This is to certify that the

thesis entitled

ASSEMBLY LINE MOLD SCHEDULING

presented by

KEVIN DELAND MARKLE

has been accepted towards fulfillment
of the requirements for

M.S. degree in COMPUTER SCIENCE

[illegible]
Major professor

Date 4/12/78

O-7639

ASSEMBLY LINE MOLD SCHEDULING

by

Kevin D. Markle

A Thesis

submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science
1977

Thesis Advisor: Dr. Anil Jain

ABSTRACT

ASSEMBLY LINE MOLD SCHEDULING

BY

KEVIN D. MARKLE

Assembly Line Mold Scheduling is a mathematical model which will generate an efficient production schedule of molds assigned to a specific foam assembly line. The objective of this model is to minimize product costs (mainly mold set-up charges), minimize inventory holding costs, and to eliminate back-order production costs.

The model will generate a new mold configuration each week. More than the minimum number of set-ups necessary to meet demand may be needed to generate a feasible sequence of molds on the foam line. If excess capacity is available, the model should look forward to following weeks' demands and load molds that tend to minimize future problems or bottlenecks. The model should also insure that demand is met for each mold at the end of each shipping day. The model must tell how many of each mold type to have mounted on the assembly line, and the exact position of each mold on the 181 fixed carriers so that the plant's assembly line sequencing rules are obeyed.

ACKNOWLEDGEMENTS

After completing my class requirements in the Computer Science curriculum at Michigan State University, I returned from an educational leave of absence to General Motors Manufacturing Development. There I was assigned to the Manufacturing Operations Systems Department on a project entitled: Assembly Line Mold Scheduling. This investigation and thesis is being submitted as part of the requirements for obtâining a Master of Science degree in Computer Science from Michigan State University.

I would like to extend my appreciation to the following people: Mr. James Caie Jr., Senior Project Engineer and Plant Advisor and Mr. Robert Harder, Staff Development Engineer, Manufacturing Operations Systems of Manufacturing Development, for their suggestions and advice concerning the success of this project. I would also like to thank Dr. Philip Carter, Professor of Management, School of Business and Dr. Anil Jain, Assistant Professor of Computer Science, School of Engineering, for their assistance as Faculty Advisors at Michigan State University.

ASSEMBLY LINE MOLD SCHEDULING

TABLE OF CONTENTS

ASSEMBLY LINE MOLD SCHEDULING

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

I. INTRODUCTION

General Motors Corporation is interested in creating a mathematical model which will generate an efficient production schedule of molds assigned to a specific foam assembly line. The objective of this model is to minimize production costs (mainly mold set-up charges), minimize inventory holding costs, and to eliminate back-order production costs. The foam line in question consists of 181 fixed carriers connected together to form a large circular chain which rotates past operation points. Carriers are defined as the portion of the assembly line where a mold can be attached. A mold must be mounted in a frame prior to attaching it to a carrier. Setup, therefore, involves framing a mold if it is not currently framed and bolting this mold-frame assembly to the carrier on the assembly line. Since all frames are mounted on molds, the model must decide which mold to unframe to make a frame available for mounting on another mold.

The normal operation of the foam line is 120 hours per week (i.e., 8 hours/shift * 3 shifts / day * 5 days/week). Saturday operation will occur only if it is absolutely necessary to meet shipping requirements. Since the whole line must be stopped to change a mold, major set-up activities for the week will occur on Saturday if production is not scheduled, or on Sunday otherwise. It should be noted that management has indicated that carrier positions on the assembly line must not remain empty when the line is running. Therefore, it may be necessary to produce parts on the foam line that are not currently needed.

There are twenty-five different mold types currently assigned to the foam line. A mold type is identified by a two character code, and is categorized as either a "large" or a "small" mold. Most mold types have more than one physical mold available for production at any specific time. A mold type consisting of a double cavity will produce two pieces at a time. Finally, since scrap rates vary by mold type, each mold type has a unique production rate (i.e., pieces/time/carrier location).

The scheduling project was undertaken to investigate, analyze, design, evaluate and implement an efficient production schedule of molds assigned to a foam assembly line. It is essential that the model minimize production costs and inventory holding cost, and eliminate back-order production costs. The model must tell how many of each mold type should be mounted on the assembly line, and the exact position of each mold on the 181 fixed carriers so that the plant's assembly line sequencing rules are obeyed. It must also determine how many production shifts are necessary while staying within the plant's warehouse limitations and production shipping requirements. This report summarizes the results and findings of this investigation.

After the introduction, Chapter 2 discusses the background of the mold scheduling problem. It briefly explains the assembly line operations required to produce the foam seats. Finally, it examines the sequencing rules and costs of the model, and describes the current operating procedures and problems of the plant.

Chapter 3 discusses the structure and ideas behind the assembly line scheduling model. It describes in great detail the objectives and constraints of the mixed integer program, relating the plant's problems to the mathematical model. Finally, the scheduling and sequencing algorithm is explained, in effect, satisfying the plant's scheduling objectives.

The next chapter discusses the scheduling system overview, which describes the various subroutines of the foam model. Each of the various subroutines are examined in greater detail. Chapter 4 also examines a typical assembly line scheduling example. It explains the required inputs and flow of the model through the final sequencing line-up and mold changes.

Chapter 5 discusses an experimental run made by the plant to determine the assembly line sequence of molds.

It is easy to see from these results that an assembly line is more difficult to schedule than one would believe.

The final chapter examines the conclusions of the foam model with respect to the various advantages and disadvantages of an optimization model. It discusses the constraints which were satisfied versus the total set-up and inventory costs of the solution.

II. DEFINITION OF THE PROBLEM

Few people really understand how difficult it is to consistently schedule any type of assembly line. There are rules and constraints which must be followed in the manufacturing plant. The assembly line also consists of people working in a plant environment. There are labor problems, material shortages, machine break downs and production to ship, and little time to produce the required parts. In short, for an outsider, pure chaos reigns in the plant.

This chapter discusses the operations and background of the foam assembly line. It briefly describes the assembly line operations required to produce foam seats. It assumes that the workers know their specific job requirements, and that the material is always available. However, this is not always true in the plant situation.

The definition of the problem also considers the plant's constraints, the sequencing rules which "must" and "should" be obeyed. The plant has assigned labor costs to the various functions, such as changing a mold (set-up costs), storing a part in inventory (inventory holding costs), and a cost associated with framing a required mold (framing costs). Each of these costs will be considered in the mixed integer program to determine the optimal assembly line configuration.

Finally, there is a discussion of the current operating procedures and problems which the plant faces every day. There is an example of how the assembly line is currently scheduled and the difficulties that occur in changing any mold on the assembly line.

2.1 ASSEMBLY LINE OPERATIONS

The following operations summarize the foam assembly line production procedures:

1. Loading bolster wires and border wires into the mold. Wire loading work on some mold types requires that operators work on a special platform. It should also be noted that each mold type has a specific wire loading difficulty factor associated with it.

2. Pour foam into the mold with an automatic "gun".

3. Bake the foam in an oven.

4. Cure foam. Several mold types require taping after post cure.

5. Pack the foam parts in 5491 standarized (identical) containers. The packing area consists of thirty upender container locations, fourteen on one side of a moving conveyor and sixteen on the other side. A specific container stores only one mold type, but certain mold types can have more than one container location associated with it. The assignment of mold types to specific container locations is normally made at the same time as a major set-up takes place. A packing operator is responsible for a group of adjacent container locations co-located on one side of the moving conveyor. Normally, there are five or six packers working at the same time.

6. When two containers (called a lift) are full, they are both taken and stored in the warehouse. The warehouse can hold a maximum of 900 containers at one time. If the warehouse is filled, containers may be routed immediately into rail cars for storage. Ten containers fit into one rail car, and the plant expects to ship thirty-six rail cars of foam parts five days a week. There are no rail shipments made on Saturday or Sunday.

2.2 SEQUENCING RULES AND COSTS

There are several sequencing constraints that must be adhered to for a schedule to be feasible.

1. A "large" frame mold MUST have a "small" frame mold before and after it.

2. Carrier #1 MUST be the same as carrier #180, and carrier #2 MUST be the same as carrier #181. This insures that the "gun" is cleaned out at least once per cycle.

3. Molds with letter codes "AN" and "MM" should be lined up in a group because they require platform work and the wire loading operator should only move to the platform once per cycle.

4. The platform is only 40 feet long allowing limited room for float. Therefore, any "AN" or "MM" molds should have some other style before and after it.

5. The biggest problem on the foam line is the wire loading difficulty. The total number of wire loading points for any three consecutive molds should not exceed 27.

6. Parts requiring tape after post cure should be spread out so that one operator can handle the work.

7. The mold container sequence in the packing area should be consistent with the mold sequence on the foam line to insure that a particular packing operator does

not get overloaded. For instance, if a packing operator is responsible for packing several different mold types, and if these mold types are all sequenced together, then the packing operator will probably be overloaded at certain times of the day.

Given the current assignment of molds to carriers on the foam line, the model to be developed must generate a new configuration of molds on carriers so that:

1. All the next week's demand is satisfied.

2. Set-up and inventory holding costs are minimized.

3. Mold sequencing rules #1 and #2 MUST be obeyed while rules #3-#7 should be obeyed if at all possible.

4. Warehousing constraints MUST be obeyed.

The model will generate a new mold configuration each week. It should be noted that more than the minimum number of set-ups necessary to meet demand may be needed to generate a feasible sequence of molds on the foam line. If excess capacity is available, the model should look forward to following weeks demands and load molds that tend to minimize future problems or bottlenecks. The model should also insure that demand is met for each mold at the end of each shipping day. We can assume that a shipping day's demand for a mold is one-fifth its week's demand. Figure 1 on the following page, gives a summary of the assembly line costs assigned by the plant. These are by no means all the costs associated with the production of foam parts, they are valuable in describing the major costs of a simplified mathematical model.

Figure 1. ASSEMBLY LINE MODEL COSTS

Cost of framing a mold	$75.00
Cost of removing a mold from a carrier and setting up a new framed mold on the same carrier.	$15.00
Inventory holding costs	10% of part value
Cost of having one upender assigned to a part.	$ 1.00

2.3 CURRENT OPERATING PROCEDURES AND PROBLEMS

The molds required for each part are currently determined by the following proportion:

$$\frac{\text{Total schedule for each part}}{\text{Total schedule for the line}} * 181 = \text{molds required}$$

These proportions are usually calculated once a month; the drawback here is that the proportions overstate molds required for high volume parts and under-estimate molds required for low volume parts. Production Control tries to perform the following calculations on a regular basis:

1. On Thursday, use the current mold line-up to estimate production for the next seven days.

2. Take Thursday's initial inventory plus the estimated production for seven days minus seven days of shipping requirements to get the forecasted inventory at the end of next week.

FORECASTED INVENTORY = INITIAL INVENTORY + ESTIMATED PRODUCTION - SHIPMENT

3. Add the Forecasted Inventory for each part on a line to get a Total Inventory. Divide by the Average Daily Production for that line to get an approximate number of days bank (safety stock) in the warehouse. If there is less than two days bank for the line, a decision is usually made to work overtime. If the Forecasted Inventory for any part number is negative or a very low figure, production staff is told to add molds. An equal

number of molds, however, must be removed. To make this decision, Production Control finds the parts with the highest inventory and removes molds of those part numbers.

It should be noted that the above calculations are tedious and time consuming. All information must be copied from four different sources and then the calculations performed. It takes between 4 to 6 hours per week to do this and sometimes it is not done each week as it should be. Also, it is easy to make a mistake in these calculations. Another major disadvantage is that seven days may not be enough lead time to prevent back orders on the seventh day. If major changes must be made or not enough molds are mounted, production staff may not be able to make the changes until the following week and back orders will result.

There are certainly other considerations which must be taken into account before a mold on the assembly line can be changed.

1. The whole line must be stopped to change one mold.

2. Due to the bulkiness of foam and fire hazards associated with it, both assembly plants and manufacturing plants are restricted from holding large inventories. (Typically a two day safety bank is held in foam parts.)

3. Fire laws prevent foam from being stacked more than three baskets high. This limits effective space utilization.

4. Fork lift trucks can only carry two baskets at one time so movement, labelling and storage has always been done on the basis of two baskets (called a lift). In other words, baskets are seldom stacked three high in the warehouse.

5. Four baskets may be stacked on top of each other in the inventory storage system because it has an intermediate sprinkling system.

6. If the schedule for one line is slightly greater than machine capacity, the warehouse may be overloaded if Saturday overtime is worked, because no shipments are made on weekends. Therefore, it is often necessary to accept shortages.

7. The line cannot run empty, so to prevent shortages on some parts it is possible to overload the warehouse with parts that are not needed.

8. Components are involved. Some foam parts require border and bolster wires, which are made in another division of the plant.

9. When production staff is told how many of each mold type to put on the line, they cannot always make the required changes. This often results because the molds needed may not be mounted in frames, or serious

sequencing constraints might be violated.

10. The mere addition of one mold may force the whole line to be rearranged. Hence, it may be cheaper to accept shortages than to rearrange the whole line.

11. It takes between 15 to 20 minutes to change a mold. The plant very seldom stops the line to perform this change; instead, they try to do it during breaks and lunch time. If more than 25 molds have to be changed, production control schedules these changes to be made on a weekend.

12. When molds are first put on the assembly line, they have to warm up for approximately three hours. If any foam is shot into the mold before it warms up, scrap is generated.

13. Scrap and downtime are irregular. Sometimes a line can run for several weeks with little downtime, and then in one week be plagued with a major breakdown lasting 8 to 10 hours.

2.4 SUMMARY

It is very hard to visualize or totally understand the operation of a foam manufacturing plant, without actually observing the assembly line. There are many problems which exist in the plant, and the past sections have explained the assembly line operations, the sequencing rules and costs, and the current operating procedures and problems.

There is certainly a need to create a computerized model, to remove the "trial and error" scheduling and mold sequencing procedures which now exist. The future chapters will consider and examine the assembly line scheduling model which was developed and implemented for the plant.

III. ASSEMBLY LINE SCHEDULING MODEL

The assembly line scheduling model discusses the structure and ideas behind the foam scheduling project. It describes in great detail the objectives and constraints of the mixed integer program, relating the plant's problems to the mathematical model. It begins by discussing linear and mixed integer programming, and gives some of the reasons why an optimization model can be used to solve a scheduling problem. This chapter also covers the mathematical model itself, and explains briefly how the objective function and constraints were translated from the plant's description into the model.

The final section of the chapter deals with the scheduling and sequencing algorithm. It briefly describes where to place the molds on the assembly line, once the model knows how many molds are necessary to satisfy production shipping requirements. The section then discusses the heuristic algorithm that was implemented in the mathematical model. Finally, the sequencing and scheduling method will be used to explain how the objectives, minimizing set-up costs and inventory storage charges, were obtained.

<u>3.1 MIXED INTEGER PROGRAMMING MODEL</u>

Linear programming is a mathematical technique for determining the solution to a system of linear constraints that maximizes or minimizes a linear objective function. An example of a typical solution is an optimum allocation of resources to achieve a particular objective when there are alternative uses for the resources (1,2).

Mixed integer programming is a mathematical technique that permits one to solve linear programming problems in which certain variables must take integer values. This possibility allows the study of a large class of important applications that cannot be handled by classical linear programming techniques (7).

1. Continuous variables, which can have any value (classical programming problems have only continuous variables).

2. Integer variables, which are limited to integer values (...,-2,-1,0,1,2,...).

Both types must, of course, satisfy the constraints of the problem.

The ability to introduce integer variables into the linear programming model provides a means for efficiently handling certain problems that otherwise could not be studied, could be studied only approximately, or could be studied only through a long sequence of linear programming runs for which a great deal of preparation is demanded.

The Assembly Line Mold Scheduling Mixed Integer Programming Model utilizes the MPSX Extended Control Language (ECL) written in PL/1 (3). The model contains 132 linear programming rows and 125 integer variables. The mixed integer programming (MIP) objective function and constraints will be covered in greater detail in the following sections.

3.1.1 Objective Function

The aim of any linear or mixed integer programming model is to maximize or minimize some objective function. Figure 2, the Assembly Line Mold Constraint Definitions, and Figure 3, the Mixed Integer Programming Model Objective Function, will describe the model's objective function. Let us consider the four types of costs which the assembly line scheduling model is seeking to minimize. The costs of producing foam parts are as follows:

1. inventory storage costs
2. mold set-up costs
3. mold framing costs
4. upender availability costs

The inventory holding costs represent the average inventory storage costs per 2 week period. It is calculated by taking the production rate for two weeks, multiplied by the number of molds currently on the line and 10% of the value of the part, then is divided by 52 weeks per year. It is not a true indication, however, of the actual inventory storage costs. Since the inventory levels are low, typically less than a two day bank, the parts which are produced early in the week are shipped that same week. This means that all the foam parts which are produced are not automatically stored in inventory. In fact, many of the parts produced are loaded into boxcars for immediate rail shipment.

The mold set-up costs are really the key to the success of the model. The foam model seeks first to satisfy production demand, then to minimize set-up costs. The two ideas are very closely linked together, for set-up costs would be non-existent if demand for a particular foam part remained constant. These costs are calculated by multiplying the number of molds added and removed from the assembly by the constant (say $7.50/mold set-up). For each mold added, there must be one mold removed, for a total set-up replacement cost of $15.00. This cost, like all others, must be kept in balance, for it takes much longer to frame a mold than to make a mold replacement on the assembly line. The framing costs are very similar to the mold set-up costs. The mold framing costs represent the time and labor involved in first unbolting a previously framed mold package, and then constructing a new mold-frame assembly. Since a mold-frame assembly is heavy, there is a need for a forklift truck and driver and two workers to accomplish the task. The higher cost is reflected in the amount the plant has assigned (see Figure 1. Assembly Line Model Costs). The framing cost is calculated by multiplying the number of molds to be framed by a constant (say $75.00). The first three conditions, the inventory, set-up and framing costs, represent the main objective function costs of the foam model.

The final objective function cost is the upender availability costs. A cost of $1.00 is given to each upender that is assigned to a particular mold. This objective function cost tends to reduce to a minimum, the number of upenders required to handle and pack the foam parts for each type. This upender constraint will be examined in more detail in the next section.

<u>Figure 2</u>. <u>ASSEMBLY LINE MOLD CONSTRAINT DEFINITIONS</u>

N = total number of different mold types

WARE = total number of storage positions available in warehouse

M(i) = total number of molds REQUIRED for the ith part

A(i) = number of molds for the ith part to be ADDED to the assembly line

R(i) = number of molds for the ith part to be REMOVED from the assembly line

F(i) = number of molds for the ith part to be MOUNTED on frames

D(i) = wire load difficulty associated with the ith part

P(i) = production rate per week associated with the ith part (currently based on 3 shifts/day - 6 days/week)

PK(i) = standard pack for the ith part (number of pieces which fit into one basket)

SH1(i) = number of pieces shipped the 1st week for the ith part

SH2(i) = number of pieces shipped the 2nd week for the ith part

INV(i) = initial inventory for the ith part

SAFETY(i) = safety stock required for the ith part

SC(i) = set-up costs associated with the ith part

IC(i) = inventory storage costs associated with the ith part (value of the ith part/260.)

FC(i) = cost to mount a mold on a frame associated with the ith part

UP(i) = cost associated with having an upender (basket) assigned to the ith part

M-tape = molds that require tape after post cure

M-large = large molds

Mp = mold position

Figure 3. MIXED INTEGER PROGRAMMING MODEL OBJECTIVE FUNCTION

$$\text{MINIMIZE COST} = \sum_{i=1}^{N} (M(i)*P(i))*IC(i) +$$

inventory holding costs

$$\sum_{i=1}^{N} (SC(i)*A(i)+SC(i)*R(i)) +$$

mold set-up costs

$$\sum_{i=1}^{N} (FC(i)*F(i) +$$

framing costs

$$\sum_{i=1}^{N} UP(i)$$

upender availability costs

3.1.2 Constraints

This section deals with the mixed integer programming constraints of the assembly line mold sequencing model. Each of the various constraints will be discussed, and the questions of how the constraints were implemented should be answered. One must remember, in a mixed integer program, that if the constraints are not general enough or do not have enough flexibility, the problem will become infeasible. If the problem is too general, however, valuable time will be spent in calculating answers. Therefore, careful attention was given to allow the problem a greater degree of freedom while minimizing the objective function. The Assembly Line Mold Constraints are available in Figure 4, and can quickly aid in understanding the mixed integer programming model.

The mixed integer programming model currently contains 132 constraints and 125 integer variables. The integer variables can be broken down into the following five categories for each mold type:

1. total number of molds required
2. total number of upenders required
3. number of molds to be ADDED
4. number of molds to be REMOVED
5. number of molds to be FRAMED

The first constraint considered here is that of LINE CAPACITY. As previously stated, it is unacceptable to leave a vacant position on the assembly line. Therefore, any feasible MIP solution must fill the 181 mold positions with an available mold. This is done even at the expense of producing foam parts which are currently not necessary for the production shipping requirements.

ADDING and REMOVING MOLDS from the assembly line are the next constraints to be considered. Since the model is trying to determine the total number of molds required for the ith part, M(i), and at the start of the model we know how many molds are currently on the line, M(i) current, the number of molds to add for the ith part, A(i), represents the difference between the required number of molds and the current number on the line. In the line, R(i), represents the difference between the current number of molds and the required number on the line. For every mold that is added on the assembly line, there is another mold which is removed. An empty position on the line is never created.

The FRAMING MOLDS constraints is handled slightly differently than the two previous constraints. The model, at some starting point, knows how many molds are currently framed. The number of molds to be framed, F(i), represents the difference between the number of currently framed molds and the number of molds required. Typically, however, there usually seems to be enough extra molds framed to meet the future production requirements. That is why there is seldom any need to frame more molds. To speed up the MIP subroutine, a limited number of molds can be framed, for any particular part. Two molds of each type can be framed, except for mold codes: "BN" or "BR", in which case the model allows four framing changes. This is due to the fact that the mold codes "BN" and "BR" are high production volume parts which account for half of the total assembly line production.

The WIRE LOAD DIFFICULTY constraints were included in the model to make future mold scheduling and sequencing easier. The idea behind this constraint was that the total wire load difficulty of the assembly line should be maintained at some constant level of wire load difficulty. This implies that before a series of molds are added or removed from the assembly line, the sum of the wire load difficulties be almost equal. One must remember to maintain some flexibility in the model, in order to insure a feasible solution. If no molds are added or subtracted from the assembly line, the wire load difficulty of the line remains constant and the constraint is not binding. In an attempt to maintain this balance, the MIP model allows the user to specify a range of difficulty values (i.e. range from -5 to +5). By examining the row constraint: "WIRE", one can see how the solution affected the total wire load difficulty of the assembly line. The wire load difficulty constraint was not necessary to satisfy the plant requirements, but it greatly aided in the scheduling and sequencing algorithm.

The LARGE and SMALL MOLD constraint was also really not necessary from the plant's standpoint. Generally, these constraints helped limit the types of mold changes which can occur on the assembly line. Whenever a large or small mold is removed from the line, it should be replaced by a mold of the same size. Again, some flexibility in the model must be maintained for feasibility and, therefore, both totals should have a range of values (i.e. range from -5 to +5). By examining the row constraints: "LARGE" and "SMALL" constraints should sum to zero. This makes sense if one considers adding for an example, say a total of two extra small molds. The molds which are removed must be large molds to compensate for the small molds which were added. One must remember, however, that we are considering just the total numbers of large and small molds on the assembly line.

The MAXIMUM MOLD AVAILABILITY constraint represents the upper limit of the total number of molds required.

Clearly, the plant cannot mount more molds on the assembly line than are currently available or physically present. If M(i) equalled M(i) available for several parts, serious problems in framing costs would result. Typically, the solution to the MIP problem would be infeasible to implement.

The MINIMUM PRODUCTION constraint is the direct opposite of the MAXIMUM MOLD AVAILABILITY constraint. It represents the lower limit of the total number of molds required. In other words, this constraint determines the minimum number of molds to meet the current production shipping requirements. The MINIMUM PRODUCTION constraint uses a weighted average of shipping requirements for a 4 week period, minus the initial inventory plus the guaranteed safety banks. The weighted average is composed of 70% first week shipped, 20% second week shipped, and 5% the third and fourth week shipped. These percentages seem to reflect the true shipping forecasts and with the safety banks, guarantees that the first week shipping requirements are met. This quantity is then divided by the production rate per week to give the required minimum number of molds. Since this quantity generates a real number with a fractional component, the MIP model truncates the answer to an integer value. In examining the solution to the MIP problem, the plant can look to see if any of the required molds for any part are at their lower limit. Typically, this condition rarely exists in the program unless the upper and lower limit values are fixed.

The INVENTORY CONSIDERATIONS constraint is perhaps the most important of the plant objectives. The space requirements in the warehouse are a constant headache to the plant management. The imbalance between production and inventory control often forces the production of too many foam parts. The warehouse becomes full of foam parts and with no co-ordination between the different sections of the plant, shipping requirements are not met. Therefore, there must be some inventory overflow constraint which takes into account the limited inventory storage space available in the plant. The INVENTORY CONSIDERATIONS constraint is calculated for each individual foam part. Two weeks production in terms of the number of baskets, must be less than or equal to the available positions in the warehouse plus the number of baskets shipped and available to store in the warehouse. The total inventory and number of parts shipped in one week is calculated and given in the row constraint: "INVEN". If inventory becomes a problem in the solution of the MIP problem, and if surplus production is available, the model will tend to select smaller foam parts. The idea here is that more small foam parts can be packed into a basket, and less storage space required for equivalent production. Therefore, this constraint tends to keep the warehouse from becoming filled to capacity, and a stable production and inventory policy will result.

The UPENDER AVAILABILITY constraint represents the total number of upenders or baskets available in the loading area for the packing of foam parts. The plant would like to see the total number of upenders kept in the plant to be less than or equal to 30. In the plant environment, however, there is actually room for as many as 32 positions if the scrap area size is reduced. Again, this constraint is necessary because of the limited floor space which exists in the plant. The optimal solution to any MIP problem must try to keep the total number of upenders to a minimum. This is one of the reasons why a $1.00 per upender cost was assigned to each upender required for production packing.

The UPENDER ASSIGNMENTS represent the last set of constraints which the mixed integer program considers. These two constraints indicate the upper and lower limits of the number of upenders associated with a particular part. The upper limit was set at 6 upenders per mold code since in the plant environment, there are rarely more than 5 upenders actually assigned. The lower limit, however, is the real key to the assignment of upenders by the MIP model. We knew that the production cycle to produce one piece takes about 15 minutes or roughly 4 pieces per hour. The UPENDER ASSIGNMENTS are then calculated by taking 4 parts per hour and multiplying it by the number of molds required. The result is divided by the standard pack which is the number of foam parts per basket. This lower limit assumes that the baskets can be moved into the inventory/ shipping area at least once per hour. On high volume parts, however, such as mold codes "BN" or "BR", the assumption is made that these baskets can be replaced every half hour. High volume parts always require dedicated truck drivers because of the high production turnover rate. Another interesting point is that if one mold is assigned to the assembly line, there must be one basket allocated for that foam part. If there is a questionable mold which is a low volume part, the MIP model will try to eliminate this mold from the new mold line-up. Since the mold is eliminated, there is no longer a need to allocate an upender for that particular part. Upenders are in such short supply that better use of these limited resources can be determined by the model.

