVALUE   PART,6,2,0  ;part C is in pull mode
2470          MSAVEVALUE   PART,11,2,1  ;part B is in push mode
2480          MSAVEVALUE   PART,12,2,1  ;part D is in push mode
2490          MSAVEVALUE   PART,9,2,1  ;part J is ordered from outside
2500          MSAVEVALUE   PART,8,2,1  ;part K is ordered from outside
2510          MSAVEVALUE   PART,1,2,1  ;part L is ordered from outside
2520 *** CART COUNTER AT EACH DESTINATION  ***
2530          MSAVEVALUE   CART,2,1,$CART_F
2540          MSAVEVALUE   CART,5,1,$CART_H
2550          MSAVEVALUE   CART,3,1,$CART_E
2560          MSAVEVALUE   CART,4,1,$CART_I
2570          MSAVEVALUE   CART,7,1,$CART_G
2580          MSAVEVALUE   CART,6,1,$CART_C
2590 *** CART COUNTER AT SOURCE  ***
2600          MSAVEVALUE   SCART,2,1,$SCART_F
2610          MSAVEVALUE   SCART,5,1,$SCART_H
2620          MSAVEVALUE   SCART,3,1,$SCART_E
2630          MSAVEVALUE   SCART,4,1,$SCART_I
2640          MSAVEVALUE   SCART,7,1,$SCART_G
2650          MSAVEVALUE   SCART,6,1,$SCART_C
2660 *** WHIRLYGIGS TO MOVE PARTS  ***
2670          MSAVEVALUE   FGIG,2,1,$TRUCK2  ;part F is transported by TRUCK2
2680          MSAVEVALUE   FGIG,5,1,$TRUCK3  ;part H is transported by TRUCK3
2690          MSAVEVALUE   FGIG,3,1,$TRUCK1  ;part E is transported by TRUCK1
2700          MSAVEVALUE   FGIG,4,1,$TRUCK3  ;part I is transported by TRUCK3
2710          MSAVEVALUE   FGIG,7,1,$TRUCK2  ;part G is transported by TRUCK2
2720          MSAVEVALUE   FGIG,6,1,$TRUCK1  ;part C is transported by TRUCK1
2730          TERMINATE
2740 ***************************************
2750 *      ASSEMBLY LINE Y
2760 ***************************************
2770          GENERATE     V$TIME11
2780          ASSIGN       2,5               ;station 5 is ASSY5
2790          TRANSFER     SBR,ASM,RTRN1
2800          ASSIGN       2,6               ;station 6 is TASK6
2810          TRANSFER     SBR,TASK,RTRN1
2820          ENTER        PA_8,1
2830          TERMINATE
2840 ***************************************
2850 *      ASSEMBLY LINE X
2860 ***************************************
2870          GENERATE     V$TIME7
2880          ASSIGN       2,1               ;station 1 is ASSY1
2890          TRANSFER     SBR,ASM,RTRN1
```