Figure 4. ASSEMBLY LINE MOLD CONSTRAINTS

Description	Constraint
LINE CAPACITY	$\sum_{i=1}^{N} M(i) = 181$
ADDING MOLDS for i=1..N	$M(i)-A(i) <= M(i)current$
REMOVING MOLDS for i=1..N	$M(i)+R(i) => M(i)current$
FRAMING MOLDS for i=1..N	$M(i)-F(i) = M(i)framed$
	$F(i) <= 2$
if mold = 'BN' or 'BR'	$F(i) <= 4$
WIRE LOAD DIFFICULTY	$\sum_{i=1}^{N} (D(i)A(i)-D(i)R(i)) <= 5$
	$\sum_{i=1}^{N} (D(i)A(i)-D(i)R(i)) => -5$
LARGE MOLD	$\sum_{i=1}^{N} (A(i)large-R(i)large) <= 5$
	$\sum_{i=1}^{N} (A(i)large-R(i)large) => -5$
SMALL MOLD	$\sum_{i=1}^{N} (A(i)small-R(i)small) <= 5$
	$\sum_{i=1}^{N} (A(i)small-R(i)small) => -5$
MAXIMUM MOLD AVAILABILITY for i=1..N	$M(i) <= M(i)available$

MINIMUM for i=1..N
PRODUCTION

$$M(i) \Rightarrow \frac{.70*SH1(i)+.20*SH2(i)+.05*SH3(i)+.05*SH4(i)-INV(i)+SAFETY(i)}{P(i)}$$

INVENTORY CONSIDERATIONS

$$\frac{2.0*P(i)*M(i)}{PK(i)} <= WARE + \frac{SH2(i)+SH1(i)-INV(i)+SAFETY(i)}{PK(i)}$$

UPENDER AVAILABILITY $\sum_{i=1}^{N} UP(i) <= 33$

UPENDER for i=1..N
ASSIGNMENTS

$$UP(i) \Rightarrow \frac{4.0*M(i)}{PK(i)}$$

if mold = 'BN' or 'BR'

$$UP(i) \Rightarrow \frac{2.0*M(i)}{PK(i)}$$

for i=1..N UP(i) <= 6

PLATFORM LOADING for i=2..(N-1)
RESTRICTIONS
Mp(i-1)='AN'ǂMp(i)='MM'ǂMp(i+1)='AN'
Mp(i-1)='MM'ǂMp(i)='AN'ǂMp(i+1)='MM'

"GUN" RESTRICTION
Mp(1)=Mp(180)
Mp(2)=Mp(181)

TAPE AFTER POST for i=2..(N-1)
CURE RESTRICTION Mp(i-1)=M-tapeǂMp(i)=M-tapeǂMp(i+1)=M-tape

MOLD SIZE for i=2..(N-1)
RESTRICTION Mp(i-1)=M-largeǂMp(i)=M-largeǂMp(i+1)=M-large

Figure 4. (continued)

3.2 SCHEDULING AND SEQUENCING ALGORITHM

Once the model knows what molds to add, remove, or frame, it is necessary to determine where on the assembly these molds belong. Briefly, this is the purpose of the scheduling and sequencing subroutine. At this point in the program, the model is attempting to eliminate past problem areas, satisfy the sequencing rules, and minimize the number of mold changes. The problem now exists in trying to determine the minimum set of possible mold substitution points which satisfy the MIP solution and the sequencing constraints. First, however, we should discuss the plant constraints that could not be satisfied in the MIP problem. They deal exclusively with the positioning relationships of the molds on the assembly line.

The PLATFORM LOADING RESTRICTIONS is a constraint which attempts to equalize the workload on the operator who must load wires into the upper sections of particular molds. A special 40 foot platform has been designed to handle these complicated foam parts. It is critical that only one man handle the task, and the work be spread out. This is required, simply so as not to overload or overwork one operator. The constraint states that when molds are added to the line, careful consideration should be given to not to place "AN" or "MM" molds next to each other.

Both molds together are difficult as far as wire load difficulty is concerned, and to have one man perform the wire loading carries him off the platform. All one could do would be to let one mold go by empty, thus producing a bad part. This is a totally unacceptable solution to the problem, so the scheduling and sequencing algorithm should never let this condition happen.

The "GUN" RESTRICTION constraint is perhaps one of the most interesting constraints to see on the assembly line. It states that mold positions #1 and #180, and #2 and #181 be the same mold type. The reason for this constraint is that at the end of the assembly line cycle, the "gun" must be cleaned. Cleaning is accomplished by sending a burst of high pressure air through the line, to remove any excess foam plastic that may have collected in the "gun". There are physically more than 181 mold positions on the line, but only 181 positions are capable of accepting molds. Timing is a very important consideration, for the assembly line must be maintained at a constant speed even though cleaning is required. Four positions are necessary, and the mold spacing between positions #180-2 are slightly different than the remainder of the line.

The TAPE AFTER POST CURE RESTRICTION is another constraint which attempts to solve a sequencing problem on the assembly line. There are several parts which require stapling a cloth piece to the foam part after it has gone through the curing oven. Mold codes: "MM", "AJ", "AH" and "KL" must be spread out so that one operator can perform the work. Recall, however, that mold code "MM" was one of the molds which required platform wire loading. The PLATFORM LOADING RESTRICTION and the TAPE AFTER POST CURE RESTRICTION then are in direct conflict with each other on this particular mold. One constraint forces the molds together while the other requires that they be separated. In any case, some balance and compromise between constraints is often the solution.

The MOLD SIZE RESTRICTION is the last scheduling and sequencing constraint to be considered. It simply states that when scheduling molds on the assembly line, two large molds cannot be placed next to each other. Besides being physically impossible to install, the large molds typically have a higher wire load difficulty which makes them tougher to sequence. This is one of the few constraints which MUST be obeyed at all times for a feasible scheduling and sequencing solution.

3.3 SUMMARY

The past two sections have described the assembly line scheduling model constraints which were implemented in the mixed integer programming subroutine and the scheduling and sequencing subroutine. The mixed integer programming objective function and optimization constraints were explained in respect to the plant's production plans and problems. The following chapters deal with how these constraints were actually implemented, and describe the output which resulted from each of the subroutines.

IV. SCHEDULING SYSTEM OVERVIEW

This chapter describes the scheduling system overview which represents a detailed analysis of the foam scheduling model. The main program and the five assorted subroutines will be briefly described in this section to acquaint the reader with the flow of the program model. The objective of this chapter is to explain the computer programs from the input through to the final scheduling and sequencing section. A structured programming, modular design approach was used in the model to help reduce the possibility of errors, to make the code understandable to others, and to break up the various program functions (11).

The assembly line mold scheduling model is composed of a main program and five subroutines as follows:

1. main program - MAIN
2. input processor - INPUT
3. assembly line pictorial subroutine - LINEUP
4. generate constraints subroutine - GENER
5. mixed integer programming subroutine - MIP
6. scheduling and sequencing subroutine - SCHED

The computer programs written for Assembly Line Mold Scheduling are presently operating on an IBM 370/145 VM/VS1 (virtual storage) computer at General Motors Manufacturing Development. It could, however, run on any computer system which makes use of the Programming Language One (PL/1) Optimizing Compiler (8,9,10), and IBM's MPSX/MIP Mixed Integer Programming package (2,3,4,5,6,7).

The mold scheduling subroutines were specifically designed to run in less than 768K (due to MPSX/MIP) of virtual core memory. The five mold scheduling subroutines and main program were compiled and stored in an object module library for increased speed and easy access. Typical execution times averaged about one minute (CPU) with 40-50 seconds of execution time spent in the mixed integer programming subroutine.

SYSTEM SUBROUTINE OVERVIEW

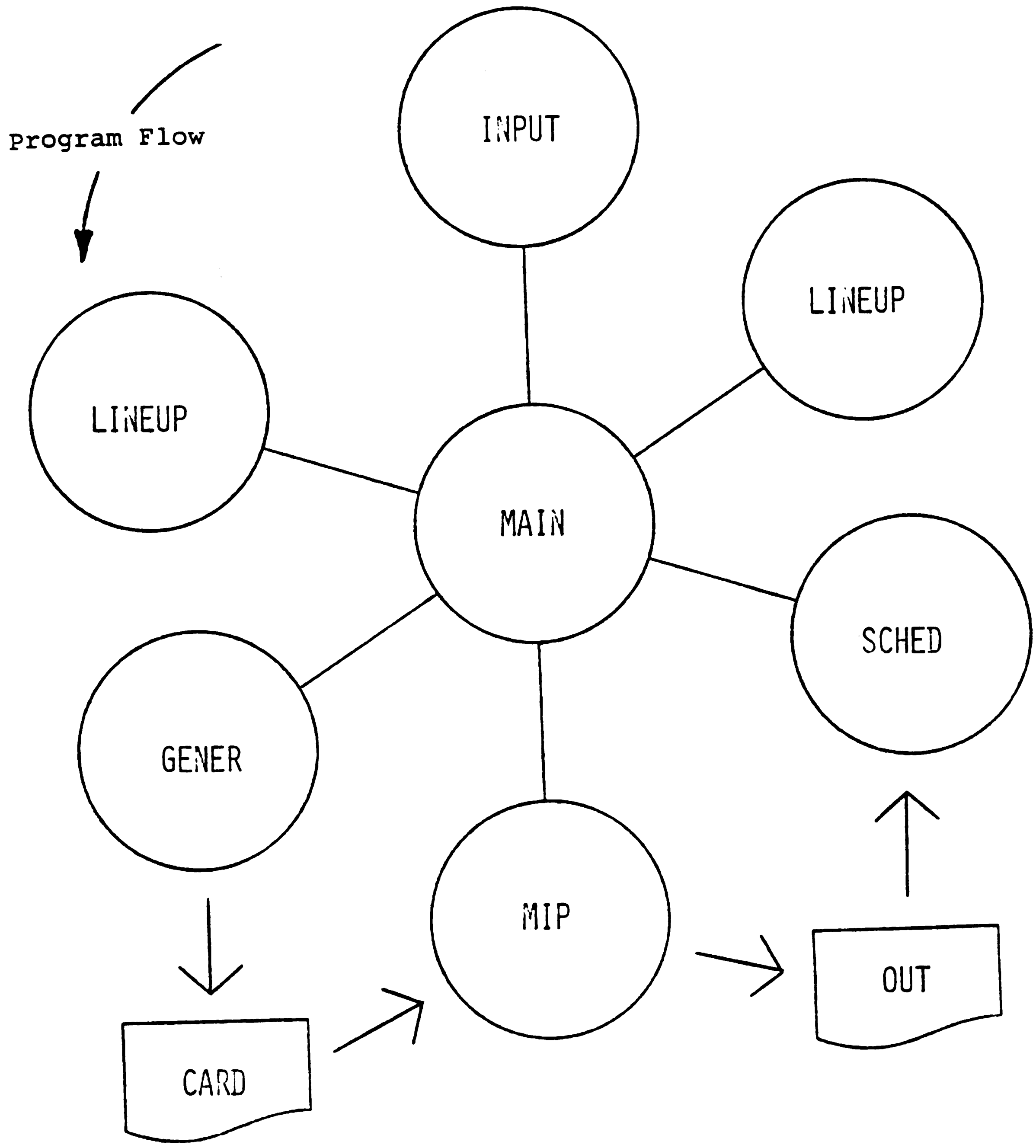


Figure 5. System Subroutine Overview

4.1 MAIN PROGRAM - MAIN

The MAIN program of the foam scheduling model follows many of the structured programming conventions. It merely represents a calling program to tie together the various subroutines. MAIN gives a thorough listing of the static external (i.e. similar to common in FORTRAN) references and explains the meaning of each of the various arrays and variables. Before the program calls a subroutine, however, a message is displayed at the operator's console, so that the user can determine what subroutine is presently being executed. This helps in debugging the model, especially when the mixed integer programming subroutine is found to be infeasible and terminates abnormally. In any case, it represents an easy method of determining the time and position of execution during the flow of the program. MAIN rarely has to be changed since the control of the program remains fairly constant.

4.2 INPUT PROCESSOR - INPUT

The input processor subroutine, INPUT, represents the first working program of the foam scheduling model. Its purpose is to initialize the static external variables and arrays, and to read the various input parameters and current mold information. The program begins by reading the mold line-up title card, which is an 80 character description of the problem to be solved.

The parameter input card follows the mold title card. It has variables necessary for the problem, such as the total number of mold types (NUM), the number of shifts (SHIFT), the total number of molds on the assembly line (IPOS), the number of currently available positions in the warehouse (WARE), and the average wire load difficulty for three consecutive positions on the line (LWD).

The following NUM (typically 25) cards are now read into their respective arrays. This group of cards is often referred to as the "static" section, because the values of the parameters rarely change. Variables such as mold codes (CODE), part numbers (PART), mold size (SIZE), wire load difficulty (LOAD), number of part per cavity (NUMPRT), platform loading (PLAT), total molds available (MOLDS), total molds framed (FRAME), tape after post cure (TAPE), standard pack per basket (PACK), production rates per week (RATE), value of the parts (VALUE), and safety stock (SAFETY) are read into the program. These variables are required in creating the mixed integer programming constraints, and for future scheduling considerations.

The final section of the input processor deals with the mold line-up information. This group is known as the "mold line-up" section because the current assembly line configuration must be known to the foam scheduling model. It consists of reading the 181 assembly line mold codes as they appear on the line in their current sequence and position. The subroutine then returns to the MAIN program before being transferred to the LINEUP subroutine.

```
                    ***** BENCH LINE #2 - MOLD LINE SET-UP INFORMATION *****

25   3     181     1500     28
AC   1657569 S  0 1 N 41 27 N   55 415   1.85   2145
YR   1660106 S  9 2 N 19 14 N   78 820   1.89   3042
YL   1660107 S  9 2 N 19 13 N   78 820   1.89   3042
AH   1677286 L 12 1 N  2  2 Y   20 385   5.02    720
AJ   1677290 L 12 1 N  4  2 Y   20 340   4.99    720
AL   1681929 S  7 1 N  2  1 Y   28 325   4.33    336
AN   1690551 L 11 1 Y  9  4 N   22 375   3.94    858
AO   1690553 S  6 1 N  9  5 N   22 405   4.15    858
AY   1690562 S 11 2 N  5  5 N   64 790   1.80   1280
AZ   1690563 S 11 2 N  4  4 N   64 790   1.80   1280
AW   1694482 L  6 2 N 21 18 N   72 830   2.17   2280
AX   1694483 S  0 2 N 21 19 N   66 830   1.51   2706
AV   1694709 S  0 1 N 21 20 N   28 415   3.38   1176
BG   1733947 S 11 2 N  1  1 N   64 650   1.80   2304
BH   1733948 S 11 2 N  2  0 N   64 710   1.80   1152
BJ   1737211 L 22 2 N  6  7 N   48 820   2.52   1920
DK   1737223 S 16 2 N  6  6 N   60 810   1.89   2340
KR   3062748 S  9 2 N  6  2 Y   56 780   1.93   2128
KL   3062749 S  9 2 N  6  3 Y   56 790   1.93   2128
GI   3063026 L  7 2 N  4  4 N   62 800   1.74   2356
MM   9679612 S  5 1 Y 23 15 Y   28 415   3.55   1288
VV   9679613 S  3 2 N 22 18 N   84 830   1.28   4158
JI   9736082 S 14 1 N  5  2 N   24 380   4.24    864
BR  20001557 L  0 1 N 50 49 N   22 410   3.96   3144
BN  20001606 S 11 1 N 63 47 N   24 410   4.09   3456


YR YR YR YL YL AC BJ AC AC YL YL AC BN AC AJ AC BN BR BN BR
BN BR BN BR BN BR BN BR BN BR BN BR BN BR BN BR BN BR BN BR
BN BR MM AN MM VV MM AN VV MM VV MM AX BJ MM VV MM VV MM BJ
MM VV MM VV MM AH VV MM VV MM BK MM YL MM BK AC AC BR BK BR
AX BN BR BN AX BN AX BN AX BN AX BN BR KL AX AW AX AY AX AW
AC KR AC AW VV AZ VV AW VV AW AV BJ AV AX AV BJ AV AV YL AW
BN AX BR BN BR AC BN BR BN BR BN BR BN BR BN BR BN BR BN BR
BN BR BN BR BN BR BN BR BN BR BN BR BN BR BN BR BN BR BN BR
BN BR BN BR BN AC BN AC AX AX VV BG BR AO BR DK BR AC BK YR
YR
```

Figure 6. Input Processor Summary

4.3 ASSEMBLY LINE PICTURE SUBROUTINE - LINEUP

The assembly line picture subroutine, LINEUP, is perhaps the most important subroutine in the foam scheduling model. It conveys a picture of the current assembly line mold configuration and does a great amount of error checking. A mold group is composed of a mold position (1-181), a wire load difficulty for that particular mold (0-22), the mold code (i.e. AC,BN,BR), and a space for the problem area or special assembly line functions.

The lower left side of Figure 7 summarizes, by mold type, the number of various molds on the line. The 181 mold positions, at a glance can then be examined to determine where the problem areas exist. These are denoted by a character string of '*****' under the molds in question. A problem area results when the sum of wire load difficulties for any three consecutive molds exceeds some wire load difficulty average (say 27).

The symbolic characters "@@" and "$$" denote tape after post cure and platform loading work, respectively. This can give the user valuable information about the sequencing rules to determine whether certain mold types are bunched too closely together or spread out too far. These symbols also appear along with the problem area field, under the mold codes. In several instances, one mold type (i.e. MM) may have both functions, and appears on the computer printout as an over printed character string.

In the lower right hand corner of Figure 7, a summary of the problem areas is given. It states the mold positions which exceed the wire load difficulty average and also gives its total. If no problem areas exist, a comment about platform loading and tape after post cure will appear. This implies to the user that the run was successful, and that no major problem areas exist on the proposed future line-up.

Finally, after the mixed integer program and the scheduling and sequencing subroutine has been completed, the LINEUP subroutine once again is printed. This time, however, there are several distinct changes. In the lower center of the page, there are three columns (i.e. Mold Position, Add Mold, Remove Mold) which summarize the line position and mold changes that have to be made. This in effect, is the solution of the foam scheduling problem, and quickly aids the user in making his weekly mold decisions and changes. If no modifications have been made to the assembly line, this section will then be empty. The final scheduling sheet then, refers to the assembly line as it would be if all the changes had already been made. It is the only output which is necessary for Production Control personnel to use in making their weekly mold decisions on the assembly line.

```
                              ***** BENCH LINE #2 - MOLD LINE SET-UP INFORMATION *****         04/04/77

    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
    9    9    9    9    9    0   22    0    0    9    9    0   11    0   12    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0
   YR   YR   YR   YL   YL   AC   BJ   AC   AC   YL   YL   AC   BN   AC   AJ   AC   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR
                        ******                                      @@

   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
   11    0   11    0   11    0   11    0   11    0    5   11    5    3    5   11    3    5    3    5    0   22    5    3    5    3    5   22    5    3    5    3
   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   MM   AN   MM   VV   MM   AN   VV   MM   VV   MM   AX   BJ   MM   VV   MM   VV   MM   BJ   MM   VV   MM   VV
                                                     @@   $$   @@        @@   $$        @@        @@           **@@**        @@     **@@*******@@**     @@

   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96
    5   12    3    5    3    5   16    5    9    5   16    0    0    0   16    0    0   11    0   11    0   11    0   11    0   11    0   11    0    9    0    6
   MM   AW   VV   MM   VV   MM   BK   MM   YL   MM   BK   AC   AC   BR   BK   BR   AX   BN   BR   BN   AX   BN   AX   BN   AX   BN   AX   BN   BR   KL   AX   AH
   @@   @@        @@        @@      **@@**  **@@**                                                                                               @@

   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128
    0   11    0    6    0    9    0    6    3   11    3    6    3    6    0   22    0    0    0   22    0    0    9    6   11    0    0   11    0    0   11    0
   AX   AY   AX   AW   AC   KR   AC   AW   VV   AZ   VV   AW   VV   AW   AV   BJ   AV   AX   AV   BJ   AV   AV   YL   AW   BN   AX   BR   BN   BR   AC   BN   BR
                            @@

  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160
   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0
   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR

  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181
   11    0   11    0   11    0   11    0    0    0    3   11    0    6    0   16    0    0   16    9    9
   BN   BR   BN   BR   BN   AC   BN   AC   AX   AX   VV   BG   BR   AO   BR   BK   BR   AC   BK   YR   YR
                                                                                             ******

      TOTAL NUMBER OF MOLDS                                                        PROBLEM AREAS
             ON LINE                                                        POSITION      TOTAL WIRE DIFFICULTY
             AC=  14                                                           180                34
             YR=   5                                                            60                32
             YL=   6                                                             6                31
             AH=   1                                                            55                30
             AJ=   1                                                            61                30
             AL=   0                                                            72                30
             AN=   2                                                            74                30
             AO=   1                                                            59                30
             AY=   1
             AZ=   1
             AW=   6
             AX=  13
             AV=   5
             BG=   1
             BH=   0
             BJ=   5
             BK=   5
             KR=   1
             KL=   1
             GI=   0
             MM=  15
             VV=  11
             II=   0
             BR=  41
             BN=  41
```

Figure 7. Mold Line-up Summary

4.4 CONSTRAINT GENERATION SUBROUTINE - GENER

The purpose of the GENER subroutine is to construct a set of card image constraints to be used as input for the mixed integer programming subroutine. GENER begins by reading the "dynamic" forecasting information; the name referring to variables which change value after every program run. This section in fact, summarizes the current inventory levels and the future plant shipping requirements. Variables such as initial inventory balance (INVBAL), production required for 1st week shipped (SHIP1), 2nd week shipped (SHIP2), 3rd week shipped (SHIP3), and 4th week shipped (SHIP4) are included for each mold type. The model checks to see whether any information is missing or mispelled, and totals the initial inventory and parts to be shipped in the first two weeks. A summary of the forecasted production information follows the initial line-up computer printout, and aids the user in finding errors in the input data.

At this point, the model has enough information to start generating the input constraints. The card images must appear in certain columns and must be sorted in the proper order. The input consists of five major sections:

1. row cards
2. column cards
3. right hand sides
4. mold range cards
5. bounds section

It is important to note that a file, called "CARD" is created within the subroutine, and that all the card images are stored there for future use by the mixed integer subroutine.

Figure 8 on the following page, represents a summary of the inventory and shipping requirements at the start of the model. These values are necessary to determine the future weeks production demands and mixed integer programming constraints.

FORECASTING - PRODUCTION INFORMATION

	FCD BAL	INITIAL INVENTORY	1ST WEEK SHIPPED	2ND WEEK SHIPPED	3RD WEEK SHIPPED	4TH WEEK SHIPPED	SAFETY STOCK
AC	192	2778	2970	2750	2750	4290	2145
YR	2226	1518	3744	3744	3432	5460	3042
YL	1492	1784	3276	3588	3900	5616	3042
AH	182	138	320	360	400	680	720
AJ	-57	337	280	240	280	480	720
AL	-110	110	0	112	56	0	336
AN	499	689	1188	1188	968	1936	858
AO	-451	1419	968	1364	880	2156	858
AY	-1244	1884	640	1024	512	1536	1280
AZ	200	696	896	1280	640	2176	1280
AW	5015	745	5760	5184	5184	8496	2280
AX	-2859	8139	5280	5940	5412	8580	2706
AV	3341	-765	2576	2520	2744	4816	1176
BG	1	255	256	384	256	768	2304
BH	-189	317	128	128	0	128	1152
BJ	1829	667	2496	3072	3168	4992	1920
BK	-334	2974	2640	2880	3120	5040	2340
KR	407	713	1120	1344	784	2240	2128
KL	988	244	1232	1232	896	2352	2128
GI	-2777	2777	0	0	0	0	2356
MM	2480	2056	4536	4480	4536	8344	1288
VV	8764	146	8910	9108	8910	17820	4158
II	-150	390	240	240	192	480	864
BR	-2312	15688	13376	15224	15886	27224	3144
BN	-2000	17456	15456	17568	18432	32621	3456

TOTAL PARTS SHIPPED IN TWO WEEKS = 163242
TOTAL INITIAL INVENTORY = 63155

. Figure 8. Forecasting - Production Information

4.5 MIXED INTEGER PROGRAMMING SUBROUTINE - MIP

The Mathematical Programming System Extended (MPSX/MIP) package is composed of a set of procedures, a subset of which deals only with linear programming. The strategy for solving an linear programming problem is the ordered execution of a series of these procedures. The user conveys the proposed strategy to MPSX via the MPSX extended control language (ECL) written in PL/1. The procedure call statement of the control language calls the linear programming procedures and transfers arguments to them.

The linear programming procedures of MPSX use the bounded variable/product form of the inverse/revised simplex method. The simplex method is based upon the fact that if there are m constraints (or rows) in the constraint matrix and these are linearly independent, then there is a set of m columns (variables or vectors) which are also linearly independent. Hence, any right-hand side (RHS) can be expressed in terms of these m columns (called a basis). The simplex method uses these basic solutions, stepping from one to another (by exchanging one column with one column not in the basis on each step or iteration), until a solution (called a basic feasible solution) is obtained that meets all the criteria, including the requirement that all the column values be non-negative.

Problems for which this last condition does not hold are automatically subjected by MPSX to an internal linear transformation to bring them to this form. The bounded variable feature allows the user to specify limits on the activities levels for any or all of the variables. Either upper or lower bounds, or both, may be specified. Since the bounds would otherwise have to be represented by explicit constraints, use of this feature leads to economies in the number of constraints and in computing time.

After a basic feasible solution is found, the simplex method steps along, examining a series of basic feasible solutions, to find one that satisfies the requirement that the value of the functional (or objective) row be a maximum or minimum; this is called the optimal solution. Not all linear programming problems have an optimal solution. If there is no solution at all in non-negative variables, or none that keeps the variables within their specified bounds, the linear programming problem is said to be infeasible. If a feasible solution is found, but the constraint rows do not confine the value of the functional row to finite values, the linear programming problem is said to be unbounded.

If it is assumed that the nonbasis variables all have zero value, then there are m basis variables left whose values have to be chosen to satisfy m constraints. The solution of these constraints for the values of the m basis variables requires the inverse of the m*m matrix of the coefficients of the basis variables in the constraints. The recognition of the role of this inverse leads to the revised (as opposed to the original) simplex method. The product form of the inverse is a representation that leads to economies in computing time and storage requirements and to increased numerical accuracy.

In the product form, the inverse is represented by the product of a sequence of m*m matrices, only one column of each matrix differing from a column of the unit matrix. It is necessary only to record which column, and the nonzero elements in that column, to have a full description of one matrix in the sequence. (This column is termed the "eta" vector.)

There is one matrix in the sequence (and, therefore, one eta vector) for each iteration that has been carried out. Clearly, as the sequence lengthens, the computational advantages decrease. However, the product form can be consolidated by "reinversion", which, in effect, replaces the existing product form of the current inverse by a minimal (in regard to the number of eta vectors and number of nonzero coefficients) product form.

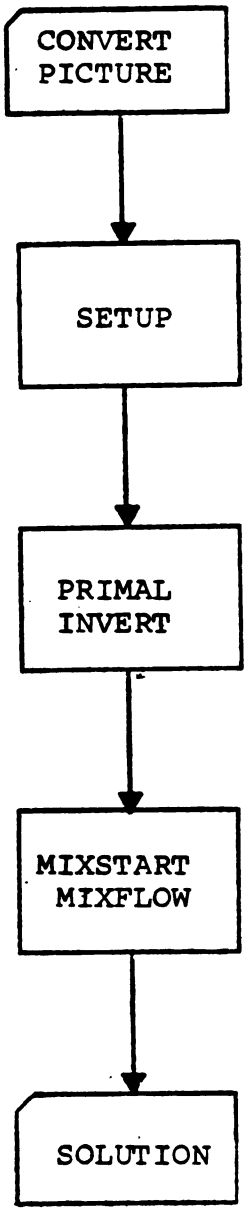
The purpose of Mixed Integer Programming (MIP) is to meaningfully increase the scope of MPSX by providing the capability for studying mixed integer linear programming problems. A mixed integer linear programming problem is a linear programming problem with two kinds of variables: integer variables and continuous variables. Integer variables can take only integer values, that is, ...,-2,-1,0,1,2, etc. Continuous variables can take any real number as a value (classical linear programming problems have continuous variables exclusively).

The study of a mixed integer linear programming problem is performed in two distinct stages. First, the problem is optimized by considering all integer variables as being continuous. It is, therefore, an ordinary linear programming problem whose optimization is performed by the linear programming module of MPSX. The optimal solution obtained is called an optimal continuous solution. Then the problem is searched for integer solutions, that is, feasible solutions satisfying the constraints and giving integral values to integer variables.

The search for integer solutions is aimed at finding an optimal integer solution. A straightforward strategy leads to a series of integer solutions tending towards the optimal integer solution (in other words, the values that these integer solutions give to the objective function become better and better). When an integer solution is found, it is not immediately known whether it is optimal. The search must, therefore, continue until either a better solution is found or it is proven that no better solution exists. Occasionally, particularly for problems with many integer variables and relatively loose constraints, good solutions are quickly found, but a long computation is necessary either to improve them slightly or to prove their optimality.

The MPSX/MIP control program is composed of a set of procedures which perform various linear programming functions. Figure 9, the Mixed Integer Programming Flowchart, summarizes the basic building blocks of this linear programming package. These procedures start with the LP constraints, obtain a continuous solution, and finally determine an optimal integer solution.

Figure 9. Mixed Integer Programming Flowchart



Problem Input - CONVERT, PICTURE

Initiate a Problem Solution - SETUP

Obtain a Continuous Solution - PRIMAL
INVERT

Search For Integer Solutions - MIXSTART
MIXFLOW

Print Solution - SOLUTION

CONVERT is the basic means of problem input. The procedure reads the input data, converts it into packed binary format, and writes it on the PROBFILE. Not only does the PROBFILE built by CONVERT contain all the problem data, but it may also be augmented by bases which are saved during a run.

SETUP is the basic means of initiating a solution of this problem. It has three main purposes:

1. Storage allocation and I/O initialization.
2. Creation of the work matrix.
3. Determination of an initial solution.

PRIMAL, the main optimization procedure, optimizes the problem contained at present on the work matrix. PRIMAL usually terminates in one of three states:

1. the optimal solution
2. the infeasible solution
3. the unbounded solution

This procedure initially requies a complete starting basis; this is accomplished by SETUP, which supplies an all-logical basis. The user may modify this basis or supply a complete basis on his own. PRIMAL exists with the present basis stored internally (this basis may or may not be optimal). It uses a composite algorithm and the revised simplex method. It progresses from the initial basis to the optimal basis by a series of vector interchanges; one vector is introduced into the basis and one is removed. Each of these interchanges is known as an iteration.

INVERT is the procedure that takes a current basis and produces its inverse in terms of eta vectors. At each iteration in the optimizing process, the inverse of the current basis is not computed but is represented by a set of eta vectors. Each iteration produces a new eta vector. At certain times, it is profitable to do a complete basis inversion, both for time considerations and for the removal of possible accuracy troubles.

PICTURE creates a "picture" of the current matrix in condensed format; it contains 45 rows and up to 55 columns per output page, and the pages are numbered in matrix notation for easy identification. PICTURE must be called after SETUP. The magnitude of the nonzero coefficients is indicated by an alphabetic code or an asterisk. A summary of magnitude classes, together with the meaning of the alphabetic code, is given at the end of the output. The RHS ranges, and the bounds on variables will be indicated, if they exist.

SOLUTION tabulates the current solution of the linear programming (LP) problem. Normally, this tabulation is printed (that is, written on the system device SYSPRINT), but, by using the keyword parameter FILE, the user can direct that it be filed in Communications or Standard Format on some other designated file. The status of the solution can either be:

a. FEASIBLE

b. NONOPTIMAL (feasible by implication)

c. OPTIMAL (feasible by implication)

MIXSTART is the basic means for preparing the search for integer solutions. It has three distinct uses:

1. Initializations to begin the search for integer solutions from the optimal continuous solution.

2. Restoration of a tree and the associated search status previously saved by MIXSAVE in the problem file. The MIXSTART parameter for this option is RESTORE. It is possible to forbid certain nodes which are presently waiting in the restored tree from being chosen as branching nodes during the new part of the search. Nodes and prenodes are not distinguished here because they have similar processing.

3. Continuation of the search initiated by MIXSTART in this run in certain special cases. One such case is when the search has been interrupted at an integer node and MIXFIX has been called. In this case, MIXSTART has created a mixed phase, destroyed later on by MIXFIX. This mixed phase is now to be restored in the same run.

MIXFLOW searches for integer solutions using the 'branch and bound' method. A MIP/370 tree is scanned by two main processes:

1. Node analysis: choice of a branching variable and determination of its new bounds, creation of new nodes, and choice of a branching node.

2. Branching: optimization of a subproblem.

The search for integer solutions must be initialized by MIXSTART, which initializes the beginning of a search, restores a tree previously saved by MIXSAVE, or continues a search already initiated. The primary elements of the problem, XOBJ (name of the objective function) and SRHS (name of the right-hand side), must not have changed since MIXSTART was called.

When MIXSTART is used to initiate the beginning of the search, the current solution need not be an optimal continuous one. When MIXFLOW begins the search, the current solution must be an optimal continuous one and PRIMAL should have checked its optimality-feasibility in the run.

MIXFLOW normal exit is taken when the last integer solution found is proved to be optimal. This solution is automatically restored and becomes the current solution.

4.6 SCHEDULING AND SEQUENCING SUBROUTINE - SCHED

The scheduling and sequencing subroutine brings to a conclusion the work begun by the assembly line mold scheduling model. At this point in time, several important questions have been resolved by the mixed integer programming subroutine. They involve what molds should be added, removed, or framed to meet the production shipping and inventory requirements of the plant. The problem for the scheduling and sequencing subroutine to solve is what mold positions should be chosen to:

1. minimize the mold set-up changes
2. eliminate past problem areas
3. obey the assembly line sequencing rules
4. insure that the proper number of each mold type is placed on the assembly line
5. minimize the average wire load difficulty for all positions on the assembly line

The assembly line scheduling and sequencing subroutine analyzes this problem and heuristically solves this problem. The subroutine begins by reading the optimal or best integer solution obtained from the mixed integer programming subroutine. The output, stored in a file named "OUT", is the MIP solution written in a standard format file. The solution is printed for inspection by the user and appears in

Figure 10 on the following page.

In this figure, the name and activity columns are perhaps the keys to understanding the solution to the MIP problem. They combine a two letter mold code with a three letter section identification. In the ROW SECTION, there are only several values which are important to the reader. The objective function: "OBJ", assembly line capacity: "LINCAP", inventory production level: "INVEN", and the total number of upenders required: "UPENDER", represent the key elements of the model. At a glance, one can see the total cost of changing the production level and whether or not the imposed constraints were met. The solution of the COLUMN SECTION, however, gives the detailed analysis of the solution of the model. The naming convention of the column section is slightly different than in the row section. The name column combines a two letter mold code with a two letter section identification and a mold size.

The two letter section identification can be described as follows:

1. II - integer number of molds required for a mold type
2. UP - number of upenders assigned to a mold type
3. AA - number of molds to be added to a mold type
4. SS - number of molds to be subtracted from a mold type
5. FF - number of molds to be framed for a mold type

```
                          SOLUTION OF THE COLUMN SECTION

ACTIVITY      ICOST        LLIMIT       ULIMIT       RCOST        NUMBER        STATUS     NAME

14.0000       2.9700       5.0000       41.0000      0.0000       133.0000      IV         ACIIS
 6.0000       5.8700       6.0000       19.0000      14.5000      134.0000      IV         IPIIS
 6.0000       5.8700       5.0000       19.0000      0.0000       135.0000      IV         YLIIS
 2.0000       7.3700       2.0000        2.0000      91.0000      136.0000      IV         AHIIL
 1.0000       6.5000       1.0000        4.0000      7.6300       137.0000      IV         AJIIL
 3.0000       5.6200       3.0000        9.0000      13.2500      139.0000      IV         ANIIL
 1.0000       6.3700       1.0000        9.0000      0.0000       140.0000      IV         AOIIS
 1.0000       5.3700       0.0000        5.0000      0.0000       141.0000      IV         AYIIS
 2.0000       5.3700       2.0000        4.0000      14.0000      142.0000      IV         AZIIS
 8.0000       6.8700       8.0000       21.0000      15.5000      143.0000      IV         AWIIL
13.0000       4.7500       0.0000       21.0000      0.0000       144.0000      IV         AXIIS
11.0000       5.3700      11.0000       21.0000      14.0000      145.0000      IV         AVIIS
 1.0000       4.3700       1.0000        1.0000      73.0000      146.0000      IV         BGIIS
 1.0000       4.8700       1.0000        2.0000      88.5000      147.0000      IV         BHIIS
 5.0000       7.8700       4.0000        6.0000      1.5000       148.0000      IV         BJIIL
 5.0000       5.7500       2.0000        6.0000      0.0000       149.0000      IV         BKIIS
 3.0000       5.7500       3.0000        6.0000      89.3800      150.0000      IV         KRIIS
 3.0000       5.7500       3.0000        6.0000      89.3800      151.0000      IV         KLIIS
15.0000       5.6200       9.0000       23.0000      0.0000       153.0000      IV         MMIIS
15.0000       4.0000      15.0000       22.0000      12.6300      154.0000      IV         VVIIS
 1.0000       6.1200       1.0000        5.0000      14.7500      155.0000      IV         IIIIS
32.0000       6.1200       4.0000       50.0000      0.0000       156.0000      IV         BRIIL
32.0000       6.3700       6.0000       63.0000      0.0000       157.0000      IV         BNIIS

 2.0000       1.0000       0.0000        6.0000      1.0000       158.0000      IV         ACUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       159.0000      IV         YRUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       160.0000      IV         YLUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       161.0000      IV         AHUPL
 1.0000       1.0000       0.0000        6.0000      1.0000       162.0000      IV         AJUPL
 1.0000       1.0000       0.0000        6.0000      1.0000       164.0000      IV         ANUPL
 1.0000       1.0000       0.0000        6.0000      1.0000       165.0000      IV         AOUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       166.0000      IV         AYUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       167.0000      IV         AZUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       168.0000      IV         AWUPL
 1.0000       1.0000       0.0000        6.0000      1.0000       169.0000      IV         AXUPS
 2.0000       1.0000       0.0000        6.0000      1.0000       170.0000      IV         AVUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       171.0000      IV         BGUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       172.0000      IV         BHUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       173.0000      IV         BJUPL
 1.0000       1.0000       0.0000        6.0000      1.0000       174.0000      IV         BKUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       175.0000      IV         KRUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       176.0000      IV         KLUPS
 3.0000       1.0000       0.0000        6.0000      1.0000       178.0000      IV         MMUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       179.0000      IV         VVUPS
 1.0000       1.0000       0.0000        6.0000      1.0000       180.0000      IV         IIUPS
 3.0000       1.0000       0.0000        6.0000      1.0000       181.0000      IV         BRUPL
 3.0000       1.0000       0.0000        6.0000      1.0000       182.0000      IV         BNUPS

 1.0000       7.5000       0.0000       20.0000      0.0000       184.0000      IV         YRAAS
 1.0000       7.5000       0.0000       20.0000      0.0000       186.0000      IV         AHAAL
 1.0000       7.5000       0.0000       20.0000      0.0000       189.0000      IV         ANAAL
 1.0000       7.5000       0.0000       20.0000      0.0000       192.0000      IV         AZAAS
 2.0000       7.5000       0.0000       20.0000      0.0000       193.0000      IV         AWAAL
 6.0000       7.5000       0.0000       20.0000      0.0000       195.0000      IV         AVAAS
 1.0000       7.5000       0.0000       20.0000      0.0000       197.0000      IV         BHAAS
 2.0000       7.5000       0.0000       20.0000      0.0000       200.0000      IV         KRAAS
 2.0000       7.5000       0.0000       20.0000      0.0000       201.0000      IV         KLAAS
 7.0000       7.5000       0.0000       20.0000      0.0000       204.0000      IV         VVAAS
 1.0000       7.5000       0.0000       20.0000      0.0000       205.0000      IV         IIAAS

 9.0000       7.5000       0.0000       40.0000      0.2500       231.0000      IV         BRSSL
11.0000       7.5000       0.0000       40.0000      0.0000       232.0000      IV         BNSSS

 1.0000      75.0000       0.0000        2.0000      0.0000       247.0000      IV         BHFFS
 1.0000      75.0000       0.0000        2.0000      0.0000       250.0000      IV         KRFFS
```

Figure 10. MIP Standard Format "SOLUTION"

For the example in Figure 10, the name: "BNIIS" tells the plant that 32.0 small "BN" molds are required to satisfy the production inventory and shipping requirements. "ACUPS" implies that 2.0 upenders MUST be assigned to mold type "AC". Finally, "BHAAS" states that 1 small "BH" mold must be added to the current assembly line configuration.

The scheduling and sequencing subroutine tries to determine the problem mold in the specific problem area on the assembly line. the model would remove this mold and replace it with the largest wire load difficulty mold which staisfies the sequencing constraints from the set of ADD molds. At all times, the intent of the model is not to exceed some specified wire load difficulty average (say 27 as a first attempt). There will be molds, however, which cannot be added or removed from the set of problem areas on the assembly line. Therefore, the model arbitrarily starts at some mold position (typically position 60), and hunts on either side of this mold position until the set of SUBTRACTED molds is satisfied.

At this point, we have solved many of the problem areas, and at least know the set of mold positions which must be changed. This strategy insures that the minimum number of mold changes occur, and that these mold changes are as close as possible to one another. We also know the set of molds which have to be added to the assembly line at this point. The model systematically places the required ADD molds into positions, keeping in mind the assembly line constraints. If the assembly line cannot be satisfied at the stated wire load difficulty, the model increases the difficulty and tries to solve the sequencing problem again.

Figure 11, shows the intermediate results of the scheduling and sequencing algorithm. The four columns represent the position of the mold changes, the molds to be added, subtracted, and the wire load difficulty needed to satisfy that particular mold position. For example, if the model was unable to schedule a particular mold type, this sheet would explain which molds caused the trouble, and what mold positions could not be filled. It represents temporary output, but also an alternative schedule if the plant can tolerate the scheduling violations.

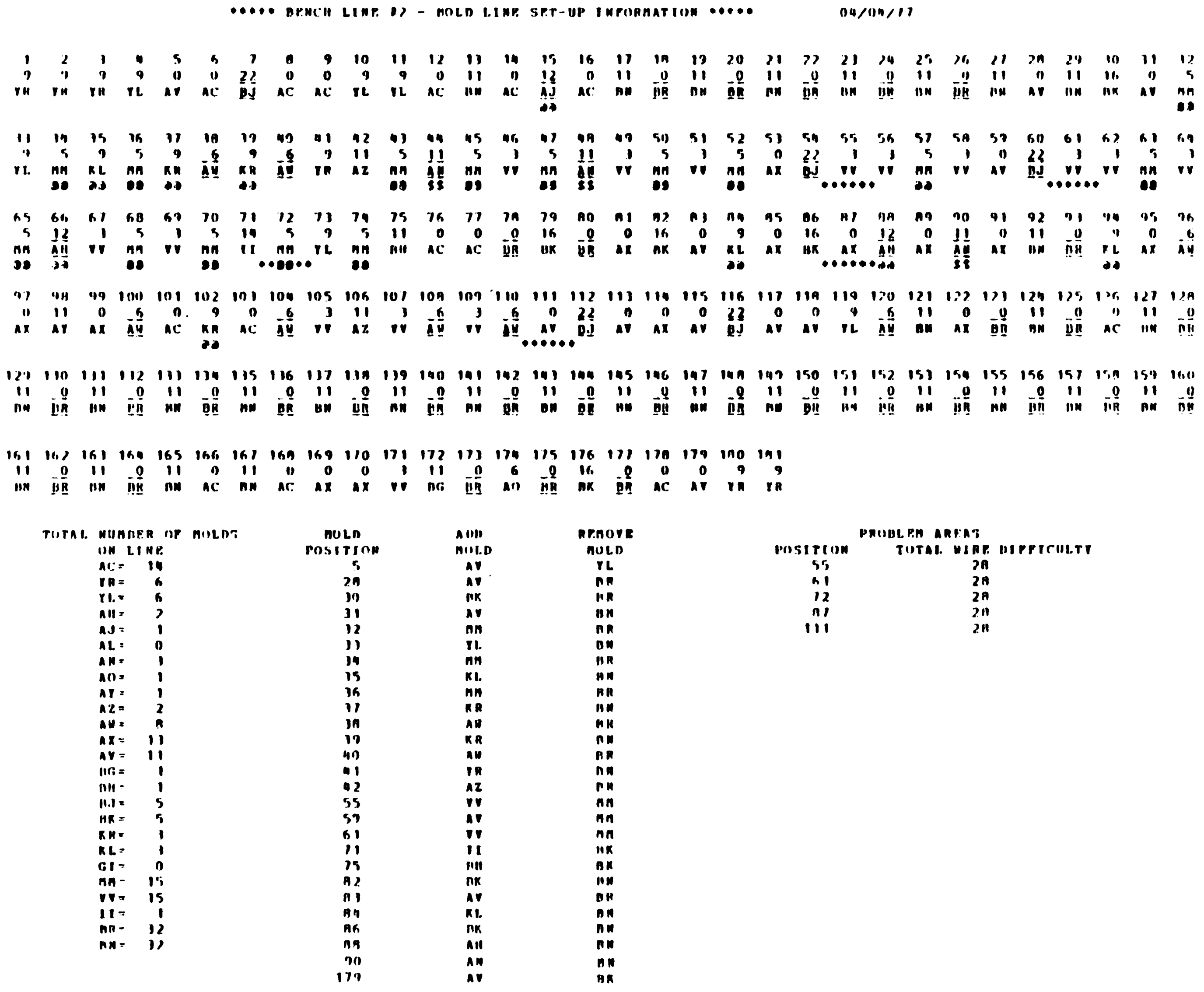
The assembly line scheduling model and in particular the scheduling subroutine, tries to take into account the human factors of the assembly line. Up to this point, it has been told that an average wire load difficulty of 27 is acceptable at the plant. So the question arises during the scheduling program; what is the upper limit of wire load difficulty that an average assembly line worker can handle without falling behind?

```
4 ICNT=    0   AVAIL  SDIFF

 4 POS   ADD  SUB   NEED
     59       AV     MM        1
     61       VV     MM        3
     55       VV     MM        3
     71       II     BK       14
     75       BH     BK       14
     42       AZ     BR       12
     41       YR     BN       12
     40       AW     BR        8
     39       KR     BN       13
     38       AW     BR        8
     82       BK     BN       17
     83       AV     BR        3
     37       KR     BN       13
     36       MM     BR        8
     84       KL     BN       12
     35       KL     BN       14
     34       MM     BR        8
     86       BK     BN       17
     33       YL     BN       14
     32       MM     BR        8
     88       AH     BN       12
     31       AV     BN       14
     30       BK     BR       17
     90       AN     BN       16
     28       AV     BR        1
      5       AV     YL        6
    179       AV     BK       10



THE SCHEDULING & SEQUENCING ALGORITHM WAS SOLVED WITH

    AN AVERAGE WIRE LOAD DIFFICULTY OF  28
```

Figure 11. Intermediate Scheduling Results

This is an important consideration, since a person must perform the work. An average wire load difficulty of 33 might be acceptable for one or two positions on the assembly line, but is cannot be maintained for any long period of time. The model considers this fact, and even though there are problem areas, the model tends to help spread out the difficult positions. In other words, the assembly line model considers the human aspects of producing foam parts, and tries not to overload the amount of work the wire load team can accomplish. Previous models have just considered the wire load difficulty average as a number, and although answers were obtained, they were far from being feasible to implement.

Figure 12, on the following page shows the final mold line-up summary sheet. It summarizes the positions and the molds to be added or removed from the assembly line. It also describes the problem areas which resulted after implementing the new line-up and shows the new configuration of molds in their proper positions. In other words, this figure represents the final mold scheduling solution and future mold change line-up of the foam assembly line.



```
                    ***** BENCH LINE #2 - MOLD LINE SET-UP INFORMATION *****        04/04/77

  1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
  9    9    9    9    0    0   22    0    0    9    9    0   11    0   12    0   11    0   11    0   11    0   11    0   11    0   11    0   11   16    0    5
 YR   YR   YR   YL   AV   AC   BJ   AC   AC   YL   YL   AC   BN   AC   AJ   AC   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   AV   BN   BK   AV   MM
                                                                        @@                                                                                88

 33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
  9    5    9    5    9    6    9    6    9   11    5   11    5    3    5   11    3    5    3    5    0   22    3    3    5    3    0   22    3    3    5    3
 YL   MM   KL   MM   KR   AW   KR   AW   YR   AZ   MM   AH   MM   VV   MM   AH   VV   MM   VV   MM   AX   BJ   VV   VV   MM   VV   AV   BJ   VV   VV   MM   VV
      88   @@   88   @@        @@                  88   $$   88        88   $$        88        88             ******     @@                  ******     88

 65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96
  5   12    3    5    3    5   14    5    9    5   11    0    0    0   16    0    0   16    0    9    0   16    0   12    0   11    0   11    0    9    0    6
 MM   AH   VV   MM   VV   MM   II   MM   YL   MM   BH   AC   AC   BR   BK   BR   AX   BK   AV   KL   AX   BK   AX   AH   AX   AH   AX   BN   BR   KL   AX   AW
 88   @@        88        88       **88**        88                                               @@            *******@@       $$                  @@

 97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128
  0   11    0    6    0    9    0    6    3   11    3    6    3    6    0   22    0    0    0   22    0    0    9    6   11    0    0   11    0    0   11    0
 AX   AY   AX   AW   AC   KR   AC   AW   VV   AZ   VV   AW   VV   AW   AV   BJ   AV   AX   AV   BJ   AV   AV   YL   AW   BN   AX   BR   BN   BR   AC   BN   BR
                          @@                                              ******

129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160
 11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0   11    0
 BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR   BN   BR

161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181
 11    0   11    0   11    0   11    0    0    0    3   11    0    6    0   16    0    0    0    9    9
 BN   BR   BN   BR   BN   AC   BN   AC   AX   AX   VV   BG   BR   AO   BR   BK   BR   AC   AV   YR   YR

  TOTAL NUMBER OF MOLDS        MOLD          ADD         REMOVE                PROBLEM AREAS
        ON LINE              POSITION        MOLD         MOLD          POSITION      TOTAL WIRE DIFFICULTY
        AC=   14                 5            AV           YL              55               28
        YR=    6                28            AV           BR              61               28
        YL=    6                30            BK           BR              72               28
        AH=    2                31            AV           BN              87               28
        AJ=    1                32            MM           BR             111               28
        AL=    0                33            YL           BN
        AN=    1                34            MM           BR
        AO=    1                35            KL           BN
        AY=    1                36            MM           BR
        AZ=    2                37            KR           BN
        AW=    8                38            AW           BK
        AX=   13                39            KR           BN
        AV=   11                40            AW           BR
        BG=    1                41            YR           BN
        BH=    1                42            AZ           BR
        BJ=    5                55            VV           MM
        BK=    5                59            AV           MM
        KR=    1                61            VV           MM
        KL=    1                71            II           BK
        GI=    0                75            BH           BK
        MM=   15                82            BK           BN
        VV=   15                83            AV           BR
        II=    1                84            KL           BN
        BR=   32                86            BK           BN
        BN=   32                88            AH           BN
                                90            AN           BN
                               179            AV           BR
```

Figure 12. Final Mold Line-up Summary

4.7 SUMMARY

This chapter has tried to examine the various subroutines which comprise the assembly line mold scheduling model. We examined each subroutine's function and objective, trying to relate each piece to the overall model. The mixed integer programming subroutine and the scheduling and sequencing subroutine described here determine the proper number of molds, and the placement of these molds on the assembly line. Now that the model has solved the assembly line scheduling problem, let us examine what conclusions can be drawn from this.

V. EXPERIMENTS

This chapter will examine an experimental run made by the plant to determine the assembly line sequence of molds. It summarizes in pictorial form the input and output as it actually appears in a production run. The keypoints in the following pages are denoted by a series of numbered circles which can now be described:

1. assembly line header - 80 character title card (Figure 13).

2. parameter input card - includes the total number of mold types, the number of shifts, total number of molds on assembly line, number of currently available positions in warehouse, average starting wire load difficulty (Figure 13).

3. static mold information card - includes the mold code, part number, mold size, wire load difficulty, number of parts per cavity, platform loading, total molds available, total molds framed, tape after post cure, standard pack per basket, production rate per week, value of the part, safety stock (Figure 13).

4. assembly line input mold position summary - the molds are listed in the order they actually appear on the assembly line before re-scheduling (Figure 13).

5. constraint solution - denotes the name and activity of a row, represents the answer of the value indicated (Figure 15).

6. <u>required molds section</u> - denoted by an "II" in the name, this group represents the required number of molds necessary to satisfy the production requirements (Figure 16).

7. <u>required upender section</u> - denoted by an "UP" in the name, this group represents the required number of upenders or baskets needed to pack the foam parts for each mold type (Figure 16).

8. <u>added molds section</u> - denoted by an "AA" in the name, this group represents the number of molds to be added to the assembly line for each mold type (Figure 16).

9. <u>subtracted molds section</u> - denoted by an "SS" in the name, this group represents the number of molds to be removed from the assembly line for each mold type (Figure 16).