```
2900          ASSIGN      2,2               ;station 2 is TASK2
2910          TRANSFER    SBR,TASK,RTRN1
2920          ASSIGN      2,3               ;station 3 is ASSY3
2930          TRANSFER    SBR,ASM,RTRN1
2940          ASSIGN      2,4               ;station 4 is INSP4
2950          TRANSFER    SBR,INSP,RTRN1
2960          ENTER       PA_A,1
2970          TERMINATE
2980 ***************************************
2990 *       ASSEMBLY LINE 2
3000 ***************************************
3010          GENERATE    V$TIME12
3020          ASSIGN      2,7               ;station 7 is ASSY7
3030          TRANSFER    SBR,ASM,RTRN1
3040          ASSIGN      2,8               ;station 8 is ASSY8
3050          TRANSFER    SBR,ASM,RTRN1
3060          ENTER       PA_B,1
3070          TERMINATE
3080 ***************************************************************
3090 *                    INSPECTION STATION                       *
3100 ***************************************************************
3110 INSP    ASSIGN      3,MX$STAN(P2,2)  ;insp. station
3120         ASSIGN      4,MX$IPERC(P3,1) ;inspection rate
3130         ASSIGN      5,MX$ITIME(P3,1) ;time for inspection
3140         ASSIGN      6,MX$ITIME(P3,2) ;time for repair
3150         QUEUE       MX$STAN(P2,1)    ;count entering parts
3160         DEPART      MX$STAN(P2,1)    ;and pass
3170         TRANSFER    ,FN*4            ;to be checked?
3180 CHECK   QUEUE       MX$ISTA(P3,1)    ;wait on the insp. facility
3190         SEIZE       MX$ISTA(P3,1)    ;seize the insp. facility
3200         DEPART      MX$ISTA(P3,1)    ;leave the insp. queue
3210         ADVANCE     V*5              ;inspecting part
3220         RELEASE     MX$ISTA(P3,1)    ;release the insp. facility
3230         ASSIGN      4,MX$IPERC(P3,2) ;repair rate
3240         TRANSFER    ,FN*4            ;to be repaired?
3250 REPAIR  QUEUE       MX$ISTA(P3,2)    ;wait on repairing facility
3260         SEIZE       MX$ISTA(P3,2)    ;seize the repairing facility
3270         DEPART      MX$ISTA(P3,2)    ;leave the queue
3280         ADVANCE     V*6              ;repairing parts
3290         RELEASE     MX$ISTA(P3,2)    ;release the repairing facility
3300         ASSIGN      4,MX$IPERC(P3,3) ;scrap rate
3310         TRANSFER    ,FN*4            ;is part scrapped?
3320 PASS    TRANSFER    P,RTRN1,1        ;finish inspection and return
3330 SCRAP   QUEUE       MX$ISTA(P3,3)    ;count scrapped parts
3340         DEPART      MX$ISTA(P3,3)    ;and pass
3350         TERMINATE                    ;terminate the transaction
3360 ***************************************************************
3370 *                    TASK STATION                             *
3380 ***************************************************************
3390 TASK     ASSIGN     3,MX$STAN(P2,1)  ;name of the task station
3400          ASSIGN     6,MX$STIME(P2,1) ;time for operation
3410          QUEUE      P3               ;wait on the facility
3420          SEIZE      P3               ;seize the facility
3430          DEPART     P3               ;leave the queue
3440          ADVANCE    V*6              ;perform operation
3450          RELEASE    P3               ;release the facility
3460          TRANSFER   P,RTRN1,1        ;finish and return
3470 ***************************************************************
3480 *                    ASSEMBLY STATION                         *
3490 ***************************************************************
3500 ASM      ASSIGN     3,MX$STAN(P2,1)   ;name of the station
3510          ASSIGN     7,MX$STAN(p2,2)   ;no. of part types required
3520          ASSIGN     6,MX$STIME(P2,1)  ;assembly time
3530          ASSIGN     8,1               ;index for part type
3540          ASSIGN     9,2               ;index for units of part
3550          QUEUE      P3                ;wait on the facility
3560 PRO      ASSIGN     8+,2              ;point to next type
3570          ASSIGN     9+,2              ;point to next no. of units
3580          ASSIGN     5,MX$STAN(P2,P8)  ;part id
3590          ASSIGN     10,MX$PART(P5,1)  ;name of part
3600          ASSIGN     20,MX$STAN(P2,P9) ;number of units of part
3610          QUEUE      P10               ;wait to get parts
3620          TRANSFER   SBR,TAKEP,RTRN2   ;get parts
```