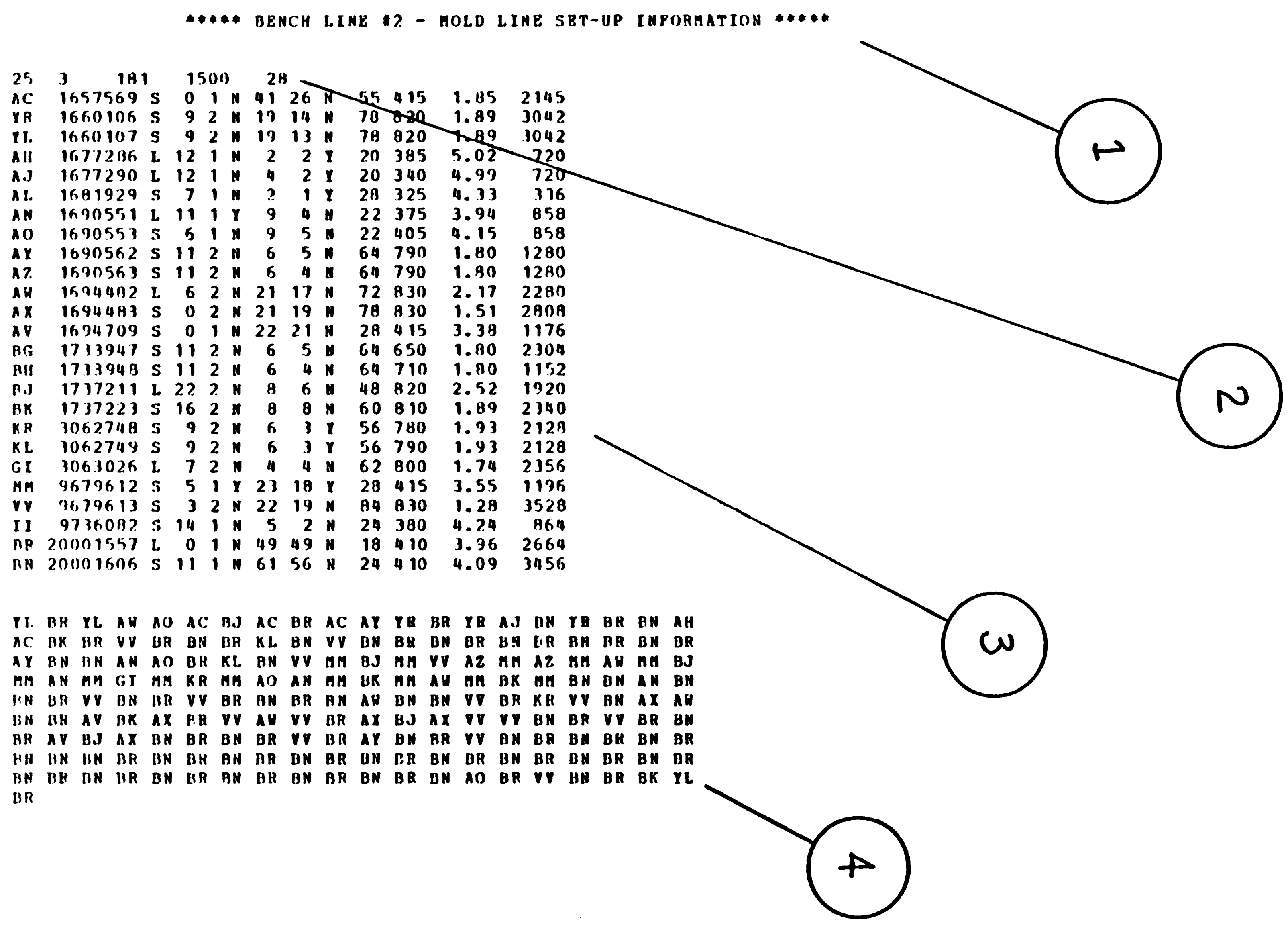
10. <u>available molds array</u> - this section represents a summary of the mold types and their particular wire load difficulties that are available for scheduling at some assembly line wire load difficulty average (Figure 17(a)).

11. <u>current scheduled solution</u> - this section represents the mold position, molds added and subtracted, and the highest wire load difficulty which can be fitted into this slot on the assembly line (Figure 17(a)).

12. <u>solution statement</u> - this temporary printout states that the model has solved the problem with a particular wire load difficulty average (Figure 17(e)).

13. <u>final mold line-up</u> - a summary by mold position showing the line-up of molds on the assembly line once they have been sequenced and scheduled (Figure 18).

14. <u>mold position summary</u> - a listing by mold position of the molds added and subtracted to establish the new line-up (Figure 18).



```
                 ***** BENCH LINE #2 - MOLD LINE SET-UP INFORMATION *****

25   3     181     1500    28
AC   1657569 S  0  1 N 41 26 N  55 415  1.85  2145
YR   1660106 S  9  2 N 19 14 N  78 820  1.89  3042
YL   1660107 S  9  2 N 19 13 N  78 820  1.89  3042
AH   1677286 L 12  1 N  2  2 Y  20 385  5.02   720
AJ   1677290 L 12  1 N  4  2 Y  20 340  4.99   720
AL   1681929 S  7  1 N  2  1 Y  28 325  4.33   316
AN   1690551 L 11  1 Y  9  4 N  22 375  3.94   858
AO   1690553 S  6  1 N  9  5 N  22 405  4.15   858
AY   1690562 S 11  2 N  6  5 N  64 790  1.80  1280
AZ   1690563 S 11  2 N  6  4 N  64 790  1.80  1280
AW   1694482 L  6  2 N 21 17 N  72 830  2.17  2280
AX   1694483 S  0  2 N 21 19 N  78 830  1.51  2808
AV   1694709 S  0  1 N 22 21 N  28 415  3.38  1176
BG   1713947 S 11  2 N  6  5 N  64 650  1.80  2304
BH   1733948 S 11  2 N  6  4 N  64 710  1.80  1152
BJ   1737211 L 22  2 N  8  6 N  48 820  2.52  1920
BK   1737223 S 16  2 N  8  8 N  60 810  1.89  2340
KR   3062748 S  9  2 N  6  3 Y  56 780  1.93  2128
KL   3062749 S  9  2 N  6  3 Y  56 790  1.93  2128
GI   3063026 L  7  2 N  4  4 N  62 800  1.74  2356
MM   9679612 S  5  1 Y 23 18 Y  28 415  3.55  1196
VV   9679613 S  3  2 N 22 19 N  84 830  1.28  3528
II   9736082 S 14  1 N  5  2 N  24 380  4.24   864
BR  20001557 L  0  1 N 49 49 N  18 410  3.96  2664
BN  20001606 S 11  1 N 61 56 N  24 410  4.09  3456


YL BR YL AW AO AC BJ AC BR AC AY YR BR YR AJ BN YR BR BN AH
AC BK BR VV BR BN BR KL BN VV BN BR BN BR BN BR BN BR BN BR
AY BN BN AN AO BR KL BN VV MM BJ MM VV AZ MM AZ MM AW MM BJ
MM AN MM GI MM KR MM AO AN MM BK MM AW MM BK MM BN BN AN BN
BN BR VV BN BR VV BR BN BR BN AW BN BN VV BR KR VV BN AX AW
BN BR AV BK AX BR VV AW VV BR AX BJ AX VV VV BN BR VV BR BN
BR AV BJ AX BN BR BN BR VV BR AY BN BR VV BN BR BN BR BN BR
BN BN BN BR BN BR BN BR BN BR BN BR BN BR BN BR BN BR BN BR
BN BR BN BR BN BR BN BR BN BR BN BR BN AO BR VV BN BR BK YL
BR
```

Figure 13. Sample - Mold Line-up Summary

```
                     FORECASTING - PRODUCTION INFORMATION

         PCD        INITIAL      1ST WEEK     2ND WEEK     3RD WEEK     4TH WEEK     SAFETY
         BAL       INVENTORY     SHIPPED      SHIPPED      SHIPPED      SHIPPED      STOCK

AC       770          990         1760         1980         2640          440        2145
YR       936         1404         2340         2184         3432          624        3042
YL       468         1716         2184         2340         2964         1092        3042
AH       -40          360          320          240          280          200         720
AJ       120          120          240          160          160          120         720
AL       -84          140           56            0           56            0         336
AN       574          878         1452          968         1452          880         858
AO        88         1188         1276         1056         1496          748         858
AY       768          256         1024          768         1024          512        1280
AZ       768          384         1152         1152         1400          768        1280
AW      -866         4754         3888         5760         5184         4320        2280
AX     -4568         8624         4056         6240         4368         4992        2808
AV     -1008         3080         2072         2576         2464         2576        1176
BG      -168          168            0          384          384          384        2304
BH         0            0            0            0            0          128        1152
BJ      1622         1834         3456         3072         3168         3264        1920
BK      2400         1080         3480         3000         3240         3120        2340
KR     -1456         2464         1008         1232         1120         1008        2128
KL     -1456         2576         1120         1232         1120         1008        2128
GT       372          372          744         1116          620          372        2356
MM     -1470         4902         3432         5980         4680         5928        1196
VV      4302         3258         7560        11700         9720        11880        3528
II      -225          225            0          192          192          144         864
BR     17406         2333        19739        16992        18180        17460        2664
BN     18500         1276        19776        17088        18336        17856        3456

  TOTAL PARTS SHIPPED IN TWO WEEKS =     169547
  TOTAL INITIAL INVENTORY =      44382
```

Figure 14. Sample - Production Information

```
                              SOLUTION OF THE ROW SECTION

  ACTIVITY       SLACK          LLIMIT        ULIMIT         DUALACT      NUMBER        STATUS     NAME

 1284.2500    -1284.2500        0.0000        0.0000         1.0000       1.0000         BS       OBJ
    7.0000       34.0000        0.0000       41.0000         0.0000       2.0000         BS       ACREQ
    4.0000       15.0000        0.0000       19.0000         0.0000       3.0000         BS       YRREQ
    4.0000       15.0000        0.0000       19.0000         0.0000       4.0000         BS       YLREQ
    1.0000        1.0000        0.0000        2.0000         0.0000       5.0000         BS       AHREQ
    2.0000        2.0000        0.0000        4.0000         0.0000       6.0000         BS       AJREQ
    0.0000        2.0000        0.0000        2.0000        -6.8929       7.0000         LL       ALREQ
    3.0000        6.0000        0.0000        9.0000         0.0000       8.0000         BS       ANREQ
    2.0000        7.0000        0.0000        9.0000         0.0000       9.0000         BS       AOREQ
    3.0000        3.0000        0.0000        6.0000         0.0000      10.0000         BS       AYREQ
    2.0000        4.0000        0.0000        6.0000         0.0000      11.0000         BS       AZREQ
    2.0000       19.0000        0.0000       21.0000         0.0000      12.0000         BS       AWREQ
    5.0000       16.0000        0.0000       21.0000         0.0000      13.0000         BS       AXREQ
    2.0000       20.0000        0.0000       22.0000         0.0000      14.0000         BS       AVREQ
    3.0000        3.0000        0.0000        6.0000         0.0000      15.0000         BS       BGREQ
    1.0000        5.0000        0.0000        6.0000         0.0000      16.0000         BS       BHREQ
    4.0000        4.0000        0.0000        8.0000         0.0000      17.0000         BS       BJREQ
    5.0000        3.0000        0.0000        8.0000         0.0000      18.0000         BS       BKREQ
    0.0000        6.0000        0.0000        6.0000         0.0000      19.0000         BS       KRREQ
    0.0000        6.0000        0.0000        6.0000         0.0000      20.0000         BS       KLREQ
    3.0000        1.0000        0.0000        4.0000         0.0000      21.0000         BS       GIREQ
   12.0000       11.0000        0.0000       23.0000         0.0000      22.0000         BS       MMREQ
   16.0000        6.0000        0.0000       22.0000         0.0000      23.0000         BS       VVREQ
    1.0000        4.0000        0.0000        5.0000         0.0000      24.0000         BS       IIREQ
   48.0000        1.0000        0.0000       49.0000         0.0000      25.0000         BS       HRREQ
   51.0000       10.0000        0.0000       61.0000         0.0000      26.0000         BS       HNREQ
    4.0000        0.0000        0.0000        4.0000         7.5000      27.0000         UL       ACADD
    3.0000        0.0000        0.0000        3.0000         7.5000      28.0000         UL       YRADD
    3.0000        0.0000        0.0000        3.0000         7.5000      29.0000         UL       YLADD
    1.0000        0.0000        0.0000        1.0000         7.5000      30.0000         UL       AHADD
    1.0000        0.0000        0.0000        1.0000         7.5000      31.0000         UL       AJADD
    0.0000        0.0000        0.0000        0.0000         0.0000      32.0000         BS       ALADD
    3.0000        1.0000        0.0000        4.0000         0.0000      33.0000         BS       ANADD
    2.0000        2.0000        0.0000        4.0000         0.0000      34.0000         BS       AOADD
    3.0000        0.0000        0.0000        3.0000         0.0000      35.0000         BS       AYADD
    2.0000        0.0000        0.0000        2.0000         0.0000      36.0000         BS       AZADD
    2.0000        4.0000        0.0000        6.0000         0.0000      37.0000         BS       AWADD
    5.0000        0.0000        0.0000        5.0000         1.3700      38.0000         UL       AXADD
    2.0000        0.0000        0.0000        2.0000         0.7500      39.0000         UL       AVADD
    0.0000        0.0000        0.0000        0.0000         7.5000      40.0000         UL       BGADD
    0.0000        0.0000        0.0000        0.0000         7.5000      41.0000         UL       BHADD
    4.0000        1.0000        0.0000        5.0000         0.0000      42.0000         BS       BJADD
    5.0000        0.0000        0.0000        5.0000         7.5000      43.0000         UL       BKADD
    0.0000        2.0000        0.0000        2.0000         0.0000      44.0000         BS       KRADD
    0.0000        2.0000        0.0000        2.0000         0.0000      45.0000         BS       KLADD
    1.0000        0.0000        0.0000        1.0000         7.5000      46.0000         UL       GIADD
   12.0000        1.0000        0.0000       13.0000         0.0000      47.0000         BS       MMADD
   16.0000        0.0000        0.0000       16.0000         2.1200      48.0000         UL       VVADD
    0.0000        0.0000        0.0000        0.0000         7.5000      49.0000         UL       IIADD
   48.0000        1.0000        0.0000       49.0000         0.0000      50.0000         BS       HRADD
   50.0000        0.0000        0.0000       50.0000         7.5000      51.0000         UL       HNADD
    7.0000       -3.0000        4.0000        0.0000         0.0000      52.0000         BS       ACSUM
    4.0000       -1.0000        3.0000        0.0000         0.0000      53.0000         BS       YRSUM
    4.0000       -3.0000        1.0000        0.0000         0.0000      54.0000         BS       YLSUM
    1.0000        0.0000        1.0000        0.0000         0.0000      55.0000         BS       AHSUM
```

Figure 15. Sample - MIP Row Solution

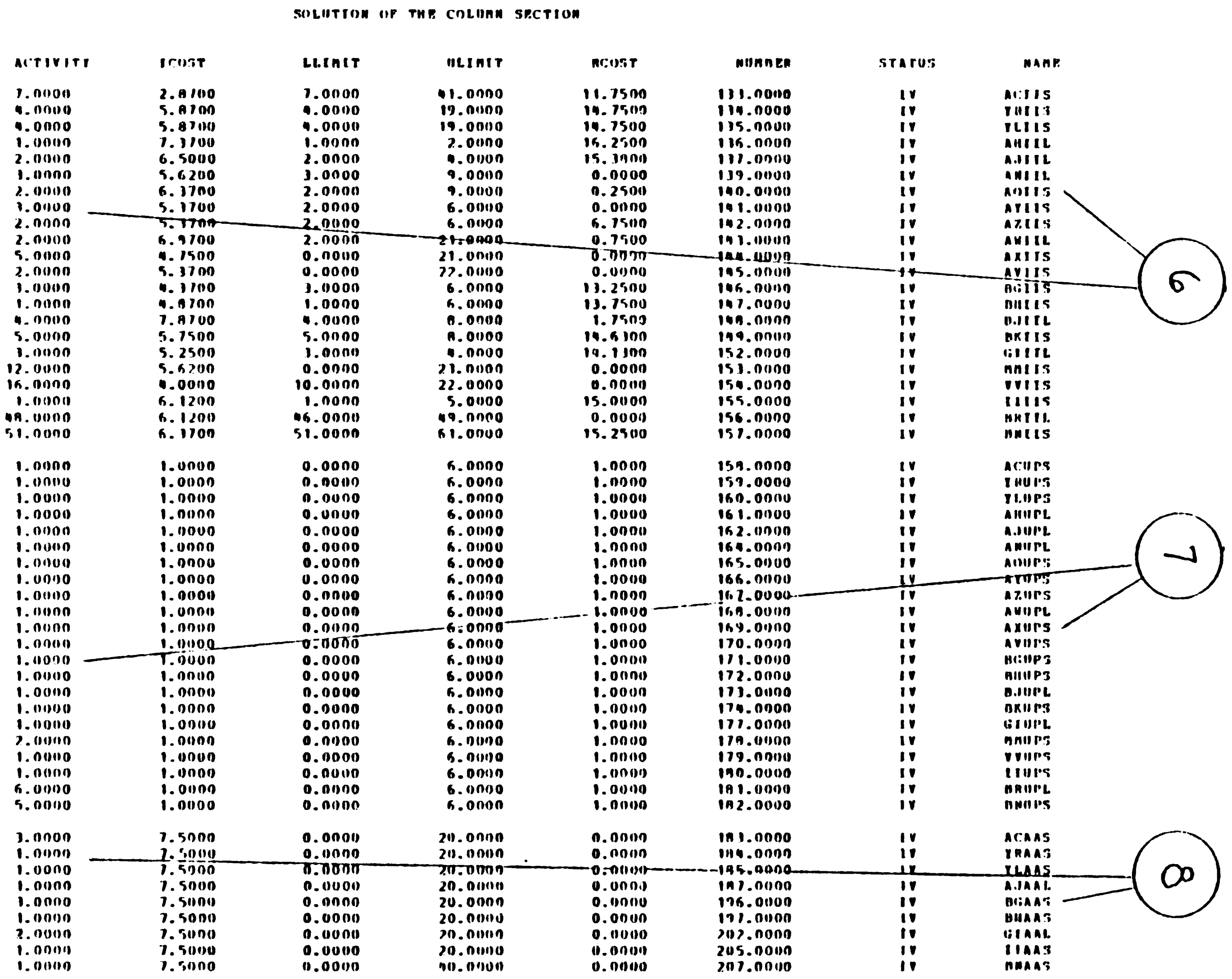
2.0000	-1.0000	1.0000	0.0000	0.0000	56.0000	BS	AJSUB
0.0000	0.0000	0.0000	0.0000	0.0000	57.0000	BS	ALSUB
4.0000	0.0000	4.0000	0.0000	-7.0000	58.0000	LL	ANSUB
4.0000	0.0000	4.0000	0.0000	-7.5000	59.0000	LL	AOSUB
3.0000	0.0000	3.0000	0.0000	-6.7500	60.0000	LL	AYSUB
2.0000	0.0000	2.0000	0.0000	0.0000	61.0000	BS	AZSUB
6.0000	0.0000	6.0000	0.0000	-7.5000	62.0000	LL	AWSUB
5.0000	0.0000	5.0000	0.0000	-7.5000	63.0000	LL	AXSUB
2.0000	0.0000	2.0000	0.0000	-7.5000	64.0000	LL	AVSUB
3.0000	-3.0000	0.0000	0.0000	0.0000	65.0000	BS	BGSUB
1.0000	-1.0000	0.0000	0.0000	0.0000	66.0000	BS	BHSUB
5.0000	0.0000	5.0000	0.0000	-7.5000	67.0000	LL	BJSUB
5.0000	0.0000	5.0000	0.0000	0.0000	68.0000	BS	BKSUB
2.0000	0.0000	2.0000	0.0000	-7.2014	69.0000	LL	KRSUB
2.0000	0.0000	2.0000	0.0000	-7.2014	70.0000	LL	KLSUB
3.0000	-2.0000	1.0000	0.0000	0.0000	71.0000	BS	GISUB
13.0000	0.0000	13.0000	0.0000	-7.0000	72.0000	LL	MMSUB
16.0000	0.0000	16.0000	0.0000	-7.5000	73.0000	LL	VVSUB
1.0000	-1.0000	0.0000	0.0000	0.0000	74.0000	BS	IISUB
49.0000	0.0000	49.0000	0.0000	-7.5000	75.0000	LL	BRSUB
51.0000	-1.0000	50.0000	0.0000	0.0000	76.0000	BS	BNSUB
7.0000	19.0000	0.0000	26.0000	0.0000	77.0000	BS	ACFRM
4.0000	10.0000	0.0000	14.0000	0.0000	78.0000	BS	YRFRM
4.0000	9.0000	0.0000	13.0000	0.0000	79.0000	BS	YLFRM
1.0000	1.0000	0.0000	2.0000	0.0000	80.0000	BS	AHFRM
2.0000	0.0000	0.0000	2.0000	0.0000	81.0000	BS	AJFRM
0.0000	1.0000	0.0000	1.0000	0.0000	82.0000	BS	ALFRM
3.0000	1.0000	0.0000	4.0000	0.0000	83.0000	BS	ANFRM
2.0000	3.0000	0.0000	5.0000	0.0000	84.0000	BS	AOFRM
3.0000	2.0000	0.0000	5.0000	0.0000	85.0000	BS	AYFRM
2.0000	2.0000	0.0000	4.0000	0.0000	86.0000	BS	AZFRM
2.0000	15.0000	0.0000	17.0000	0.0000	87.0000	BS	AWFRM
5.0000	14.0000	0.0000	19.0000	0.0000	88.0000	BS	AXFRM
2.0000	19.0000	0.0000	21.0000	0.0000	89.0000	BS	AVFRM
3.0000	2.0000	0.0000	5.0000	0.0000	90.0000	BS	BGFRM
1.0000	3.0000	0.0000	4.0000	0.0000	91.0000	BS	BHFRM
4.0000	2.0000	0.0000	6.0000	0.0000	92.0000	BS	BJFRM
5.0000	3.0000	0.0000	8.0000	0.0000	93.0000	BS	BKFRM
0.0000	3.0000	0.0000	3.0000	0.0000	94.0000	BS	KRFRM
0.0000	3.0000	0.0000	3.0000	0.0000	95.0000	BS	KLFRM
1.0000	1.0000	0.0000	4.0000	0.0000	96.0000	BS	GIFRM
12.0000	6.0000	0.0000	18.0000	0.0000	97.0000	BS	MMFRM
16.0000	3.0000	0.0000	19.0000	0.0000	98.0000	BS	VVFRM
1.0000	1.0000	0.0000	2.0000	0.0000	99.0000	BS	IIFRM
48.0000	1.0000	0.0000	49.0000	0.0000	100.0000	BS	BAFRM
51.0000	5.0000	0.0000	56.0000	0.0000	101.0000	BS	BNFRM
0.4911	-0.4911	0.0000	0.0000	0.0000	102.0000	BS	ACEND
0.7948	-0.7948	0.0000	0.0000	0.0000	103.0000	BS	YREND
0.7948	-0.7948	0.0000	0.0000	0.0000	104.0000	BS	YLEND
0.8000	-0.8000	0.0000	0.0000	0.0000	105.0000	BS	AHEND
0.6000	-0.6000	0.0000	0.0000	0.0000	106.0000	BS	AJEND
0.0000	0.0000	0.0000	0.0000	-1.0000	107.0000	LL	ALEND
0.4546	-0.4546	0.0000	0.0000	0.0000	108.0000	BS	ANEND
0.6164	-0.6164	0.0000	0.0000	0.0000	109.0000	BS	AOEND
0.8125	-0.8125	0.0000	0.0000	0.0000	110.0000	BS	AYEND
0.8750	-0.8750	0.0000	0.0000	0.0000	111.0000	BS	AZEND
0.8889	-0.8888	0.0000	0.0000	0.0000	112.0000	BS	AWEND
0.7415	-0.7415	0.0000	0.0000	0.0000	113.0000	BS	AXEND
0.7142	-0.7142	0.0000	0.0000	0.0000	114.0000	BS	AVEND
0.8125	-0.8125	0.0000	0.0000	0.0000	115.0000	BS	BGEND

Figure 15. (continued)

```
    0.9375     -0.9375      0.0000       0.0000      0.0000    116.0000     BS     BBEND
    0.6668     -0.6668      0.0000       0.0000      0.0000    117.0000     BS     BJEND
    0.6665     -0.6665      0.0000       0.0000      0.0000    118.0000     BS     BKEND
    0.0000      0.0000      0.0000       0.0000     -1.0000    119.0000     LL     KPEND
    0.0000      0.0000      0.0000       0.0000     -1.0000    120.0000     LL     KLEND
    0.8065     -0.8065      0.0000       0.0000      0.0000    121.0000     BS     GIEND
    0.2852     -0.2852      0.0000       0.0000      0.0000    122.0000     BS     MMEND
    0.2188     -0.2188      0.0000       0.0000      0.0000    123.0000     BS     VVEND
    0.8333     -0.8333      0.0000       0.0000      0.0000    124.0000     BS     IIEND
    0.6672     -0.6672      0.0000       0.0000      0.0000    125.0000     BS     NHEND
    0.7517     -0.7517      0.0000       0.0000      0.0000    126.0000     BS     BNEND
  181.0000      0.0000    181.0000     181.0000      1.3000    127.0000     EQ     LINCAP
    8.0000      1.0000      0.0000       5.0000      0.0000    128.0000     BS     SMALL
   -8.0000      9.0000      0.0000       5.0000      0.0000    129.0000     BS     LARGE
    3.0000      2.0000      0.0000       5.0000      0.0000    130.0000     BS     WIRE
 5955.5700   1047.8300      0.0000    7001.0000      0.0000    131.0000     BS     INVEN
   32.0000      1.0000      0.0000      11.0000      0.0000    132.0000     BS     BCENDER
```

5

Figure 15. (continued)



```
                          SOLUTION OF THE COLUMN SECTION

ACTIVITY      ICOST       LLIMIT      ULIMIT      RCOST       NUMBER      STATUS    NAME

 7.0000      2.8700       7.0000     41.0000     11.7500     131.0000      IV       ACIIS
 4.0000      5.8700       4.0000     19.0000     14.7500     134.0000      IV       TRIIS
 4.0000      5.8700       4.0000     19.0000     14.7500     135.0000      IV       YLIIS
 1.0000      7.3700       1.0000      2.0000     16.2500     136.0000      IV       AHIIL
 2.0000      6.5000       2.0000      4.0000     15.3900     137.0000      IV       AJIIL
 1.0000      5.6200       3.0000      9.0000      0.0000     139.0000      IV       ANIIL
 2.0000      6.3700       2.0000      9.0000      0.2500     140.0000      IV       AOIIS
 3.0000      5.3700       2.0000      6.0000      0.0000     141.0000      IV       AYIIS
 2.0000      5.3700       2.0000      6.0000      6.7500     142.0000      IV       AZIIS
 2.0000      6.9700       2.0000     21.0000      0.7500     143.0000      IV       AWIIL
 5.0000      4.7500       0.0000     21.0000      0.0000     144.0000      IV       AXIIS
 2.0000      5.3700       0.0000     22.0000      0.0000     145.0000      IV       AVIIS
 3.0000      4.3700       3.0000      6.0000     13.2500     146.0000      IV       BGIIS
 1.0000      4.8700       1.0000      6.0000     13.7500     147.0000      IV       BHIIS
 4.0000      7.8700       4.0000      8.0000      1.7500     148.0000      IV       BJIIL
 5.0000      5.7500       5.0000      8.0000     14.6100     149.0000      IV       BKIIS
 3.0000      5.2500       3.0000      4.0000     14.1300     152.0000      IV       GIIIL
12.0000      5.6200       0.0000     21.0000      0.0000     153.0000      IV       MMIIS
16.0000      4.0000      10.0000     22.0000      0.0000     154.0000      IV       VVIIS
 1.0000      6.1200       1.0000      5.0000     15.0000     155.0000      IV       IIIIS
48.0000      6.1200      46.0000     49.0000      0.0000     156.0000      IV       MRIIL
51.0000      6.3700      51.0000     61.0000     15.2500     157.0000      IV       NNIIS

 1.0000      1.0000       0.0000      6.0000      1.0000     158.0000      IV       ACUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     159.0000      IV       TRUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     160.0000      IV       YLUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     161.0000      IV       AHUPL
 1.0000      1.0000       0.0000      6.0000      1.0000     162.0000      IV       AJUPL
 1.0000      1.0000       0.0000      6.0000      1.0000     164.0000      IV       ANUPL
 1.0000      1.0000       0.0000      6.0000      1.0000     165.0000      IV       AOUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     166.0000      IV       AYUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     167.0000      IV       AZUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     168.0000      IV       AWUPL
 1.0000      1.0000       0.0000      6.0000      1.0000     169.0000      IV       AXUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     170.0000      IV       AVUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     171.0000      IV       BGUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     172.0000      IV       BHUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     173.0000      IV       BJUPL
 1.0000      1.0000       0.0000      6.0000      1.0000     174.0000      IV       BKUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     177.0000      IV       GIUPL
 2.0000      1.0000       0.0000      6.0000      1.0000     178.0000      IV       MMUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     179.0000      IV       VVUPS
 1.0000      1.0000       0.0000      6.0000      1.0000     180.0000      IV       IIUPS
 6.0000      1.0000       0.0000      6.0000      1.0000     181.0000      IV       MRUPL
 5.0000      1.0000       0.0000      6.0000      1.0000     182.0000      IV       NNUPS

 3.0000      7.5000       0.0000     20.0000      0.0000     183.0000      IV       ACAAS
 1.0000      7.5000       0.0000     20.0000      0.0000     184.0000      IV       TRAAS
 1.0000      7.5000       0.0000     20.0000      0.0000     185.0000      IV       YLAAS
 1.0000      7.5000       0.0000     20.0000      0.0000     187.0000      IV       AJAAL
 1.0000      7.5000       0.0000     20.0000      0.0000     196.0000      IV       BGAAS
 1.0000      7.5000       0.0000     20.0000      0.0000     197.0000      IV       BHAAS
 2.0000      7.5000       0.0000     20.0000      0.0000     202.0000      IV       GIAAL
 1.0000      7.5000       0.0000     20.0000      0.0000     205.0000      IV       IIAAS
 1.0000      7.5000       0.0000     40.0000      0.0000     207.0000      IV       NNAAS
```

Figure 16. Sample - MIP Column Solution

```
1.0000      7.5000      0.0000      20.0000      0.5000      214.0000      IV      ANSSL
2.0000      7.5000      0.0000      20.0000      0.0000      215.0000      IV      AOSSS
4.0000      7.5000      0.0000      20.0000      0.0000      219.0000      IV      AVSSL
1.0000      7.5000      0.0000      20.0000      0.0000      223.0000      IV      BJSSL
2.0000      7.5000      0.0000      20.0000      0.2986      225.0000      IV      KRSSS
2.0000      7.5000      0.0000      20.0000      0.2986      226.0000      IV      KLSSS
1.0000      7.5000      0.0000      20.0000      0.5000      228.0000      IV      MNSSS
1.0000      7.5000      0.0000      40.0000      0.0000      211.0000      IV      MNSSL
```

9

Figure 16. (continued)

```
4 ICNT=    7  AVAIL  SDIFF
    AJ    12
    AW    11
    BH    11
    BN    11
    BN    11
    YL     9
    YR     9

 4 POS   ADD   SUB   NEED
     60     BN     BJ      12
     58     BN     AW      12
     66     BK     KR      16
     68            AO       7
     69            AN       7
     50     AC     MM       1
     73     GI     AW       7
     47     II     KL      14
     46     AC     BR       3
     45     AC     AO       6
     75     BG     BK      12
     42            BN       6
     79            AN       6
     91            AW       6
     28     BG     KL      14
     96     BG     KR      14
    100     AJ     AW      17
     15     GI     AJ       8
    125            BN       6
    142            BN       6

THE SCHEDULING PROBLEM CANNOT BE SOLVED WITH AN AVERAGE WIRE LOAD DIFFICULTY OF  28
LWD IS NOW BEING SET TO  29
```

10

11

Figure 17(a). Sample - Intermediate Results (LWD 28)

```
4 ICNT=    6  AVAIL  SDIFF
    AJ    12
    AN    11
    BH    11
    BN    11
    BN    11
    YR     9

 4 POS  ADD  SUB  NEED
     60     BN     BJ      13
     58     BN     AW      13
     66     BK     KR      17
     68            AO       8
     69            AN       8
     50     AC     MM       2
     73     GI     AW       8
     47     II     KL      15
     46     AC     BR       4
     45     AC     AO       7
     75     BG     BK      13
     42     GI     BN       7
     79            AN       7
     91            AW       7
     28     BG     KL      15
     96     BG     KR      15
    100     AJ     AW      18
     15     YL     AJ       9
    125            BN       7
    142            BN       7

 THE SCHEDULING PROBLEM CANNOT BE SOLVED WITH AN AVERAGE WIRE LOAD DIFFICULTY OF  29
 LWD IS NOW BEING SET TO  30
```

Figure 17(b). Sample - Intermediate Results (LWD 29)

```
4 ICNT=    5  AVAIL  SDIFF
    AJ    12
    AN    11
    BH    11
    BN    11
    BN    11

 4 POS  ADD  SUB  NEED
     60     II     BJ      14
     58     BN     AW      11
     66     BK     KR      18
     68            AO       9
     69            AN       9
     50     AC     MM       3
     73     YL     AW       9
     47     BN     KL      16
     46     GI     BR       8
     45     AC     AO       8
     75     BG     BK      14
     42     GI     BN       8
     79     AC     AN       8
     91            AW       8
     28     BG     KL      16
     96     BG     KR      16
    100     AJ     AW      19
     15     YR     AJ      10
    125            BN       8
    142            BN       8

 THE SCHEDULING PROBLEM CANNOT BE SOLVED WITH AN AVERAGE WIRE LOAD DIFFICULTY OF  30
 LWD IS NOW BEING SET TO  31
```

Figure 17(c). Sample - Intermediate Results (LWD 30)

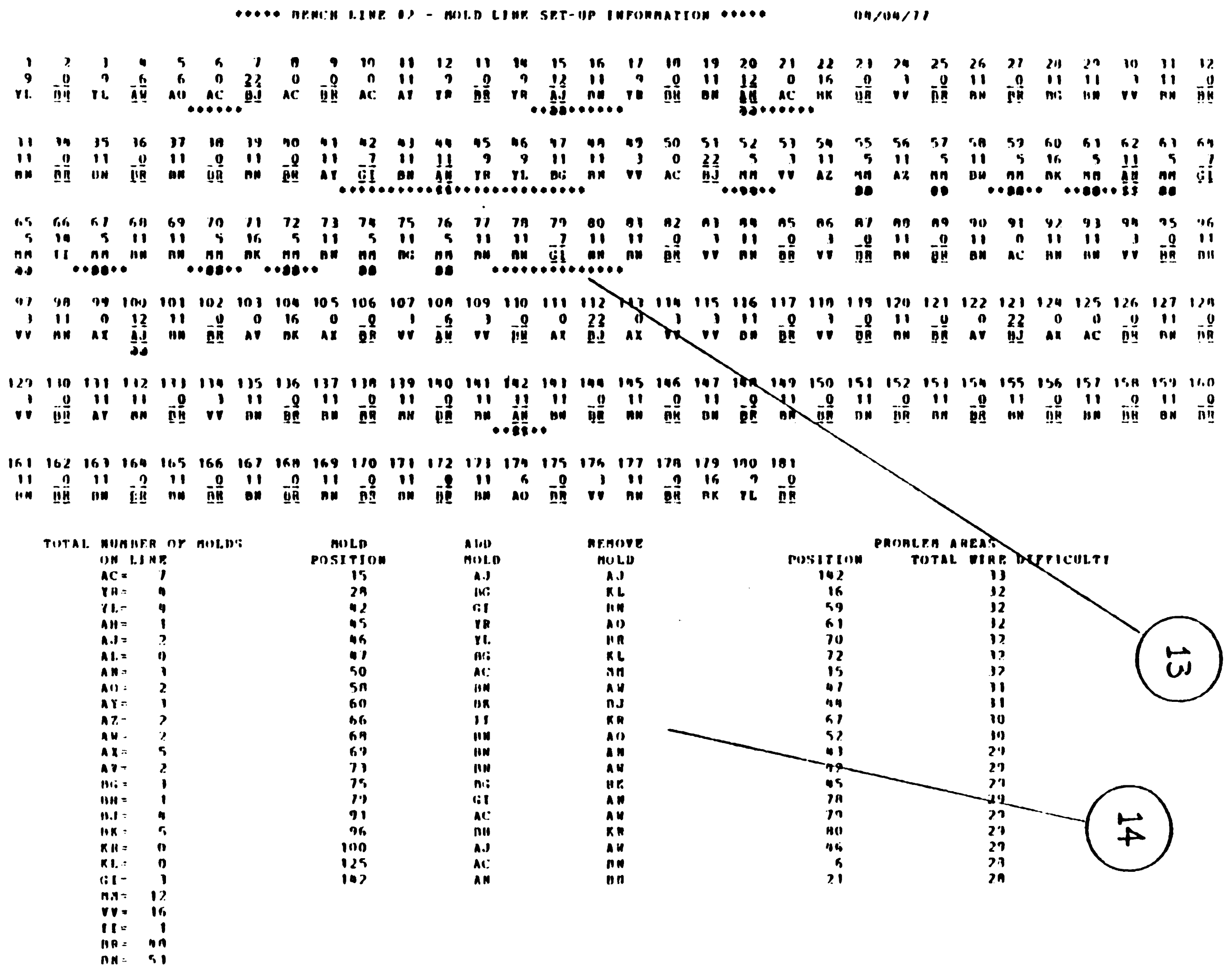
```
4 ICNT=    4  AVAIL  SDIFF
    AJ     12
    BH     11
    BN     11
    BN     11

 4 POS   ADD   SUB   NEED
     60      II     BJ      15
     58      BN     AW      12
     66      BK     KR      19
     68             AO      10
     69             AN      10
     50      AC     MM       4
     73      YL     AW      10
     47      BN     KL      13
     46      YR     BR       9
     45      AC     AO       9
     75      BG     BK      15
     42      GI     BN       9
     79      GI     AN       9
     91      AC     AW       9
     28      BG     KL      17
     96      BG     KR      17
    100      AJ     AW      20
     15      AN     AJ      11
    125             BN       9
    142             BN       9

THE SCHEDULING PROBLEM CANNOT BE SOLVED WITH AN AVERAGE WIRE LOAD DIFFICULTY OF  31
LWD IS NOW BEING SET TO  32
```

Figure 17(d). Sample - Intermediate Results (LWD 31)

```
4 ICNT=    1  AVAIL   SDIFF
    AN     11

 4 POS   ADD   SUB   NEED
     60        BK       HJ       16
     58        BN       AW       11
     66        II       KR       20
     68        BN       AO       13
     69        BN       AN       11
     50        AC       MM        5
     73        BN       AW       11
     47        BG       KL       12
     46        YL       BR       10
     45        YR       AO       10
     75        BG       BK       16
     42        GI       BN       10
     79        GT       AN       10
     91        AC       AW       10
     28        BG       KL       18
     96        BH       KR       18
    100        AJ       AW       21
     15        AJ       AJ       12
    125        AC       BN       10
    142                 BN       10

THE SCHEDULING & SEQUENCING ALGORITHM WAS SOLVED WITH AN AVERAGE WIRE LOAD DIFFICULTY OF  32
```

12

Figure 17(e). Sample - Intermediate Results (LWD 32)



```
***** BENCH LINE #2 - MOLD LINE SET-UP INFORMATION *****          08/04/77

  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32
  9   0   9   6   6   0  22   0   0   0  11   9   0   9  12  11   9   0  11  12   0  16   0   3   0  11   0  11  11   3  11   0
 YL  DR  YL  AW  AO  AC  BJ  AC  BR  AC  AY  YR  BR  YR  AJ  BN  YR  BR  BN  AN  AC  HK  BR  VV  BR  BN  BR  BG  BN  VV  BN  BR
                    ******                              **********                 ********

 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64
 11   0  11   0  11   0  11   0  11   7  11  11   9   9  11  11   3   0  22   5   3  11   5  11   5  11   5  16   5  11   5   7
 BN  BR  DN  BR  BN  BR  BN  BN  AY  GI  BN  AN  YR  YL  BG  BN  VV  AC  BJ  MM  VV  AZ  MM  AZ  MM  DW  MM  BK  MM  AN  MM  GI
                                 **************************              ******      **      **      *****   ******$$  **

 65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96
  5  14   5  11  11   5  16   5  11   5  11   5  11  11   7  11  11   0   3  11   0   3   0  11   0  11   0  11  11   3   0  11
 MM  II  MM  BN  BN  MM  BK  MM  BN  MM  BG  MM  BN  BN  GI  BN  BN  BR  VV  BN  BR  VV  DR  BN  BN  BN  AC  BN  BN  VV  HR  DH
 **    ******      ******  ******      **      **      ***********

 97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
  3  11   0  12  11   0   0  16   0   0   3   6   3   0   0  22   0   3   3  11   0   3   0  11   0   0  22   0   0   0  11   0
 VV  MN  AX  AJ  BN  BR  AV  BK  AX  BR  VV  AW  VV  BH  AX  BJ  AX  VV  VV  BN  BR  VV  DR  BN  BR  AV  HJ  AX  AC  DR  BN  DR
             **

129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
  3   0  11  11   0   3  11   0  11   0  11   0  11  11  11   0  11   0  11   0  11   0  11   0  11   0  11   0  11   0  11   0
 VV  DR  AY  BN  DR  VV  BN  BR  BN  BR  BN  DR  BN  AN  BN  BR  BN  BK  BN  BR  BN  DR  DN  DR  BN  BR  BN  DR  BN  BH  BN  DR
                                                   *****

161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181
 11   0  11   0  11   0  11   0  11   0  11   0  11   6   0   3  11   0  16   9   0
 BN  BR  DN  ER  BN  DR  BN  DR  BN  BR  DN  DR  BN  AO  DR  VV  BN  BR  BK  YL  DR

TOTAL NUMBER OF MOLDS        MOLD        ADD        REMOVE             PROBLEM AREAS
       ON LINE             POSITION      MOLD        MOLD         POSITION     TOTAL WIRE DIFFICULTY
       AC=   7                15          AJ          AJ             142            33
       YR=   4                28          BG          KL              16            32
       YL=   4                42          GI          BN              59            32
       AH=   1                45          YR          AO              61            32
       AJ=   2                46          YL          BR              70            32
       AL=   0                47          BG          KL              72            32
       AN=   3                50          AC          MM              15            32
       AO=   2                58          BN          AW              47            31
       AY=   1                60          BR          BJ              44            31
       AZ=   2                66          II          KR              67            30
       AW=   2                68          BN          AO              52            30
       AX=   5                69          BN          AN              43            29
       AV=   2                73          BN          AW              49            29
       BG=   1                75          BG          HK              45            29
       BH=   1                79          GI          AN              78            29
       BJ=   4                91          AC          AW              79            29
       BK=   5                96          DH          KR              80            29
       KR=   0               100          AJ          AW              46            29
       KL=   0               125          AC          BN               6            28
       GI=   3               142          AN          BB              21            28
       MM=  12
       VV=  16
       II=   1
       BR=  48
       BN=  51
```

Figure 18. Sample - Final Mold Line-up Summary

VI. CONCLUSIONS

This final chapter examines the performance of the foam assembly line scheduling model. An optimization model has the advantage of determining the optimal set of molds which should be placed on the assembly line for any particular shipping and inventory requirements. It has the capability of minimizing some cost objective function, and yet provide the best possible combination of molds. The assembly line scheduling model has satisfied all of the plant's constraints as follows:

1. Placed the proper molds on the assembly line to meet the weekly shipping demand.
2. Assigned upenders to each particular mold type.
3. Minimized set-up and inventory costs.
4. Obeyed all of the mold sequencing rules.
5. Kept the average wire load difficulty to a minimum.
6. Obeyed the warehousing constraints.
7. Considered the human aspects of assembly line work.
8. Kept the distance between mold changes to a minimum.

The model generates a new mold configuration each week. It looks into future weeks demand, adds and removes molds which tend to minimize future problems and bottlenecks. Finally, the assembly line mold scheduling model generates a simple summary report to help Production Control solve the plant's foam scheduling problems.

There are several assumptions and problems, however, that could not be put into the assembly line model in its first version. The model assumes that the production requirements and number of shifts are known. The assembly line is currently scheduled for 3 shifts, but economically this is not always a feasible solution. Due to the economy, car sales, transportation problems, and even weather conditions, the plant may be forced to limit the number of shifts. Future versions of this program must account for a changing number of shifts and whether or not Saturday overtime should be scheduled to eliminate backorder production.

Ideally, if the production forecasting information could be relied on, the model should schedule the assembly line for a two week period. This would give the plant more information to help solve future problems or bottlenecks which could occur the following week. The production shipping requirements would have to become better estimates of the actual production. The linear programming algorithm of the model, however, would become much more complicated and would require more computer execution time.

Finally, something must be said about the performance of the model as it compares to the actual production of the plant. The assembly line mold scheduling model, in trial runs, has compared very favorably with a solution which is as good or better than the plant's actual mold assignments. Production activities very closely model the changes which are actually taking place on the assembly line. One of the reasons for the model's success is that a great deal of time and effort was spent defining the problem and analyzing the steps needed to determine an optimal solution. Differences in the mold scheduling line-up were due to past scheduling weeks. The assembly line, however, closely approximates the manual scheduling activities which have occurred in the past. In conclusion, the model has recently been implemented in the plant environment, and is responsible for the plant's assembly line mold scheduling functions.

LIST OF REFERENCES

1. Hillier, Frederick S. & Lieberman, Gerald J., _Operations Research_, Holden-Day, Inc., San Francisco, 1967.

2. _IBM Mathematical Programming System Extended/370 (MPSX/370) Basic Reference Manual_, IBM Corp., SH19-1127-0, April 1976.

3. _IBM Mathematical Programming System Extended/370 (MPSX/370) Control Languages_, IBM Corp., SH19-1094-1, April 1975.

4. _IBM Mathematical Programming System Extended/370 (MPSX/370) Messages_, IBM Corp., SH19-1096-0, October 1976.

5. _IBM Mathematical Programming System Extended/370 (MPSX/370) Mixed Integer Programming/370 (MIP/370) Program Reference Manual_, IBM Corp., SH19-1094-1, April 1975.

6. _IBM Mathematical Programming System Extended/370 (MPSX/370) Operations Guide (OS/VS)_, IBM Corp., SH19-1097-1, October 1976.

7. _IBM Mathematical Programming System Extended/370 (MPSX/370) Program Reference Manual_, IBM Corp., SH19-1099-1, October 1976.

8. _OS PL/I Checkout & Optimizing Compilers: Language Reference Manual_, IBM Corp., GC33-0009-3, July 1974.

9. _OS PL/I Optimizing Compiler: Programmer's Guide_, IBM Corp., SC33-0006-3, March 1976.

10. _OS PL/I Optimizing Compiler: Messages_, IBM Corp., SC33-0027-3, July 1975.

11. Wirth, Niklaus, _Algorithms + Data Structures = Programs_, Prentice-Hall, Inc., New Jersey, 1976.

APPENDIX A

SOURCE PROGRAMS

```
FFFFFFFFFFFF  OOOOOOOOOOOO   AAAAAAAAAA   MM         MM
FFFFFFFFFFFF  OOOOOOOOOOOO  AAAAAAAAAAAA  MMM       MMM
FF            OO        OO  AA        AA  MMMM     MMMM
FF            OO        OO  AA        AA  MM MM  MM  MM
FF            OO        OO  AA        AA  MM  MMMM   MM
FFFFFFFF      OO        OO  AAAAAAAAAAAA  MM   MM    MM
FFFFFFFF      OO        OO  AAAAAAAAAAAA  MM         MM
FF            OO        OO  AA        AA  MM         MM
FF            OO        OO  AA        AA  MM         MM
FF            OO        OO  AA        AA  MM         MM
FF            OOOOOOOOOOOO  AA        AA  MM         MM
FF            OOOOOOOOOOOO  AA        AA  MM         MM
```

```
//FOAM   JOB (60011,0,1),MARKLE,MSGLEVEL=1,CLASS=L,PRTY=8
//S3     EXEC PGM=LINK
//STEPLIB DD DSN=FOAM.LOADLIB,DISP=SHR
//          DD DSN=OPL.MPSX370,DISP=SHR,VOL=SER=SYSPAG,UNIT=3330
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//CARD   DD UNIT=3330,SPACE=(CYL,(2,2)),
//          DISP=(OLD,KEEP),VOL=SER=EEEEEE,DSN=P60011.MARKLE.CARD,
//          DCB=(LRECL=80,BLKSIZE=12960,RECFM=FB)
//OUT    DD UNIT=3330,SPACE=(CYL,(5,3)),DSN=P169999.MARKLE.OUT,
//           DCB=(RECFM=VB,LRECL=204,BLKSIZE=1024),DISP=(OLD,KEEP),
//           VOL=SER=EEEEEE
//ETA1         DD UNIT=3330,SPACE=(CYL,(3,1),,CONTIG)
//ETA2         DD UNIT=3330,SPACE=(CYL,(3,1),,CONTIG)
//MATRIX1 DD  UNIT=3330,SPACE=(CYL,(10),,CONTIG)
//MATRIX2 DD  UNIT=3330,SPACE=(CYL,(10),,CONTIG)
//MIXWORK DD UNIT=3330,SPACE=(CYL,(3,3))
//PROBFILE DD UNIT=3330,SPACE=(CYL,(5,3))
//SCRATCH1 DD UNIT=3330,SPACE=(CYL,(5),,CONTIG)
//SCRATCH2 DD UNIT=3330,SPACE=(CYL,(5),,CONTIG)
//G.STATIC DD *
                ***** BENCH LINE #2 - MOLD LINE SET-UP INFORMATION *****
 25   3     181     1500     27
AC    1657569 S   0 1 N 41 27 N   55 415   1.85   2145
YR    1660106 S   9 2 N 19 14 N   78 820   1.89   3042
YL    1660107 S   9 2 N 19 13 N   78 820   1.89   3042
AH    1677286 L  12 1 N  2  2 Y   20 385   5.02    720
AJ    1677290 L  12 1 N  4  2 Y   20 340   4.99    720
AL    1681929 S   7 1 N  2  1 Y   28 325   4.33    336
AN    1690551 L  11 1 Y  9  4 N   22 375   3.94    858
AO    1690553 S   6 1 N  9  5 N   22 405   4.15    858
AY    1690562 S  11 2 N  5  5 N   64 790   1.80   1280
AZ    1690563 S  11 2 N  4  4 N   64 790   1.80   1280
AW    1694482 L   6 2 N 21 18 N   72 830   2.17   2280
AX    1694483 S   0 2 N 21 19 N   66 830   1.51   2706
AV    1694709 S   0 1 N 21 20 N   28 415   3.38   1176
BG    1733947 S  11 2 N  1  1 N   64 650   1.80   2304
BH    1733948 S  11 2 N  2  0 N   64 710   1.80   1152
BJ    1737211 L  22 2 N  6  7 N   48 820   2.52   1920
BK    1737223 S  16 2 N  6  6 N   60 810   1.89   2340
KR    3062748 S   9 2 N  6  2 Y   56 780   1.93   2128
KL    3062749 S   9 2 N  6  3 Y   56 790   1.93   2128
GI    3063026 L   7 2 N  4  4 N   62 800   1.74   2356
MM    9679612 S   5 1 Y 23 15 Y   28 415   3.55   1288
VV    9679613 S   3 2 N 22 18 N   84 830   1.28   4158
II    9730082 S  14 1 N  5  2 N   24 380   4.24    864
BR   20001557 L   0 1 N 50 49 N   22 410   3.96   3144
BN   20001606 S  11 1 N 63 47 N   24 410   4.09   3456
//G.MOLD   DD *
 YR  YR  YR  YL  YL  AC  BJ  AC  AC  YL  YL  AC  BN  AC  AJ  AC  BN  BR  BN  BR
 BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR
 BN  BR  MM  AN  MM  VV  MM  AN  VV  MM  VV  MM  AX  BJ  MM  VV  MM  VV  MM  BJ
 MM  VV  MM  VV  MM  AH  VV  MM  VV  MM  BK  MM  YL  MM  BK  AC  AC  BR  BK  BR
 AX  BN  BR  BN  AX  BN  AX  BN  AX  BN  AX  BN  BR  KL  AX  AW  AX  AY  AX  AW
 AC  KR  AC  AW  VV  AZ  VV  AW  VV  AW  AV  BJ  AV  AX  AV  BJ  AV  AV  YL  AW
 BN  AX  BR  BN  BR  AC  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR
 BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR  BN  BR
 BN  BR  BN  BR  BN  AC  BN  AC  AX  AX  VV  BG  BR  AO  BR  BK  BR  AC  BK  YR
 YR
```

```
//G.DYNAM DD *
AC       2778       2970       2750       2750       4290
YR       1518       3744       3744       3432       5460
YL       1784       3276       3588       3900       5616
AH        138        320        360        400        680
AJ        337        280        240        280        480
AL        110          0        112         56          0
AN        639       1188       1188        968       1936
AO       1419        968       1364        880       2156
AY       1884        640       1024        512       1536
AZ        696        896       1280        640       2176
AW        745       5760       5184       5184       8496
AX       8139       5280       5940       5412       8580
AV       -765       2576       2520       2744       4816
BG        255        256        384        256        768
BH        317        128        128          0        128
BJ        667       2496       3072       3168       4992
BK       2974       2640       2880       3120       5040
KR        713       1120       1344        784       2240
KL        244       1232       1232        896       2352
GI       2777          0          0          0          0
MM       2056       4536       4480       4536       8344
VV        146       8910       9108       8910      17820
II        390        240        240        192        480
BR      15688      13376      15224      15886      27224
BN      17456      15456      17568      18432      32621
**
//S4  EXEC PGM=IEBGENER,COND=(1000,NE,S3)
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=P169999.MARKLE.OUT,UNIT=3330,VOL=SER=EEEEEE,DISP=OLD,
//   DCB=(RECFM=VBA,LRECL=204,BLKSIZE=1024)
//SYSUT2 DD SYSOUT=A
//SYSIN DD *
/*
```

```
MM        MM   AAAAAAAAAA     IIIIIIIIII   NN        NN
MMM      MMM  AAAAAAAAAAAA    IIIIIIIIII   NNN       NN
MMMM    MMMM  AA        AA        II       NNNN      NN
MM MM  MM MM  AA        AA        II       NN NN     NN
MM  MMMM  MM  AA        AA        II       NN  NN    NN
MM   MM   MM  AAAAAAAAAAAA        II       NN   NN   NN
MM        MM  AAAAAAAAAAAA        II       NN    NN  NN
MM        MM  AA        AA        II       NN     NN NN
MM        MM  AA        AA        II       NN      NNNN
MM        MM  AA        AA        II       NN       NNN
MM        MM  AA        AA    IIIIIIIIII   NN        NN
MM        MM  AA        AA    IIIIIIIIII   NN         N
```