```
3630          DEPART    P10                   ;leave the queue
3640          LOOP      7,PAO                 ;received all parts req'd?
3650          SEIZE     P3                    ;seize the facility
3660          DEPART    P3                    ;leave the queue
3670          ADVANCE   V*6                   ;assembling parts
3680          RELEASE   P3                    ;release the facility
3690          TRANSFER  P,RTRN1,1             ;return to the calling statement
3700 ******************************************************************
3710 *                      INVENTORY CONTROL                         *
3720 ******************************************************************
3730 TAKEP  TEST E    MX$PART(P5,2),1,PULL ;pull or push mode?
3740 PUSH   TEST GE   S*10,P20              ;available >= req'd
3750        LEAVE     *10,P20               ;take units requested
3760        TRANSFER  P,RTRN2,1             ;return
3770 PULL   ASSIGN    30,MX$CART(P5,1)      ;name of cart for the part
3780        TEST GE   S*10,P20,NEEDC        ;available >= requested?
3790 MINUSP LEAVE     *10,P20               ;take units requested
3800        SPLIT     1,USEP                ;signal use of parts
3810        TRANSFER  P,RTRN2,1             ;return
3820 NEEDC  ASSIGN    20-,S*10              ;requested - available = unfilled
3830        LEAVE     *10,S*10              ;take remaining
3840        SPLIT     1,USEP                ;signal use of parts
3850        TEST GE   S*30,1                ;any full cart available?
3860        LEAVE     *30                   ;take 1 cart
3870        ENTER     *10,MX$CSIZE(P5,1)    ;make parts available
3880        TEST GE   S*10,P20,NEEDC        ;available >= unfilled?
3890        LEAVE     *10,P20               ;take unfilled units
3900        SPLIT     1,USEP                ;signal use of parts
3910        TRANSFER  P,RTRN2,1             ;return
3920 USEP   TEST G    S*10,0,EMPTYC         ;check if cart empty
3930        TERMINATE                       ;terminate the transaction
3940 EMPTYC SPLIT     1,ORDER1              ;signal to order
3950        TEST GE   S*30,1                ;any full cart available
3960        LEAVE     *30,1                 ;take 1 cart
3970        ENTER     *10,MX$CSIZE(P5,1)    ;make parts available
3980        TERMINATE                       ;terminate the transaction
3990 ORDER1 ASSIGN    26,MX$FCIG(P5,1)      ;name of the vehicle
4000        ASSIGN    36,MX$MTIME(P5,1)     ;moving time
4010        QUEUE     P26                   ;wait on the vehicle
4020        SEIZE     P26                   ;seize the vehicle
4030        DEPART    P26                   ;leave the queue
4040        ADVANCE   V*36                  ;moving the empty cart
4050        RELEASE   P26                   ;release the vehicle
4060        SPLIT     1,GET1F               ;signal to get a full cart
4070        ASSIGN    12,P5                 ;assign part id
4080        ASSIGN    15,MX$SCART(P5,1)     ;assign name of the cart
4090 GET1C  TRANSFER  SBR,MFG,RTRN3         ;order a cart of parts
4100        ENTER     *15,1                 ;receive a cart
4110        TERMINATE                       ;terminate the transaction
4120 GET1F  ASSIGN    31,MX$SCART(P5,1)     ;index for a full cart
4130        QUEUE     P31                   ;wait on a full cart
4140        TEST GE   S*31,1                ;whether a full cart is available
4150        LEAVE     *31,1                 ;take 1 cart
4160        DEPART    P31                   ;leave the queue
4170 SEND1F QUEUE     P26                   ;wait on the vehicle
4180        SEIZE     P26                   ;seize the vehicle
4190        DEPART    P26                   ;leave the queue
4200        ADVANCE   V*36                  ;moving the full cart
4210        RELEASE   P26                   ;release the vehicle
4220        ENTER     *30,1                 ;increase inventory by 1 cart
4230        TERMINATE                       ;terminate the transaction
4240 ******************************************************************
4250 *                      MANUFACTURING CELL                        *
4260 ******************************************************************
4270 MFG    ASSIGN    13,MX$CELL(P12,1)     ;name of manuf. cell
4280        ASSIGN    14,MX$CTIME(P12,1)    ;setup time of a cart
4290        ASSIGN    16,MX$CTIME(P12,2)    ;manuf. time of a part
4300        QUEUE     P13                   ;wait on the facility
4310        ASSIGN    7,MX$CSIZE(P12,1)     ;no. of parts in a cart
4320 CARTO  ASSIGN    17,MX$ITEM(P12,1)     ;no. of items types req'd
4330        ASSIGN    8,0                   ;index for item type
4340        ASSIGN    9,1                   ;index for units of item
4350 PARTO  ASSIGN    8+,2                  ;point to next type
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
4360            ASSIGN      9+,2                  ;point to next no. of units
4370            ASSIGN      5,MX$ITEM(P12,P8)     ;id of the item req'd
4380            ASSIGN      10,MX$PART(P5,1)      ;name of the item
4390            ASSIGN      20,MX$ITEM(P12,P9)    ;units of the item req'd
4400            QUEUE       P10                   ;wait on the items
4410            TRANSFER    SBR,TAKEP,RTRN2       ;get items
4420            DEPART      P10                   ;leave the queue
4430            LOOP        17,PARTQ              ;loop for next item type req'd
4440            LOOP        7,CARTQ               ;loop for next part to be made
4450 FAC        SEIZE       P13                   ;seize the facility
4460            DEPART      P13                   ;leave the queue
4470            ADVANCE     V*14                  ;set up facility
4480            ADVANCE     V$MTIME               ;manufacturing
4490 MTIME      FVARIABLE   V*16*MX$CSIZE(P12,1)  ;manufacturing time
4500            RELEASE     P13                   ;release the facility
4510            TRANSFER    P,RTRN3,1             ;manufacturing complete
```

ORIGINAL PAGE IS
OF POOR QUALITY