SOURCE LISTING

```
STMT

   1 MAIN:   PROC OPTIONS(MAIN) REORDER;

      /**********************************************************************/
      /**********  MOLD LINE-UP TITLE CARD  ********************************/
      /**********************************************************************/

      /***  TITLE         MOLD LINE-UP TITLE CARD                          ***/

      /**********************************************************************/
      /**********  PARAMETER CARD INFORMATION  *****************************/
      /**********************************************************************/

      /***  NUM           TOTAL NUMBER OF DIFFERENT MOLD TYPES             ***/
      /***  SHIFT         NUMBER OF SHIFTS WORKED                          ***/
      /***  IPOS          NUMBER OF MOLD POSITIONS                         ***/
      /***  WARE          TOTAL NUMBER OF BASKET POSITIONS IN WAREHOUSE    ***/

      /**********************************************************************/
      /**********  STATIC INFORMATION OF THE MOLDS IN THE SYSTEM  *********/
      /**********************************************************************/

      /***  CODE(50)      MOLD LETTER CODES                                ***/
      /***  PART(50)      PART NUMBERS                                     ***/
      /***  SIZE(50)      MOLD SIZE (L-LARGE/S-SMALL)                      ***/
      /***  LOAD(50)      WIRE LOAD DIFFICULTY                             ***/
      /***  NUMPRT(50)    NUMBER OF PARTS IN CAVITY (1/2)                  ***/
      /***  PLAT(50)      PLATFORM WORK (Y-YES/N-NO)                       ***/
      /***  MOLDS(50)     TOTAL NUMBER OF MOLDS AVAILABLE                  ***/
      /***  FRAME(50)     CURRENT NUMBER OF MOLDS IN FRAMES                ***/
      /***  TAPE(50)      TAPE AFTER POST CURE (Y-YES/N-NO)                ***/
      /***  PACK(50)      NUMBER OF PARTS IN STANDARD PACK (BASKET)        ***/
      /***  RATE(50)      PRODUCTION RATE PER SHIFT                        ***/
      /***  VALUE(50)     VALUE OF THE PART                                ***/
      /***  SAFETY(50)    SAFETY STOCK REQUIRED IN INVENTORY               ***/

      /**********************************************************************/
      /**********  DYNAMIC INFORMATION OF THE MOLDS IN THE SYSTEM  *********/
      /**********************************************************************/

      /***  INVBAL(50)    INITIAL INVENTORY FOR EACH PART                  ***/
      /***  SHIP1(50)     PARTS SHIPPED IN 1ST WEEK                        ***/
      /***  SHIP3(50)     PARTS SHIPPED IN 3RD WEEK                        ***/
      /***  SHIP4(50)     PARTS SHIPPED IN 4TH WEEK                        ***/
      /***  FCD(50)       INITIAL INVENTORY - 1ST WEEK SHIPPED             ***/
```

```
PL/I OPTIMIZING COMPILER         MAIN:   PROC OPTIONS(MAIN) REORDER;

    STMT

         /*******************************************************************/
         /**********  MISC. VARIABLES FROM THE LINE-UP PROGRAM  ***************/
         /*******************************************************************/

         /***  CLINE(200)    CARRIER LETTER CODE ON SCHEDULED LINE-UP         ***/
         /***  LPOS(50)      MOLD POSITIONS WHERE PROBLEM AREAS EXIST         ***/
         /***  LDIFF(50)     TOTAL WIRE LOAD DIFFICULTY FOR PROBLEM AREA      ***/
         /***  MTOT(50)      TOTAL NUMBER OF MOLDS ON LINE FOR EACH PART      ***/

         /*******************************************************************/

      2        DCL INPUT ENTRY EXTERNAL;
      3        DCL LINEUP ENTRY EXTERNAL;
      4        DCL GENER ENTRY EXTERNAL;
      5        DCL MIP ENTRY EXTERNAL;
      6        DCL SCHED ENTRY EXTERNAL;

      7        DCL (NUM,SHIFT,IPOS,LNUM,ISIM) FIXED BIN(15) STATIC EXTERNAL;
      8        DCL WARE FIXED BIN(31) STATIC EXTERNAL;
      9        DCL (SIZE(50),PLAT(50),TAPE(50)) CHAR(1) STATIC EXTERNAL;
     10        DCL (CODE(50),CLINE(200)) CHAR(2) STATIC EXTERNAL;
     11        DCL VALUE(50) FIXED DEC(7,2) STATIC EXTERNAL;
     12        DCL PART(50) CHAR(9) STATIC EXTERNAL;
     13        DCL (LOAD(50),NUMPRT(50),MOLDS(50),FRAME(50),PACK(50),RATE(50),
                    SHIP1(50),SHIP3(50),SHIP4(50),INVBAL(50),SAFETY(50),
                    FCD(50),SHIP2(50),LPOS(50),LDIFF(50),MTOT(50),LWDIFF(200))
                    FIXED BIN(15) STATIC EXTERNAL;
     14        DCL TITLE CHAR(80) STATIC EXTERNAL;
     15        DCL 1 CHANGE(100) STATIC EXTERNAL,
                    2 (POS,NEED) FIXED BIN(15),
                    2 (ADD,SUB) CHAR(2);
     16        DCL ICOUNT FIXED BIN(15) STATIC EXTERNAL;
```

```
STMT

          /************************************************/
          /***  START THE MOLD SCHEDULING SIMULATION   ***/
          /************************************************/

  17           DISPLAY('+   ENTERING STATIC INPUT PROGRAM');
  18           CALL INPUT;

  19           DISPLAY('+   ENTERING PROGRAM LINE-UP');
  20           CALL LINEUP;

  21           DISPLAY('+   ENTERING THE DATA GENERATION PROGRAM');
  22           CALL GENER;

  23           DISPLAY('+   ENTERING THE MIXED INTEGER PROGRAMMING PROBLEM');
  24           CALL MIP;

  25           DISPLAY('+   ENTERING THE SCHEDULING PROGRAM');
  26           CALL SCHED;

  27           DISPLAY('+   ENTERING FINAL PROGRAM LINE-UP');
  28           CALL LINEUP;

  29           DISPLAY('+   END OF THE PROGRAM');

  30      FINISH:
               END MAIN;
```

```
IIIIIIIIII   NN         NN  PPPPPPPPPPP    UU        UU   TTTTTTTTTTTT
IIIIIIIIII   NNN        NN  PPPPPPPPPPPP   UU        UU   TTTTTTTTTTTT
    II       NNNN       NN  PP        PP   UU        UU        TT
    II       NN NN      NN  PP        PP   UU        UU        TT
    II       NN   NN    NN  PP        PP   UU        UU        TT
    II       NN    NN   NN  PPPPPPPPPPPP   UU        UU        TT
    II       NN     NN  NN  PPPPPPPPPPP    UU        UU        TT
    II       NN      NN NN  PP             UU        UU        TT
    II       NN       NNNN  PP             UU        UU        TT
    II       NN        NNN  PP             UU        UU        TT
IIIIIIIIII   NN         NN  PP             UUUUUUUUUUUU        TT
IIIIIIIIII   NN          N  PP              UUUUUUUUUU         TT
```

```
PL/I OPTIMIZING COMPILER          INPUT:  PROC REORDER;

                    SOURCE LISTING

    STMT


       1  INPUT:  PROC REORDER;

       2       DCL (NUM,SHIFT,IPOS,LNUM,ISIM) FIXED BIN(15) STATIC EXTERNAL;
       3       DCL WARE FIXED BIN(31) STATIC EXTERNAL;
       4       DCL (SIZE(50),PLAT(50),TAPE(50)) CHAR(1) STATIC EXTERNAL;
       5       DCL (CODE(50),CLINE(200)) CHAR(2) STATIC EXTERNAL;
       6       DCL VALUE(50) FIXED DEC(7,2) STATIC EXTERNAL;
       7       DCL PART(50) CHAR(9) STATIC EXTERNAL;
       8       DCL (LOAD(50),NUMPRT(50),MOLDS(50),FRAME(50),PACK(50),RATE(50),
                    SHIP1(50),SHIP3(50),SHIP4(50),INVBAL(50),SAFETY(50),
                    FCD(50),SHIP2(50),LPOS(50),LDIFF(50),MTOT(50),LWDIFF(200))
                    FIXED BIN(15) STATIC EXTERNAL;
       9       DCL TITLE CHAR(80) STATIC EXTERNAL;
      10       DCL 1 CHANGE(100) STATIC EXTERNAL,
                    2 (POS,NEED) FIXED BIN(15),
                    2 (ADD,SUB) CHAR(2);
      11       DCL ICOUNT FIXED BIN(15) STATIC EXTERNAL;
      12       DCL LWD FIXED BIN(15) STATIC EXTERNAL;

          /***************************************************/
          /*****  INITIALIZE STATIC EXTERNAL VARIABLES  *****/
          /***************************************************/

      13       SIZE(*)=' ';          PLAT(*)=' ';
      15       TAPE(*)=' ';          CODE(*)=' ';
      17       CLINE(*)=' ';         PART(*)=' ';
      19       VALUE(*)=0.0;         TITLE=' ';
      21       LOAD(*)=0;            RATE(*)=0;
      23       NUMPRT(*)=0;          MOLDS(*)=0;
      25       FRAME(*)=0;           PACK(*)=0;
      27       SHIP1(*)=0;           SHIP2(*)=0;
      29       SHIP3(*)=0;           SHIP4(*)=0;
      31       INVBAL(*)=0;          SAFETY(*)=0;
      33       LDIFF(*)=0;           LPOS(*)=0;
      35       MTOT(*)=0;            LWDIFF(*)=0;
      37       FCD(*)=0;
```

```
   STMT

          /******************************************/
          /***  READ MOLD LINE-UP TITLE CARD  ***/
          /******************************************/

     38       ISIM=0;
     39       GET FILE(STATIC) EDIT(TITLE) (A(80));
     40       PUT SKIP EDIT(TITLE) (X(1),A(80));
     41       PUT SKIP(2);

          /*******************************************/
          /***  READ PARAMETER CARD INFORMATION  ***/
          /*******************************************/

     42       GET SKIP FILE(STATIC) EDIT(NUM,SHIFT,IPOS,WARE,LWD)
                 (2(F(3)),2(F(7)),F(5));
     43       PUT SKIP EDIT(NUM,SHIFT,IPOS,WARE,LWD)
                 (2(F(3)),2(F(7)),F(5));

          /**********************************************************/
          /***  READ STATIC INFORMATION OF THE MOLDS IN THE SYSTEM  ***/
          /**********************************************************/

     44       DO I=1 TO NUM;
     45       GET FILE(STATIC) EDIT(CODE(I),PART(I),SIZE(I),LOAD(I),NUMPRT(I),
                 PLAT(I),MOLDS(I),FRAME(I),TAPE(I),PACK(I),RATE(I),VALUE(I),
                 SAFETY(I))
                 (COL(1),A(2),A(9),X(1),A(1),F(3),F(2),X(1),A(1),2(F(3)),X(1),
                  A(1),2(F(4)),F(6,2),F(6));
     46       PUT EDIT(CODE(I),PART(I),SIZE(I),LOAD(I),NUMPRT(I),
                 PLAT(I),MOLDS(I),FRAME(I),TAPE(I),PACK(I),RATE(I),VALUE(I),
                 SAFETY(I))
                 (COL(2),A(2),A(9),X(1),A(1),F(3),F(2),X(1),A(1),2(F(3)),X(1),
                  A(1),2(F(4)),F(6,2),F(6));
     47       END;
```

```
STMT

          /*****************************************/
          /***  READ MOLD LINE-UP INFORMATION  ***/
          /*****************************************/

   48          IST=1;
   49          IEND=20;
   50          IFLAG=0;
   51          PUT SKIP(2);

   52     LINE_UP_INFO:
               GET SKIP FILE(MOLD) EDIT((CLINE(J) DO J=IST TO IEND))
                   (20(X(1),A(2)));
   53          PUT SKIP EDIT((CLINE(J) DO J=IST TO IEND)) (20(X(1),A(2)));
   54          IF IFLAG=1 THEN GO TO FINISH;
   55          IST=IST+20;
   56          IEND=IEND+20;
   57          IF IEND>=IPOS THEN IFLAG=1;
   58          IF IEND > IPOS THEN IEND=IPOS;
   59          GO TO LINE_UP_INFO;

   60     FINISH:
               END INPUT;
```

```
LL             IIIIIIIIII   NN        NN  EEEEEEEEEEEE  UU        UU  PPPPPPPPPPP
LL             IIIIIIIIII   NNN       NN  EEEEEEEEEEEE  UU        UU  PPPPPPPPPPPP
LL                 II       NNNN      NN  EE            UU        UU  PP        PP
LL                 II       NN NN     NN  EE            UU        UU  PP        PP
LL                 II       NN  NN    NN  EE            UU        UU  PP        PP
LL                 II       NN   NN   NN  EEEEEEEE      UU        UU  PPPPPPPPPPPP
LL                 II       NN    NN  NN  EEEEEEEE      UU        UU  PPPPPPPPPPP
LL                 II       NN     NN NN  EE            UU        UU  PP
LL                 II       NN      NNNN  EE            UU        UU  PP
LL                 II       NN       NNN  EE            UU        UU  PP
LLLLLLLLLLLL   IIIIIIIIII   NN        NN  EEEEEEEEEEEE  UUUUUUUUUUUU  PP
LLLLLLLLLLLL   IIIIIIIIII   NN         N  EEEEEEEEEEEE   UUUUUUUUUU   PP
```

```
PL/I OPTIMIZING COMPILER         LINEUP: PROC REORDER;

                    SOURCE LISTING

    STMT


       1  LINEUP: PROC REORDER;

       2       DCL (NUM,SHIFT,IPOS,LNUM,ISIM) FIXED BIN(15) STATIC EXTERNAL;
       3       DCL WARE FIXED BIN(31) STATIC EXTERNAL;
       4       DCL (SIZE(50),PLAT(50),TAPE(50)) CHAR(1) STATIC EXTERNAL;
       5       DCL (CODE(50),CLINE(200)) CHAR(2) STATIC EXTERNAL;
       6       DCL VALUE(50) FIXED DEC(7,2) STATIC EXTERNAL;
       7       DCL PART(50) CHAR(9) STATIC EXTERNAL;
       8       DCL (LOAD(50),NUMPRT(50),MOLDS(50),FRAME(50),PACK(50),RATE(50),
                    SHIP1(50),SHIP3(50),SHIP4(50),INVBAL(50),SAFETY(50),
                    FCD(50),SHIP2(50),LPOS(50),LDIFF(50),MTOT(50),LWDIFF(200))
                    FIXED BIN(15) STATIC EXTERNAL;
       9       DCL TITLE CHAR(80) STATIC EXTERNAL;

      10       DCL 1 CHANGE(100) STATIC EXTERNAL,
                    2 (POS,NEED) FIXED BIN(15),
                    2 (ADD,SUB) CHAR(2);
      11       DCL ICOUNT FIXED BIN(15) STATIC EXTERNAL;
      12       DCL LWD FIXED BIN(15) STATIC EXTERNAL;

      13       DCL VIOLAT(200) CHAR(4);
      14       DCL (TMOLD(200),PMOLD(200),SMOLD(200)) CHAR(2);
      15       DCL ITOT(200) FIXED BIN(15);
      16       DCL UNDER CHAR(2) INIT('__');
      17       DCL STAR CHAR(4) INIT('****');
      18       DCL STAR1 CHAR(2) INIT('**');
      19       DCL EQ CHAR(2) INIT('= ');
      20       DCL XL CHAR(1) INIT('L');
      21       DCL TNM CHAR(21) INIT('TOTAL NUMBER OF MOLDS');

      22       DCL DATE1 CHAR(6);
      23       DCL DATE BUILTIN;
      24       DCL 1 TDATE,
                  2 MONTH CHAR(2),
                  2 FILL1 CHAR(1) INIT('/'),
                  2 DAY CHAR(2),
                  2 FILL2 CHAR(1) INIT('/'),
                  2 YEAR CHAR(2);
      25       DATE1=DATE;
      26       MONTH=SUBSTR(DATE1,3,2);
      27       DAY=SUBSTR(DATE1,5,2);
      28       YEAR=SUBSTR(DATE1,1,2);
      29       DCL TDATE1 CHAR(8) BASED(P1);
      30          P1=ADDR(TDATE);
```

```
PL/I OPTIMIZING COMPILER         LINEUP: PROC REORDER;

    STMT

         /*******************************************/
         /***  INITIALIZE THE PROGRAM VARIABLES   ***/
         /*******************************************/

     31       ITOT(*)=0;
     32       LWDIFF(*)=0;
     33       LPOS(*)=0;
     34       SMOLD(*)=' ';
     35       VIOLAT(*)=' ';
     36       TMOLD(*)=' ';
     37       PMOLD(*)=' ';
     38       MTOT(*)=0;

         /****************************************************/
         /***  DETERMINE WIRE LOAD DIFFICULTY FOR EACH MOLD  ***/
         /****************************************************/

     39       DO I=1 TO IPOS;

     40       DO J=1 TO NUM;
     41       IF CLINE(I)=CODE(J) THEN GO TO FOUND;
     42       END;

     43       PUT SKIP EDIT('***ERROR - MOLD CODE ',CLINE(I),
                 ' DOES NOT EXIST IN LINE-UP***') (X(3),A,A(2),A);
     44       GO TO FINISH;

     45  FOUND:
              IF TAPE(J)='Y' THEN TMOLD(I)='@@';
     46       IF PLAT(J)='Y'  THEN PMOLD(I)='$$';
     47       LWDIFF(I)=LOAD(J);
     48       IF SIZE(J)=XL THEN SMOLD(I)=UNDER;
     49       MTOT(J)=MTOT(J)+1;
     50       END;
    STMT

          /**********************************************************/
          /***  DETERMINE VIOLATIONS IN THE SCHEDULED LINE-UP  ***/
          /**********************************************************/

     51          LNUM=1;
     52          DO J=1 TO IPOS;
     53          I=J-1;
     54          IF I=0 THEN I=IPOS-1;
     55          K=J+1;
     56          IF K=IPOS+1 THEN K=1;
     57          ITOT(J)=LWDIFF(I)+LWDIFF(J)+LWDIFF(K);

     58          IF ITOT(J)<= LWD THEN GO TO APPROVE;
     59          VIOLAT(J)=STAR;
     60          VIOLAT(J+1)=STAR1;
     61          LPOS(LNUM)=J;
     62          LDIFF(LNUM)=ITOT(J);
     63          LNUM=LNUM+1;

     64   APPROVE:
                 END;
     65          LNUM=LNUM-1;
    STMT

         /**********************************************/
         /***  PRODUCE A LINE-UP SCHEDULE PICTURE  ***/
         /**********************************************/

      66        IST=1;
      67        IEND=32;
      68        IFLAG=0;
      69        PUT PAGE EDIT(TITLE,TDATE1) (X(10),A,A(8));
      70        PUT SKIP(2);

      71  PICTURE:
                PUT SKIP EDIT((J DO J=IST TO IEND)) (32(X(1),F(3)));
      72        PUT SKIP EDIT((LWDIFF(J) DO J=IST TO IEND)) (32(X(1),F(3)));
      73        PUT SKIP(0)  EDIT((SMOLD(J) DO J=IST TO IEND)) (32(X(2),A(2)));
      74        PUT SKIP EDIT((CLINE(J) DO J=IST TO IEND)) (32(X(2),A(2)));
      75        PUT SKIP(0)  EDIT((SHOLD(J) DO J=IST TO IEND)) (32(X(2),A(2)));
      76        PUT SKIP EDIT((VIOLAT(J) DO J=IST TO IEND)) (32(A(4)));
      77        PUT SKIP(0)  EDIT((THOLD(J) DO J=IST TO IEND)) (32(X(2),A(2)));
      78        PUT SKIP(0)  EDIT((PHOLD(J) DO J=IST TO IEND)) (32(X(2),A(2)));
      79        PUT SKIP;
      80        IF IFLAG=1 THEN GO TO WIRE_LOAD;
      81        IST=IST+32;
      82        IEND=IEND+32;
      83        IF IEND>IPOS THEN IFLAG=1;
      84        IF IEND>IPOS THEN IEND=IPOS;
      85        GO TO PICTURE;

         /**********************************************/
         /***  SORT THE WIRE LOAD DIFFICULTY QUEUE  ***/
         /**********************************************/

      86  WIRE_LOAD:
                DO I=1 TO LNUM-1;
      87        DO J=I+1 TO LNUM;
      88        IF LDIFF(I) >= LDIFF(J) THEN GO TO SORT;

      89        ITEMP1=LDIFF(I);
      90        ITEMP2=LPOS(I);
      91        LDIFF(I)=LDIFF(J);
      92        LPOS(I)=LPOS(J);
      93        LDIFF(J)=ITEMP1;
      94        LPOS(J)=ITEMP2;

      95  SORT:
                END;
      96       END;
```

```
 GGGGGGGGGG    EEEEEEEEEEEE  NN        NN  EEEEEEEEEEEE  RRRRRRRRRRR
GGGGGGGGGGGG  EEEEEEEEEEEE  NNN       NN  EEEEEEEEEEEE  RRRRRRRRRRRR
GG        GG  EE            NNNN      NN  EE            RR        RR
GG            EE            NN NN     NN  EE            RR        RR
GG            EE            NN  NN    NN  EE            RR        RR
GG            EEEEEEEE      NN   NN   NN  EEEEEEEE      RRRRRRRRRRRR
GG     GGGGG  EEEEEEEE      NN    NN  NN  EEEEEEEE      RRRRRRRRRRR
GG     GGGGG  EE            NN     NN NN  EE            RR    RR
GG        GG  EE            NN      NNNN  EE            RR     RR
GG        GG  EE            NN       NNN  EE            RR      RR
GGGGGGGGGGGG  EEEEEEEEEEEE  NN        NN  EEEEEEEEEEEE  RR       RR
 GGGGGGGGGG   EEEEEEEEEEEE  NN         N  EEEEEEEEEEEE  RR        RR
```

```
PL/I OPTIMIZING COMPILER        GENER:  PROC REORDER;

                    SOURCE LISTING

    STMT

       1  GENER:  PROC REORDER;

       2       DCL (NUM,SHIFT,IPOS,LNUM,ISIM) FIXED BIN(15) STATIC EXTERNAL;
       3       DCL WARE FIXED BIN(31) STATIC EXTERNAL;
       4       DCL (SIZE(50),PLAT(50),TAPE(50)) CHAR(1) STATIC EXTERNAL;
       5       DCL (CODE(50),CLINE(200)) CHAR(2) STATIC EXTERNAL;
       6       DCL VALUE(50) FIXED DEC(7,2) STATIC EXTERNAL;
       7       DCL PART(50) CHAR(9) STATIC EXTERNAL;
       8       DCL (LOAD(50),NUMPRT(50),MOLDS(50),FRAME(50),PACK(50),RATE(50),
                    SHIP1(50),SHIP3(50),SHIP4(50),INVBAL(50),SAFETY(50),
                    FCD(50),SHIP2(50),LPOS(50),LDIFF(50),MTOT(50),LWDIFF(200))
                    FIXED BIN(15) STATIC EXTERNAL;
       9       DCL TITLE CHAR(80) STATIC EXTERNAL;
      10       DCL 1 CHANGE(100) STATIC EXTERNAL,
                    2 (POS,NEED) FIXED BIN(15),
                    2 (ADD,SUB) CHAR(2);
      11       DCL ICOUNT FIXED BIN(15) STATIC EXTERNAL;

      12       DCL BUF CHAR(80);
      13       DCL (INVEN(50),OBJ(50)) FIXED DEC(7,2);
      14       DCL (WIRE,MTOTAL,MOLD,FRM,IPROD,UPTOTAL) CHAR(7);
      15       DCL TOTAL CHAR(10);
      16       DCL (ITOTAL,TINV,TSHIP,LPROD) FIXED BIN(31);
      17       DCL (PROD,UPEND,UPPER) FLOAT DEC(10,3);
      18       DCL (INVEN1,OBJ1) CHAR(7);
      19       DCL (BAL,SH1,SH2,SH3,SH4) FIXED BIN(15);
      20       DCL XCODE CHAR(2);
      21       DCL STAR2 CHAR(2) INIT('**');
      22       DCL XL CHAR(1) INIT('L');
    STMT


         /**********************************************/
         /***  READ DYNAMIC FORECASTING INFORMATION  ***/
         /**********************************************/

              L=1;
              PUT PAGE EDIT('FORECASTING - PRODUCTION INFORMATION') (X(35),A);
              PUT SKIP(2);
              PUT EDIT('FCD','  INITIAL ','1ST WEEK','2ND WEEK','3RD WEEK',
                 '4TH WEEK','SAFETY','BAL','INVENTORY','SHIPPED','SHIPPED',
                 'SHIPPED','SHIPPED','STOCK')
                 (X(22),A,6(X(8),A),SKIP(1),X(22),A,6(X(9),A));
              PUT SKIP(1);

         FORECAST:
              GET FILE(DYNAM) EDIT(XCODE,BAL,SH1,SH2,SH3,SH4)
                 (COL(1),A(2),5(F(10)));
              IF XCODE=STAR2 THEN GO TO START;

              DO J=L TO NUM;
              IF XCODE¬=CODE(J) THEN GO TO NEXT_CODE;
              FCD(J)=SH1-BAL;
              INVBAL(J)=BAL;
              SHIP1(J)=SH1;
              SHIP2(J)=SH2;
              SHIP3(J)=SH3;
              SHIP4(J)=SH4;
              PUT SKIP EDIT(XCODE,FCD(J),INVBAL(J),SHIP1(J),SHIP2(J),SHIP3(J),
                 SHIP4(J),SAFETY(J)) (X(7),A(2),7(X(9),F(7)));
              L=L+1;
              GO TO FORECAST;
         NEXT_CODE:
              END;

         /***  ERROR SECTION OF INPUT  ***/

              PUT SKIP EDIT('***ERROR - CODE ',XCODE,' NOT IN STATIC FILE',
                 BAL,SH1,SH2,SH3,SH4) (A,A(2),A,4(F(10)));
              STOP;
    STMT


             /************************/
             /*** PRINT ROW CARDS ***/
             /************************/

       44    START:
                   ITOTAL=0;
       45          TINV=0.0;
       46          TSHIP=0.0;
       47          DO I=1 TO NUM;
       48          TINV=TINV+INVBAL(I);
       49          ITOTAL=ITOTAL+SHIP2(I)+SHIP1(I);
       50          TSHIP=TSHIP+(SHIP2(I)+FCD(I)+SAFETY(I))/PACK(I);
       51          END;
       52          PUT SKIP(2) EDIT('TOTAL PARTS SHIPPED IN TWO WEEKS = ',
                      ITOTAL)  (X(10),A,F(10));
       53          PUT SKIP EDIT('TOTAL INITIAL INVENTORY = ',TINV)  (X(10),A,F(10));

       54          BUF=' ';

       55          SUBSTR(BUF,1,4)='NAME';
       56          SUBSTR(BUF,15,4)='FOAM';
       57          WRITE FILE(CARD) FROM(BUF);   BUF=' ';

       59          BUF='ROWS';
       60          WRITE FILE(CARD) FROM(BUF);   BUF=' ';

       62          BUF=' N  OBJ';
       63          WRITE FILE(CARD) FROM(BUF);   BUF=' ';

       65          DO I=1 TO NUM;
       66          BUF=' L  ' || CODE(I) || 'REQ';
       67          WRITE FILE(CARD) FROM(BUF);   BUF=' ';
       69          END;

       70          DO I=1 TO NUM;
       71          BUF=' L  ' || CODE(I) || 'ADD';
       72          WRITE FILE(CARD) FROM(BUF);   BUF=' ';
       74          END;

       75          DO I=1 TO NUM;
       76          BUF=' G  ' || CODE(I) || 'SUB';
       77          WRITE FILE(CARD) FROM(BUF);   BUF=' ';
       79          END;

       80          DO I=1 TO NUM;
       81          BUF=' L  ' || CODE(I) || 'FRM';
       82          WRITE FILE(CARD) FROM(BUF);   BUF=' ';
       84          END;
    STMT

      85         DO I=1 TO NUM;
      86         BUF=' G  ' || CODE(I) || 'END';
      87         WRITE FILE(CARD) FROM(BUF);   BUF=' ';
      89         END;

      90         BUF=' E  LINCAP';
      91         WRITE FILE(CARD) FROM(BUF);   BUF=' ';

      93         BUF=' L  SMALL';
      94         WRITE FILE(CARD) FROM(BUF);   BUF=' ';

      96         BUF=' L  LARGE';
      97         WRITE FILE(CARD) FROM(BUF);   BUF=' ';

      99         BUF=' L  WIRE';
     100         WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     102         BUF=' L  INVEN';
     103         WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     105         BUF=' L  UPENDER';
     106         WRITE FILE(CARD) FROM(BUF);   BUF=' ';
    STMT


          /***************************/
          /*** PRINT COLUMN CARDS ***/
          /***************************/

     108        BUF='COLUMNS';
     109        WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     111        BUF='    DEBE      ''MARKER''';
     112        SUBSTR(BUF,40,8)='''INTORG''';
     113        WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     115        DO I=1 TO NUM;
     116        INVEN(I)=2.0*RATE(I)/PACK(I);
     117        PUT STRING(INVEN1) EDIT(INVEN(I)) (F(7,2));
          /***** THE OBJ(I) REPRESENTS THE AVERAGE INVENTORY STORAGE COSTS *****/
          /***** PER 2 WEEK PERIOD.  10% VALUE OF PART OVER 1/26TH YEAR.   *****/
     118        OBJ(I)=RATE(I)*VALUE(I)/260.0;
     119        PUT STRING(OBJ1) EDIT(OBJ(I)) (F(7,2));
     120        DO J=1 TO 4;
     121        SUBSTR(BUF,5,5)=CODE(I) || 'II' || SIZE(I);

     122        IF J=1 THEN DO;
     123             SUBSTR(BUF,15,5)=CODE(I) || 'REQ';
     124             SUBSTR(BUF,33,3)='1.0';
     125             SUBSTR(BUF,40,5)=CODE(I) || 'ADD';
     126             SUBSTR(BUF,58,3)='1.0';
     127             WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     129             END;

     130        IF J=2 THEN DO;
     131             SUBSTR(BUF,15,5)=CODE(I) || 'SUB';
     132             SUBSTR(BUF,33,3)='1.0';
     133             SUBSTR(BUF,40,6)='LINCAP';
     134             SUBSTR(BUF,58,3)='1.0';
     135             WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     137             END;

     138        IF J=3 THEN DO;
     139             SUBSTR(BUF,15,5)='INVEN';
     140             SUBSTR(BUF,29,7)=INVEN1;
     141             SUBSTR(BUF,40,3)='OBJ';
     142             SUBSTR(BUF,54,7)=OBJ1;
     143             WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     145             END;

     146        IF J=4 THEN DO;
     147             SUBSTR(BUF,15,5)=CODE(I) || 'PRM';
     148             SUBSTR(BUF,33,3)='1.0';
     149             UPEND=4.0/PACK(I);
    STMT


     150                IF CODE(I)='BN' | CODE(I)='BR'  THEN UPEND=UPEND/2.0;
     151                SUBSTR(BUF,40,5)=CODE(I) || 'END';
     152                PUT STRING(UPTOTAL) EDIT(-UPEND) (F(7,4));
     153                SUBSTR(BUF,54,7)=UPTOTAL;
     154                WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     156                END;
     157           END;
     158           END;

     159           DO I=1 TO NUM;
     160           DO J=1 TO 2;
     161           SUBSTR(BUF,5,5)=CODE(I) || 'UP' || SIZE(I);

     162           IF J=1 THEN DO;
     163                SUBSTR(BUF,15,7)= 'UPENDER';
     164                SUBSTR(BUF,33,3)='1.0';
     165                SUBSTR(BUF,40,5)=CODE(I) || 'END';
     166                SUBSTR(BUF,58,3)='1.0';
     167                WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     169                END;

     170           IF J=2 THEN DO;
     171                SUBSTR(BUF,15,3)='OBJ';
     172                SUBSTR(BUF,33,3)='1.0';
     173                WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     175                END;
     176           END;
     177           END;

     178           DO I=1 TO NUM;
     179           PUT STRING(WIRE) EDIT(LOAD(I)) (F(7,1));
     180           DO J=1 TO 2;
     181           SUBSTR(BUF,5,5)=CODE(I) || 'AA' || SIZE(I);

     182           IF J=1 THEN DO;
     183                SUBSTR(BUF,15,3)='OBJ';
     184                SUBSTR(BUF,33,3)='7.5';
     185                SUBSTR(BUF,40,5)='SMALL';
     186                SUBSTR(BUF,58,3)='1.0';
     187                IF SIZE(I)=XL THEN DO;
     188                     SUBSTR(BUF,40,5)='LARGE';
     189                     SUBSTR(BUF,57,4)=' 1.0';
     190                     END;
     191                WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     193                END;

     194           IF J=2 THEN DO;
     195                SUBSTR(BUF,15,5)=CODE(I) || 'ADD';
     196                SUBSTR(BUF,32,4)='-1.0';
    STMT

     197               SUBSTR(BUF,40,4)='WIRE';
     198               SUBSTR(BUF,54,7)=WIRE;
     199               WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     201               END;
     202           END;
     203           END;

     204           DO I=1 TO NUM;
     205           PUT STRING(WIRE) EDIT(-LOAD(I)) (F(7,1));
     206           IF LOAD(I)=0.0 THEN PUT STRING(WIRE) EDIT (LOAD(I)) (F(7,1));
     207           DO J=1 TO 2;
     208           SUBSTR(BUF,5,5)=CODE(I) || 'SS' || SIZE(I);

     209           IF J=1 THEN DO;
     210               SUBSTR(BUF,15,3)='OBJ';
     211               SUBSTR(BUF,33,3)='7.5';
     212               SUBSTR(BUF,40,5)='SMALL';
     213               SUBSTR(BUF,57,4)='-1.0';
     214               IF SIZE(I)=XL THEN DO;
     215                   SUBSTR(BUF,40,5)='LARGE';
     216                   SUBSTR(BUF,57,4)='-1.0';
     217                   END;
     218               WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     220               END;

     221           IF J=2 THEN DO;
     222               SUBSTR(BUF,15,5)=CODE(I) || 'SUB';
     223               SUBSTR(BUF,33,3)='1.0';
     224               SUBSTR(BUF,40,4)='WIRE';
     225               SUBSTR(BUF,54,7)=WIRE;
     226               WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     228               END;
     229           END;
     230           END;

     231           DO I=1 TO NUM;
     232           SUBSTR(BUF,5,5)=CODE(I) || 'FF' || SIZE(I);
     233           SUBSTR(BUF,15,5)=CODE(I) || 'FRM';
     234           SUBSTR(BUF,32,4)='-1.0';
     235           SUBSTR(BUF,40,3)='OBJ';
     236           SUBSTR(BUF,57,4)='75.0';
     237           WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     239           END;

     240           BUF='     FINE       ''MARKER''';
     241           SUBSTR(BUF,40,8)='''INTEND''';
     242           WRITE FILE(CARD) FROM(BUF);   BUF=' ';
```

```
   STMT


        /***********************************/
        /*** PRINT THE RIGHT HAND SIDES ***/
        /***********************************/

    244          BUF='RHS';
    245          WRITE FILE(CARD) FROM(BUF);    BUF=' ';

    247          DO I=1 TO NUM;
    248          PUT STRING(MOLD) EDIT(MOLDS(I)) (F(7,1));
    249          SUBSTR(BUF,5,3)='RHS';
    250          SUBSTR(BUF,15,5)=CODE(I) || 'REQ';
    251          SUBSTR(BUF,29,7)=MOLD;
    252          WRITE FILE(CARD) FROM(BUF);    BUF=' ';
    254          END;

    255          DO I=1 TO NUM;
    256          PUT STRING(MTOTAL) EDIT(MTOT(I)) (F(7,1));
    257          SUBSTR(BUF,5,3)='RHS';
    258          SUBSTR(BUF,15,5)=CODE(I) || 'ADD';
    259          SUBSTR(BUF,29,7)=MTOTAL;
    260          WRITE FILE(CARD) FROM(BUF);    BUF=' ';
    262          END;

    263          DO I=1 TO NUM;
    264          PUT STRING(MTOTAL) EDIT(MTOT(I)) (F(7,1));
    265          SUBSTR(BUF,5,3)='RHS';
    266          SUBSTR(BUF,15,5)=CODE(I) || 'SUB';
    267          SUBSTR(BUF,29,7)=MTOTAL;
    268          WRITE FILE(CARD) FROM(BUF);    BUF=' ';
    270          END;

    271          DO I=1 TO NUM;
    272          PUT STRING(FRM) EDIT(FRAME(I)) (F(7,1));
    273          SUBSTR(BUF,5,3)='RHS';
    274          SUBSTR(BUF,15,5)=CODE(I) || 'FRM';
    275          SUBSTR(BUF,29,7)=FRM;
    276          WRITE FILE(CARD) FROM(BUF);    BUF=' ';
    278          END;

    279          SUBSTR(BUF,5,3)='RHS';
    280          SUBSTR(BUF,15,6)='LINCAP';
    281          SUBSTR(BUF,30,6)='181.00';
    282          WRITE FILE(CARD) FROM(BUF);    BUF=' ';

    284          SUBSTR(BUF,5,3)='RHS';
    285          SUBSTR(BUF,15,5)='SMALL';
    286          SUBSTR(BUF,32,4)='5.00';
    287          WRITE FILE(CARD) FROM(BUF);    BUF=' ';
    STMT

     289        SUBSTR(BUF,5,3)='RHS';
     290        SUBSTR(BUF,15,5)='LARGE';
     291        SUBSTR(BUF,32,4)='5.00';
     292        WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     294        SUBSTR(BUF,5,3)='RHS';
     295        SUBSTR(BUF,15,5)='WIRE';
     296        SUBSTR(BUF,32,4)='5.00';
     297        WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     299        WARE=WARE+TSHIP;
     300        PUT STRING(TOTAL) EDIT(WARE) (F(10,1));

     301        SUBSTR(BUF,5,3)='RHS';
     302        SUBSTR(BUF,15,5)='INVEN';
     303        SUBSTR(BUF,26,10)=TOTAL;
     304        WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     306        SUBSTR(BUF,5,3)='RHS';
     307        SUBSTR(BUF,15,7)='UPENDER';
     308        SUBSTR(BUF,32,4)='33.0';
     309        WRITE FILE(CARD) FROM(BUF);   BUF=' ';
    STMT

          /*************************************/
          /*** PRINT THE MOLD RANGE VALUES ***/
          /*************************************/

     311        BUF='RANGES';
     312        WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     314        DO I=1 TO NUM;
     315        PUT STRING(MOLD) EDIT(MOLDS(I)) (F(7,1));
     316        SUBSTR(BUF,5,5)='RANGE';
     317        SUBSTR(BUF,15,5)=CODE(I) || 'REQ';
     318        SUBSTR(BUF,29,7)=MOLD;
     319        WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     321        END;

     322        SUBSTR(BUF,5,5)='RANGE';
     323        SUBSTR(BUF,15,4)='WIRE';
     324        SUBSTR(BUF,32,4)='10.0';
     325        WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     327        SUBSTR(BUF,5,5)='RANGE';
     328        SUBSTR(BUF,15,5)='SMALL';
     329        SUBSTR(BUF,32,4)='10.0';
     330        WRITE FILE(CARD) FROM(BUF);   BUF=' ';

     332        SUBSTR(BUF,5,5)='RANGE';
     333        SUBSTR(BUF,15,5)='LARGE';
     334        SUBSTR(BUF,32,4)='10.0';
     335        WRITE FILE(CARD) FROM(BUF);   BUF=' ';
```

```
STMT

          /*********************************/
          /*** PRINT THE BOUNDS SECTION ***/
          /*********************************/

 337           BUF='BOUNDS';
 338           WRITE FILE(CARD) FROM(BUF);   BUF=' ';

 340           DO I=1 TO NUM;
 341           PUT STRING(MOLD) EDIT(MOLDS(I)) (F(7,1));
 342           SUBSTR(BUF,2,8)='UP BOUND';
 343           SUBSTR(BUF,15,5)=CODE(I) || 'II' || SIZE(I);
 344           SUBSTR(BUF,29,7)=MOLD;
 345           WRITE FILE(CARD) FROM(BUF);   BUF=' ';
 347           END;

 348           DO I=1 TO NUM;
 349           SUBSTR(BUF,2,8)= 'UP BOUND';
 350           SUBSTR(BUF,15,5)=CODE(I) || 'UP' || SIZE(I);
 351           SUBSTR(BUF,33,3)='6.0';
 352           WRITE FILE(CARD) FROM(BUF);   BUF=' ';
 354           END;

 355           DO I=1 TO NUM;
 356           SUBSTR(BUF,2,8)= 'UP BOUND';
 357           SUBSTR(BUF,15,5)=CODE(I) || 'AA' || SIZE(I);
 358           SUBSTR(BUF,32,4)='20.0';
 359           IF CODE(I)='BN' | CODE(I)='BR'  THEN SUBSTR(BUF,32,4)='40.0';
 360           WRITE FILE(CARD) FROM(BUF);   BUF=' ';
 362           END;

 363           DO I=1 TO NUM;
 364           SUBSTR(BUF,2,8)= 'UP BOUND';
 365           SUBSTR(BUF,15,5)=CODE(I) || 'SS' || SIZE(I);
 366           SUBSTR(BUF,32,4)='20.0';
 367           IF CODE(I)='BN' | CODE(I)='BR'  THEN SUBSTR(BUF,32,4)='40.0';
 368           WRITE FILE(CARD) FROM(BUF);   BUF=' ';
 370           END;

 371           DO I=1 TO NUM;
 372           SUBSTR(BUF,2,8)= 'UP BOUND';
 373           SUBSTR(BUF,15,5)=CODE(I) || 'FF' || SIZE(I);
 374           SUBSTR(BUF,32,4)='2.00';
 375           IF CODE(I)='BN' | CODE(I)='BR'  THEN SUBSTR(BUF,32,4)='4.00';
 376           WRITE FILE(CARD) FROM(BUF);   BUF=' ';
 378           END;
```

```
    STMT

           /**********************************************************/

     379         DO I=1 TO NUM;
     380         PROD=(.70*SHIP1(I)+.20*SHIP2(I)+.05*SHIP3(I)+.05*SHIP4(I)-
                    INVBAL(I)+SAFETY(I))/RATE(I);
     381         LPROD=PROD;
     382         IF LPROD < 0 THEN LPROD=0;
     383         IF LPROD > MOLDS(I)  THEN LPROD=MOLDS(I);
     384         PUT STRING(IPROD) EDIT(LPROD) (F(7,1));
     385         SUBSTR(BUF,2,8)='LO BOUND';
     386         SUBSTR(BUF,15,5)=CODE(I) || 'II' || SIZE(I);
     387         SUBSTR(BUF,29,7)=IPROD;
     388         WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     390         END;

     391         BUF='ENDATA';
     392         WRITE FILE(CARD) FROM(BUF);   BUF=' ';
     394         CLOSE FILE(CARD);

     395   FINISH:
                 END GENER;
```

```
MM        MM    IIIIIIIIII   PPPPPPPPPPP
MMM      MMM    IIIIIIIIII   PPPPPPPPPPPP
MMMM    MMMM        II       PP        PP
MM MM  MM MM        II       PP        PP
MM  MMMM  MM        II       PP        PP
MM   MM   MM        II       PPPPPPPPPPPP
MM        MM        II       PPPPPPPPPPP
MM        MM        II       PP
MM        MM        II       PP
MM        MM        II       PP
MM        MM    IIIIIIIIII   PP
MM        MM    IIIIIIIIII   PP
```

```
                         SOURCE LISTING

      STMT

         1  MIP:     PROC REORDER;

            /*******************************************/
            /*****  MPSX 370 - MIP DECLARATIONS  *****/
            /*******************************************/

         2        DCL  (ANALYZE,ASSIGN,BCD,BCDOUT,CHECK,CLOSEF,COMMON,
                        CONVERT,COPY   ,COPYOLD,CRASH,DPLBOOT,DPLUSER,
                        DUAL,EXIT,EXPORT,FLAGS,FORCE,FREEORE,IMPORT,INQUIRE,INSERT,
                        INVALUE,INVERT,MIXBUND,MIXFIX,MIXFLOW,MIXSAVE,MIXSART,
                        MIXSATS,MODIFY,MGRW,PARACOL,PARAOBJ,PARARHS,
                        PARARIM,PARAROW,PICTURE,PRIMAL,PROBEMS,PUNCH,RANGE,
                        RECRATE,REDUCE,REPORT,RESTORE,RETREVE,REVISE,SAVE,
                        SAVERHS,SCALE,SELECT,SELIST,SETREP,SETUP,SOLUION,STATUS,
                        TIME,TRACE,TRANCOL,TRANROW      ) ENTRY EXTERNAL;

            /*****  ALGORITHMIC TOOLS EXTERNAL ENTRIES  *****/

         3        DCL  (PRICEP1,PRICED1,PREMUL,POSTMUL,INVCTL1,GETVEC1,
                        FTRANU1,FTRANL1,FIXVEC1,ELIMN1,CHUZR1,BTRANU1,BTRANL1)
                        ENTRY EXTERNAL;

            /*****  DPLPLICR MACRO = MPSX/370 COMMUNICATION REGION  *****/

         4  DCL DPLSTR(768) DEC FLOAT(16) EXTERNAL INIT(0.0);
         5  DCL DPLPTR PTR EXTERNAL ;
         6  DPLPTR = ADDR(DPLSTR);
         7  DCL 1 DPL ALIGNED BASED(DPLPTR),
                 2 XDUM1 CHAR(16), 2 XCORE BIN FIXED(31), 2 XDUM2 CHAR(16),
                 2 (XTITLE,XSUBTITL) PTR, 2 XDUM3 CHAR(16), 2 XVERSMOD CHAR(5),
                 2 XENVIRON UNALIGNED,
                     3 (XPLI,XTSO,XCONT,XDOS,XREPORT,XATTN,XMPSX,XCMS)BIT(1),
                 2 XDUM4 CHAR(5), 2 XMTRDSW BIT(8), 2 XDUM4C CHAR(16),
                 2 (XINF,XLUCTRL) DEC FLOAT(6),
                 2 (XMXPCITX,XMXWOVF,XFORMSOS,XSEPTERM,XTRYPIV,XTRYPIVB,XTRYPIVX,
                         XCMSXT)BIN FIXED(31),
                 2 (XPBNAME,XOLDNAME,XOBJ,XCHROW,XROW) CHAR(8),
                 2 (XRHS,XCHCOL,XCOLUMN,XDATA,XSAVERHS,XBOUND,XRANGE) CHAR(8),
                 2 XDUM6 CHAR(28),
                 2 XPBSTATS,
                     3 (XM,X4M,X8M,XJ,XELEM,XLMINI,XLMAXI,XMPACVEC) BIN FIXED(31),
                     3 (XBIGBAS,XSGROWS,XGUBTYPE,XMXJ,XMXSOS) BIN FIXED(31),
                 2 XDUM7 CHAR(8),
                 2 (XPARAM,XPHI,XTHETA,XXSI,XZETA) DEC FLOAT(16),
                 2 (XTAU,XSIG,XT) BIN FIXED(31),
                 2 XDUM8 CHAR(68),
STMT

                  2 (XPARPRT,XPARDELT,XSCALE,XEPS,XPARMAX) DEC FLOAT(6),
                  2 XPROCNAM CHAR(8), 2 (XFUNCT,XSIF) DEC FLOAT(16),
                  2 (XNIF,XNEGDJ,XITERNO,XMAJIT,XERROR) BIN FIXED(31),
                  2 (XKJINC,XKJOUT) BIN FIXED(31), 2 XINCDJ DEC FLOAT(6),
                  2 XTRANTIM BIN FIXED(31),
                  2 XINVNO,
                      3 (XINVERNO,XTRANNO,XTIMORG) BIN FIXED(31),
                  2 XPARSW BIT(8), 2 XDUM10 CHAR(7),
                  2 XALGSW UNALIGNED,
                      3 XDUM11 CHAR(3), 3 XDUM12 BIT(3),
                      3 (XMIXPHAS,XDUM13,XLU,XMIP,XSEP) BIT(1),
                  2 (XPRICE,XP,XINTVAL,XFREQINV) BIN FIXED(31),
                  2 (XCLOCKSW,XOLDINV,XLOGCAPT,XCHECKSW,XTRANSW) BIN FIXED(31),
                  2 (XDZPCT,XDJMIN,XDJPCT,XSCLERR,XTRUSCL) DEC FLOAT(6),
                  2 (XDJSCALE,XCIRCLE,XDEGENSW,XSCYCLE) BIN FIXED(31),
                  2 (XRECTIFY,XRECTNO,XNOFREE) BIN FIXED(31),
                  2 XDUM14 CHAR(12),
                  2 (XFREQ3,XFREQ2,XFREQ1,XDELTM,XLASTIM) BIN FIXED(31),
                  2 (XFREQLGO,XFREQLGA,XNOPRINT,XTIMES) BIN FIXED(31),
                  2 XDUM15 CHAR(92),
                  2 (XTOLPIVS,XDUM15C(18),XSSCALE) DEC FLOAT(6),
                  2 (XDUM15F(8),XMNO,XMSIZE,XCOLRC,XRHSRC,XDUM15H(4),XCYCLESW)
                          BIN FIXED(31),
                  2 XDUM15L CHAR(118),
                  2 XETASW UNALIGNED,
                      3 (XETAFULL,XETAPART,XETALU,XETATN,XETAACCU) BIT(1),
                      3 XDUM15N BIT(3), 3 (XINVFULL,XINVLU) BIT(1),
                  2 XINVDENS DEC FLOAT(6), 2 (XPARTINV,XINVCORE) BIN FIXED(31),
                  2 XDUM16 CHAR(76),
                  2 XMXPTR,
                      3 (XH,XKM,XMXLUDY,XMXFRAC,XMXCNE) PTR,
                  2 (XV,XG) PTR,
                  2 XPIO BIN FIXED(31), 2 XPI(3) PTR,
                  2 XALPHAO BIN FIXED(31), 2 XALPHA(5) PTR,
                  2 XVREGO BIN FIXED(31), 2 XVREG(5) PTR,
                  2 (XUPLIMIT,XSUPLMT,XSCLORG,XCOL,XUSTK,XTN) PTR,
                  2 XWO,
                      3 (XWNO,XWNOMAX) BIN FIXED(15),
                      3 XW(12) PTR,
                  2 (XDUM16B,XDJO) BIN FIXED(31), 2 XDJ(5) DEC FLOAT(16),
                  2 XDUM16D CHAR(176),
                  2 (XLM,XFREE,XPROC,XETACORE) BIN FIXED(31),
                  2 (XJCORE,XMXWCORE,XWREG,XNODES) BIN FIXED(31),
                  2 XDUM17 CHAR(36),
                  2 XSETLB BIN FIXED(31),
                  2 (XTOLZE,XTOLREL,XTOLV,XTOLDJ,XTOLPIV,XTOLERR) DEC FLOAT(6),
                  2 (XTOLCHK,XROWCHK,XTOLINV,XTOLI1,XTOLI2) DEC FLOAT(6),
                  2 (XTOLELEM,XKAPPA,XTRANCHK,XRHO,XZI,XPRICHK) DEC FLOAT(6),
                  2 (XVECNORM,XDELTADJ,XTOLWRIT,XDJREL) DEC FLOAT(6),
    STMT


               2 XDUM18 CHAR(12),
               2 (XOBJSC,XCHROWSC,XDUM18B(6) ) DEC FLOAT(16),
               2 XSCALMAT BIN FIXED(31), 2 XTOLDJS DEC FLOAT(6),
               2 (XR1C2,XR2C2) CHAR(8), 2 (XR1C1,XR2C1) DEC FLOAT(6),
               2 XERRCNT BIN FIXED(31), 2 XTRANCHS DEC FLOAT(6),
               2 XREAL01(10) DEC FLOAT(6),
               2 XINT01(10) BIN FIXED(31),
               2 XCHAR01(10) CHAR(8),
               2 (XREDUCE,XSORTA,XSTART,XVECTOR,XENDSW,XMXERBRT) BIN FIXED(31),
               2 XDUM19 CHAR(288),
               2 (XPIV,XZERO,XD1,XDM1) DEC FLOAT(16),
               2 (XDUM20(3),XITERINV,XMAJINV,XMAJNO,XDUM20B(7))BIN FIXED(31),
               2 (XTOLVREL,XTOLZREL,XKPMAX,XKPERR,XDEPS,XMXMUTDJ)DEC FLOAT(6),
               2 (XMXTDJ,XSIGMA2,XTOLI3) DEC FLOAT(6), 2 XOPDEGN BIN FIXED(31),
               2 XDUM21 CHAR(36),

          /*****  MIXED INTEGER CELLS  *****/

               2 XMXSTRAT CHAR(4),
               2 (XMXRATIO,XMXDROP,XMXBESTP,XMXBESTE,XMXSTEP,XMXSCAN,XMXSCF,
                  XMXSCE,XMXQI1,XMXQI2,XMXQI3,XMXTOLI,XMXST1,
                  XMXST2) DEC FLOAT(6),
               2 (XMXSTIT,XMXPCIT) BIN FIXED(31),
               2 (XMXPCPAR,XMXPCDUA,XMXTOLZE) DEC FLOAT(6),
               2 (XMXFRN,XMXNNO,XMXJ1,XMXJ2,XMXSWT,XMXSSWT) BIN FIXED(31),
               2 (XMXOVFLC,XMXBIN,XMXFNLOG) BIN FIXED(31),
               2 XMXERBR BIN FIXED(31), 2 XMXBIF DEC FLOAT(16),
               2 (XMXCAPT,XMXLOG) BIN FIXED(31),
               2 (XDUM22 CHAR(16) ,XMXPOE DEC FLOAT(6),XDUM22D CHAR(188)) ,
               2 XDUM23 CHAR(302),
               2 XSPIE BIN FIXED(31),
               2 (XDUM24 CHAR(88),XMXMAXNO BIN FIXED(31)),
               2 (XMXSTART,XMXSOSWT,XOPSTART,XOPRESTR,XOPSAVE,
                  XOPSET,XRBOUND,XRRESTA,XRNODEL,XBNEWPR) CHAR(8),
               2 (XOPFREQS,XINVCT,XERRNO,XMAXCT) BIN FIXED(31),
               2 (XMXNODE,XMXPCCNT) BIN FIXED(31), 2 XTOLDJ1 DEC FLOAT(6),
               2 XMXOPTMC BIN FIXED(31), 2 XMINMAX CHAR(8),
               2 XUSER(50) BIN FIXED(31);
```

```
STMT

          /***********************************************/
          /*****  INITIALIZATION TITLE & SUBTITLE  *****/
          /***********************************************/

     8    DCL 1 $TIT EXTERNAL STATIC,
              2 $DUM0 CHAR(20) INIT(' '),
              2 $TITLE CHAR(80) INIT('      ECL EXECUTION'),
              2 $DUM1 CHAR(6) INIT('  PAGE'),
              2 $PAGE PIC 'ZZZZZ9' INIT(0),
              2 $DUM2 CHAR(8) INIT(' '),
              2 $DATE CHAR(6);
     9    DCL DPLPRNT FILE VARIABLE EXTERNAL;

          /*********************************************************/
          /*****  ATTRIB MACRO = REMPLACEMENT ECL ATTRIBUTES  *****/
          /*********************************************************/
```

```
PL/I OPTIMIZING COMPILER         MIP:     PROC REORDER;


    STMT


           /*****************************************************/
           /*****  DPLINIPR MACRO = INITIALIZE SYSPRINT  *****/
           /*****************************************************/
      10   DCL 1 $SUBTITL BASED(XSUBTITL),
                2 $NSUB BIN FIXED(31),
                2 $ASUB (MSUBTL REFER($NSUB)) PTR;
      11   DCL $MSG CHAR(132) VARYING BASED($PSUB),
                $HEAD CHAR(126) BASED(XTITLE),
                $PSUB PTR;
      12        XTITLE = ADDR($TIT);
      13        $DATE = DATE;
      14        XSUBTITL = NULL;
      15   DCL (NULL,DATE) BUILTIN;
      16   DCL DPLINSZ BIN FIXED(15) INIT(132);
      17        DPLPRNT = SYSPRINT;

      18        OPEN FILE(SYSPRINT) PAGESIZE(57) LINESIZE(DPLINSZ) ;

      19              ON ENDPAGE(DPLPRNT)
                       BEGIN ;
      20                    DCL $I BIN FIXED(15,0) ;
      21                    $PAGE = $PAGE + 1 ;
      22                    IF XTITLE=NULL |(XTSO='1'B & DPLPRNT=SYSPRINT) THEN
                                      PUT PAGE FILE(DPLPRNT) ;
      23                    ELSE
                            DO ;
      24                         PUT LIST($HEAD) PAGE FILE(DPLPRNT) ;
      25                         PUT SKIP FILE(DPLPRNT) ;
      26                    END ;
      27                    IF XSUBTITL ¬= NULL THEN
                              DO ;
      28                         DO $I= 1 TO $NSUB ;
      29                              $PSUB = $ASUB($I) ;
      30                              PUT SKIP EDIT($MSG) (A) FILE(DPLPRNT) ;
      31                         END ;
      32                      PUT SKIP FILE(DPLPRNT);
      33                      END ;
      34               END ;
```

```
STMT

          /*********************************************************************/
          /***  DPLTOL MACRO = DPLPLICR CELLS TOLERANCES INITIALIZATIONS  ***/
          /*********************************************************************/

  35               XCLOCKSW , XFREQLGO , XINVCORE , XPARTINV = 1 ;
  36               XSCALMAT=-1 ;
  37               XCHECKSW=20;XOLDINV =   10     ;
  39               XSEPTERM=   20;XFREE=20480;XCORE=16711680;
  42               XDJPCT =0.001; XTOLI2=0.01      ; XTOLZE   =   1.0E-30;
  45               XTOLREL,   XTOLELEM =   1.0E-10;
  46               XTOLWRIT , XTOLERR  = 1.0E-6 ;
  47               XTRANCHK = 1.0E-9 ;
  48               XTOLPIV  , XTOLINV  =   1.0E-6 ;
  49               XTOLDJ   =   1.0E-8  ;
  50               XTOLDJ1  =   1.0E-5  ;
  51               XDJREL , XTOLI1 = 1E-11 ;  XTOLZREL=1E-13 ;
  53               XTOLV=1E-5 ;  XTOLI3= 0.01 ;
  55               XKPERR=1E+8 ;  XDEPS=0.1 ;
  57               XTOLCHK=1E-8 ; XKPMAX =1E+6 ; XTOLVREL = 1E-9 ;
  60               XRECTIFY=100;XINVDENS=1.   ; XLUCTRL=1.1 ;
  63               XMXTOLZE=1.0E-5; XMAXCT=5 ;
  65               XMXFNLOG=1;XMXRATIO=0.15;XMXSCF=1.2; XMXSCE=0.5 ;
  69               XMXQI1=0.1;XMXQI2=0.05;XMXQI3=0.5;
  72               XMXTOLI=0.1;XMXST1,XMXST2=2.;
  74               XMXSTIT,XMXPCIT=3 ;
  75               XMXPCPAR=0.33 ; XMXPCDUA=0.66 ;
  77               XRBOUND,XRRESTA,XRNODEL,XOPRESTR,XOPSAVE='        ';
  78               XRNEWPR='NEWPROB';XOPSTART='BEGIN';
  80               XOPFREQS=10;
  81               XMXSTART='STANDARD';XMXSOSWT='        ';
  83               XINF,XMXDROP=1E75;
  84               XMXPCITX=9;
  85               XMXERRRT=50;XMXTDJ=100.;
  87               XTOLPIVS=1E-4;XTRYPIVX=2;
  89               XSPIE=1;
```

```
    STMT

         /*********************************************************************/
         /***  DPLONCD MACRO = DEFINITION ON-UNITS FOR POSSIBLE DEMANDS   ***/
         /*********************************************************************/

     90        ON CONDITION(XCOMERR)
                    BEGIN ;
     91               DCL TOLI2 DEC FLOAT(6)  INIT(XTOLI2) ;
     92               IF XTOLI2 >=0.1 & XTRANCHK >= 1E-6 & XITERNO=XINVERNO THEN
                         DO;
     93                    IF XMIXPHAS THEN
                              DO;
     94                         CALL MIXSAVE;
     95                         CALL MIXSATS;
     96                       END;
     97                    STOP;
     98                  END;
     99               XTOLI2=0.1 ;
    100               IF XERRNO >= 3 THEN
                      DO ;
    101                    XTOLPIV,XTOLINV=1E-4 ;
    102                    IF XERRNO >=5 THEN
                           DO ;
    103                         XTRANCHK=1E-6 ;
    104                         XINVDENS=0. ;
    105                    END ;
    106                    ELSE
                                XTRANCHK=1E-8 ;
    107               END ;
    108                CALL INVERT;
    109               XINVDENS=1.0 ;
    110               XTOLI2=MIN(0.16,TOLI2+TOLI2) ;
    111                IF XINVCT >=0 THEN
                           XERRNO=XERRNO+1 ;
    112                ELSE
                         DO;
    113                    XINVCT = XMAXCT ;
    114                    XERRNO = 1 ;
    115                  END;
    116                 XCONT = '1'B;
    117             END;

    118       ON CONDITION(XDODLTM)
                    BEGIN;
    119                IF XMIXPHAS THEN
                         DO;
    120                    CALL MIXSAVE;
    121                    CALL MIXSATS;
    122                  END;
    123                STOP;
```

```
     STMT


      124           END;

      125      ON CONDITION(XDODUAL)
                    BEGIN;
      126             CALL DUAL;
      127              XCONT = '1'B;
      128           END;

      129      ON CONDITION(XDOFEAS)
                     XCONT = '1'B;

      130      ON CONDITION(XDOINV)
                    BEGIN;
      131            DCL TOLI2 DEC FLOAT(6)  INIT(XTOLI2) ;
      132            IF XERROR ¬= 0 THEN
                          XTOLI2= MAX(XTOLI2,0.1) ;
      133             CALL INVERT;
      134             IF XINVCT<0 & XERROR = 0 THEN
                        DO;
      135                 XERRNO=0;
      136                 XTOLPIV,XTOLINV=1E-6 ;
      137                 XTRANCHK=1E-9 ;
      138                 XTOLI2=MAX(0.5*TOLI2,XTOLI3) ;
      139               END;
      140               ELSE
                          XTOLI2=TOLI2 ;
      141              XCONT = '1'B;
      142           END;

      143      ON CONDITION(XDOLPS)
                    BEGIN;
      144            IF XITERNO ¬= XINVERNO | XRHO*XZI > 1E+4 THEN
                     DO ;
      145                 XERROR=4 ;
      146                 SIGNAL CONDITION(XDOINV) ;
      147            END ;
      148             XCONT = '1'B ;
      149           END;

      150      ON CONDITION(XDONFS)
                    BEGIN;
      151             IF XMIXPHAS THEN
                        DO;
      152                 CALL MIXSAVE;
      153                 CALL MIXSATS;
      154                 STOP;
      155               END;
      156             IF XEPS=0.0 THEN
                        DO;
```

```
    STMT

     157                CALL STATUS;
     158                CALL SOLUION;
     159                STOP;
     160              END;
     161            XEPS=0.0;
     162            XOPDEGN=0;
     163             XCONT = '1'B;
     164          END;

     165     ON CONDITION(XDONMX)  ;

     166     ON CONDITION(XDOOPT)   ;

     167    ON CONDITION(XDOPEMX)  ;

     168    ON CONDITION(XDOPRIM)
                 BEGIN;
     169           IF XMIXPHAS THEN
                     DO;
     170               ON CONDITION(XDONFS);
     171               ON CONDITION(XDOOPT);
     172               ON CONDITION(XDOPEAS)   XCONT = '1'B;
     173             END;
     174           CALL PRIMAL;
     175            XCONT = '1'B;
     176          END;

     177    ON CONDITION(XDOPINT)
                 BEGIN;
     178           CALL SOLUION;
     179           IF XMIXPHAS THEN
                     DO;
     180               IF XMXMAXNO=0 THEN  XCONT = '1'B;
     181               ELSE
                         DO;
     182                   XMXMAXNO=XMXMAXNO-1;
     183                   IF XMXMAXNO¬=0 THEN  XCONT = '1'B;
     184                   ELSE XMXOPTMC=1;
     185                 END;
     186             END;
     187           ELSE  XCONT = '1'B;
     188          END;

     189    ON CONDITION(XDOUNB)
                 BEGIN;
     190           IF XEPS=0.0 THEN
                     DO;
     191               CALL STATUS;
     192               CALL SOLUION;
```

```
STMT


 193                 STOP;
 194               END;
 195             XEPS=0.0;
 196             XOPDEGN=0;
 197              XCONT = '1'B;
 198           END;

 199      ON CONDITION(XMAJERR)
           BEGIN;
 200         CALL STATUS;
 201         STOP;
 202       END;

 203      ON CONDITION(XIOERR)
           BEGIN;
 204         ON CONDITION(XIOERR)  STOP ;
 205         IF XMIXPHAS THEN
               DO;
 206             CALL MIXSAVE;
 207             CALL MIXSATS;
 208           END;
 209         ELSE
               DO;
 210             CALL STATUS;
 211             XDATA='*********';
 212             CALL PUNCH;
 213           END;
 214         STOP;
 215       END;

 216      ON CONDITION(XMINERR)  ;

 217      ON CONDITION(XMXOVFL)
           BEGIN;
 218         CALL MIXSAVE;
 219         CALL MIXSATS;
 220         STOP;
 221       END;

 222      ON CONDITION(XSINULR)
           CALL RETREVE;

 223      ON FINISH CALL EXIT;
```

```
    STMT

            /*********************************************/
            /*****  USER DEFINED MPSX/MIP PROGRAM  *****/
            /*********************************************/

     224        DCL XFUNCT1 CHAR(12);
     225        DCL XMXNNO1 CHAR(6);
     226        DCL INODES BIN FIXED(31);

     227        ON CONDITION(XMXDPRN) BEGIN;
     228           CALL MIXSART('RESTORE','NODE',XMXBIN);
     229           CALL SOLUION('FILE','OUT');
     230           END;

     231        ON CONDITION(XDOPINT) BEGIN;
     232           CALL MIXSAVE('NAME','TREE');
     233           END;

     234            DPLPRNT=OUT;

     235        OPEN FILE(OUT) PRINT PAGESIZE(66) LINESIZE(132);

     236        XOBJ='OBJ';
     237        XFREE=50000;
     238        XRHS='RHS';
     239        XFREQ1=1000;
     240        XDATA='FOAM';
     241        XPBNAME='FOAM';
     242        CALL CONVERT('FILE','CARD');
     243        INODES=1000;
     244        CALL SETUP('BOUND','BOUND','RANGE','RANGE','NODES',INODES);
     245        PUT FILE(OUT) PAGE;
     246        PUT FILE(OUT) PAGE;
            /******************************
                CALL PICTURE;
            ******************************/
     247        CALL PRIMAL;
     248        XFREQ1=0;
     249        XMXPRN=0;
     250        CALL SOLUION;
     251        PUT STRING(XFUNCT1) EDIT(XFUNCT) (F(12,2));
     252        PUT STRING(XMXNNO1) EDIT(XMXNNO) (F(6));
     253        DISPLAY('                CONTINUOUS SOLUTION: OBJ = '|| XFUNCT1 ||
                   '  AT NODE '|| XMXNNO1);
     254        CALL MIXSART;
     255        XMXNNO=300;
     256        XMXPRN=300;
     257        CALL MIXFLOW;
     258        CALL SOLUION('FILE','OUT');
     259        PUT STRING(XFUNCT1) EDIT(XFUNCT) (F(12,2));
    STMT


     260          PUT STRING(XMXNNO1) EDIT(XMXNNO)  (F(6));
     261          DISPLAY('                      INTEGER SOLUTION: OBJ = '|| XFUNCT1 ||
                     '  AT NODE '|| XMXNNO1);

     262          CLOSE FILE(CARD);
     263          CLOSE FILE(OUT);

     264     FINISH:
                  END MIP;
```

```
 SSSSSSSSSS    CCCCCCCCCC   HH        HH  EEEEEEEEEEEE   DDDDDDDDD
SSSSSSSSSSSS  CCCCCCCCCCCC  HH        HH  EEEEEEEEEEEE  DDDDDDDDDD
SS        SS  CC        CC  HH        HH  EE            DD       DD
SS            CC            HH        HH  EE            DD        DD
SSS           CC            HH        HH  EE            DD        DD
 SSSSSSSSS    CC            HHHHHHHHHHHH  EEEEEEEE      DD        DD
  SSSSSSSSS   CC            HHHHHHHHHHHH  EEEEEEEE      DD        DD
         SSS  CC            HH        HH  EE            DD        DD
          SS  CC            HH        HH  EE            DD        DD
SS        SS  CC        CC  HH        HH  EE            DD       DD
SSSSSSSSSSSS  CCCCCCCCCCCC  HH        HH  EEEEEEEEEEEE  DDDDDDDDDD
 SSSSSSSSSS    CCCCCCCCCC   HH        HH  EEEEEEEEEEEE  DDDDDDDDD
```

```
PL/I OPTIMIZING COMPILER        SCHED:   PROC REORDER;

                    SOURCE LISTING

    STMT

      1   SCHED:   PROC REORDER;

      2          DCL (NUM,SHIFT,IPOS,LNUM,ISIM) FIXED BIN(15) STATIC EXTERNAL;
      3          DCL WARE FIXED BIN(31) STATIC EXTERNAL;
      4          DCL (SIZE(50),PLAT(50),TAPE(50)) CHAR(1) STATIC EXTERNAL;
      5          DCL (CODE(50),CLINE(200)) CHAR(2) STATIC EXTERNAL;
      6          DCL VALUE(50) FIXED DEC(7,2) STATIC EXTERNAL;
      7          DCL PART(50) CHAR(9) STATIC EXTERNAL;
      8          DCL (LOAD(50),NUMPRT(50),MOLDS(50),FRAME(50),PACK(50),RATE(50),
                      SHIP1(50),SHIP3(50),SHIP4(50),INVBAL(50),SAFETY(50),
                      FCD(50),SHIP2(50),LPOS(50),LDIFF(50),MTOT(50),LWDIFF(200))
                      FIXED BIN(15) STATIC EXTERNAL;
      9          DCL TITLE CHAR(80) STATIC EXTERNAL;
     10          DCL 1 CHANGE(100) STATIC EXTERNAL,
                      2 (POS,NEED) FIXED BIN(15),
                      2 (ADD,SUB) CHAR(2);
     11          DCL ICOUNT FIXED BIN(15) STATIC EXTERNAL;
     12          DCL LWD FIXED BIN(15) STATIC EXTERNAL;

     13          DCL AVAIL(100) CHAR(2);
     14          DCL SDIFF(100) FIXED BIN(15);
     15          DCL (ITEMP,ITEMP1) CHAR(2);
     16          DCL (SIZE1,SIZE2) FIXED BIN(15);
     17          DCL (ITEMP2,ITEMP3) FIXED BIN(15);
     18          DCL PROBLEM(100) FIXED BIN(15);
     19          DCL REMOVE(100) CHAR(2);

     20          DCL 1 CARD,
                         2 NAME CHAR(8),
                         2 NOCOL FIXED BIN(31) INIT(0),
                         2 DUMMY CHAR(4);

     21          DCL COLUMN(250) CHAR(8);
     22          DCL ROW(250) CHAR(8);
     23          DCL TYPE(500) FIXED BIN(31);
     24          DCL ENDSEC CHAR(8) INIT('$ENDSEC$');
     25          DCL ENDATA CHAR(8) INIT('ENDATA');

     26          DCL VALUES(500) FLOAT DEC(6);
     27          DCL VALUES8(250) FLOAT DEC(16) DEFINED(VALUES);
     28          DCL VALALF8(500) CHAR(8) DEFINED(VALUES);

     29          DCL LAST CHAR(2) INIT('II');
     30          DCL ACTIVE(250) FIXED BIN(31);
     31          DCL CNAME(250) CHAR(8);
     32          DCL (IMAX,ISUB,ISUM,CPOS,ICNT,A1,A2,A3,MM) FIXED BIN(15);
```

```
STMT


  33       DCL (ISTART,IEND,MSIZE,NSIZE,A(3)) FIXED BIN(15);
  34       LSW=0;

      /************************************/
      /***  INITIALIZE THE VARIABLES  ***/
      /************************************/

  35  BEGIN_PGM:
           ADD(*)='  ';
  36       SUB(*)='  ';
  37       POS(*)=0;
  38       NEED(*)=0;
  39       AVAIL(*)='  ';
  40       SDIFF(*)=0;
  41       ACTIVE(*)=0;
  42       CNAME(*)=' ';
  43       TYPE(*)=0;
  44       REMOVE(*)='  ';
  45       PROBLEM(*)=0;

      /***************************************/
      /*****  READ AND WRITE FILE OUT  *****/
      /***************************************/

  46       READ FILE(OUT) IGNORE(1);
  47       READ FILE(OUT) INTO(CARD);
  48       READ FILE(OUT) INTO(ROW);
  49       READ FILE(OUT) INTO(TYPE);
  50       READ FILE(OUT) INTO(VALUES);
  51       READ FILE(OUT) IGNORE(1);
```

```
PL/I OPTIMIZING COMPILER         SCHED:   PROC REORDER;

    STMT

            /********************************************/
            /*****  GET AND PRINT THE ROW SECTION  *****/
            /********************************************/

      52   LAB1:
                  READ FILE(OUT) INTO(CARD);
      53          IF NAME=ENDATA THEN GO TO LAB5;
      54          IF LSW=0  THEN
                  PUT EDIT('SOLUTION OF THE ROW SECTION') (PAGE,X(35),A);
      55          READ FILE(OUT) INTO(ROW);
      56          READ FILE(OUT) IGNORE(1);
      57          IF LSW=0  THEN
                  PUT EDIT((ROW(K) DO K=1 TO NOCOL))
                     (SKIP(3),X(6),(NOCOL) (A(8),X(7)));
      58          IF LSW=0  THEN
                  PUT SKIP(1);

      59   LAB2:
                  READ FILE(OUT) INTO(VALUES);
      60          IF VALALF8(1)=ENDSEC THEN DO;
      61             ICOUNT=1;
      62             GO TO LAB3;
      63             END;

      64          DO K=3 TO 4;
      65          IF VALUES8(K)< 1.E-6 THEN VALUES8(K)=0.0;
      66          IF VALUES8(K) > 1.E10 THEN VALUES8(K)=0.0;
      67          END;

      68          IF LSW=0  THEN
                  PUT EDIT((VALUES8(K) DO K=1 TO (NOCOL-2)),VALALF8(NOCOL-1),
                     VALALF8(NOCOL))
                     (SKIP(1),(NOCOL-2)(F(12,4),X(3)),X(8),A(2),X(9),A(8));
      69          GO TO LAB2;
```

```
STMT

        /***********************************************/
        /*****  GET AND PRINT THE COLUMN SECTION  *****/
        /***********************************************/

  70    LAB3:
              READ FILE(OUT) INTO(CARD);
  71          IF NAME=ENDATA THEN GO TO LAB5;
  72          IF LSW=0  THEN
              PUT EDIT('SOLUTION OF THE COLUMN SECTION') (PAGE,X(35),A);
  73          READ FILE(OUT) INTO(COLUMN);
  74          READ FILE(OUT) IGNORE(1);
  75          IF LSW=0  THEN
              PUT EDIT((COLUMN(K) DO K=1 TO NOCOL))
                 (SKIP(3),X(6),(NOCOL)(A(8),X(7)));
  76          IF LSW=0  THEN
              PUT SKIP(1);

  77    LAB4:
              READ FILE(OUT) INTO(VALUES);
  78          IF VALALF8(1)=ENDSEC THEN GO TO LAB3;
  79          IF VALUES8(1)=0.0 THEN GO TO LAB4;
  80          ACTIVE(ICOUNT)=VALUES8(1)+0.01;
  81          CNAME(ICOUNT)=VALALF8(NOCOL);
  82          ICOUNT=ICOUNT+1;
  83          IF SUBSTR(VALALF8(NOCOL),3,2)¬= LAST THEN PUT SKIP(1);

  84          DO K=3 TO 4;
  85          IF VALUES8(K)< 1.E-6 THEN VALUES8(K)=0.0;
  86          IF VALUES8(K) > 1.E10 THEN VALUES8(K)=0.0;
  87          END;

  88          IF LSW=0  THEN
              PUT EDIT((VALUES8(K) DO K=1 TO (NOCOL-2)),VALALF8(NOCOL-1),
                 VALALF8(NOCOL))
                 (SKIP(1),(NOCOL-2)(F(12,4),X(3)),X(8),A(2),X(9),A(8));
  89          LAST=SUBSTR(VALALF8(NOCOL),3,2);
  90          GO TO LAB4;

  91    LAB5:
              ICOUNT=ICOUNT-1;
  92          CLOSE FILE(OUT);
```

```
PL/I OPTIMIZING COMPILER         SCHED:   PROC REORDER;

    STMT


          /****************************************/
          /***  SORT THE PROBLEM AREA ARRAYS  ***/
          /****************************************/

     93          DO I=1 TO LNUM-1;
     94          DO J=I TO LNUM;
     95          IF LPOS(I) <= LPOS(J)  THEN GO TO SORTP;

     96          ITEMP2=LPOS(I);
     97          ITEMP3=LDIFF(I);
     98          LPOS(I)=LPOS(J);
     99          LDIFF(I)=LDIFF(J);
    100          LPOS(J)=ITEMP2;
    101          LDIFF(I)=ITEMP3;

    102   SORTP:
                 END;
    103          END;
```

```
   STMT

          /********************************************/
          /***  ELIMINATE REDUNDANT PROBLEM AREAS  ***/
          /********************************************/

    104         INDEX=1;
    105         I=1;

    106         DO WHILE(I <= LNUM);
    107         IF LPOS(I)+1=LPOS(I+1)  & LPOS(I)+2=LPOS(I+2)  & I<=LNUM-2 THEN DO;
    108            PROBLEM(INDEX)=LPOS(I+1);
    109            INDEX=INDEX+1;
    110            I=I+3;
    111            END;
    112         ELSE IF LPOS(I)+1=LPOS(I+1)  & I <= LNUM-1  THEN DO;
    113            PROBLEM(INDEX)=LPOS(I);
    114            INDEX=INDEX+1;
    115            I=I+2;
    116            END;
    117         ELSE DO;
    118            PROBLEM(INDEX)=LPOS(I);
    119            INDEX=INDEX+1;
    120            I=I+1;
    121            END;
    122         END;
    123         INDEX=INDEX-1;
    STMT


           /************************************************/
           /*****  PUTTING THE RESULTS OF THE MIP    *****/
           /*****  SUBTRACT SECTION IN REMOVE ARRAY  *****/
           /************************************************/

     124        ICOUNT=1;

     125        DO ISUB=1 TO 250;
     126        IF SUBSTR(CNAME(ISUB),3,2)='SS' THEN GO TO FND_SUB;
     127        END;
     128        PUT SKIP EDIT('  THE PROGRAM STOPS HERE') (A);
     129        STOP;

     130   FND_SUB:
                DO I=1 TO NUM WHILE(SUBSTR(CNAME(ISUB),3,2)='SS');

     131        DO J=1 TO ACTIVE(ISUB);
     132        REMOVE(ICOUNT)=SUBSTR(CNAME(ISUB),1,2);
     133        ICOUNT=ICOUNT+1;
     134        END;

     135   CHK_NEXT_SS:
                ISUB=ISUB+1;
     136        END;

     137        ICOUNT=ICOUNT-1;
    STMT


          /*************************************************/
          /*****  PUTTING THE RESULTS OF THE PROBLEM  *****/
          /*****  SECTION IN THE CHANGE ARRAY         *****/
          /*************************************************/

     138         ICNT=1;
     139         ISW=0;

     140         DO J=1 TO INDEX;
     141         L=1;

     142         DO I=-1 TO 1;
     143         DO K=1 TO NUM;
     144         IF CLINE(PROBLEM(J)+I)=CODE(K) THEN GO TO FOUND;
     145         END;
     146   FOUND:
                 DO LL=1 TO ICOUNT;
     147         IF CODE(K)=REMOVE(LL)  THEN DO;
     148            POS(J)=PROBLEM(J)+I;
     149            SUB(J)=CODE(K);
     150            DO K=LL TO ICOUNT-1;
     151            REMOVE(K)=REMOVE(K+1);
     152            END;
     153            ICOUNT=ICOUNT-1;
     154            GO TO CONTINUE;
     155            END;
     156         END;
     157         A(L)=LOAD(K);
     158         L=L+1;
     159         END;

     160         IMAX=MAX(A(1),A(2),A(3));

     161         IF IMAX=22  THEN DO;
     162            IF A(1)=22 THEN IMAX=MAX(A(2),A(3));
     163            IF A(2)=22 THEN DO;
     164               POS(J)=PROBLEM(J)-1;
     165               SUB(J)=CLINE(PROBLEM(J)-1);
     166               AVAIL(ICNT)=SUB(J);
     167               ICNT=ICNT+1;
     168               POS(INDEX+ISW+1)=PROBLEM(J)+1;
     169               SUB(INDEX+ISW+1)=CLINE(PROBLEM(J)+1);
     170               AVAIL(ICNT)=SUB(INDEX+ISW+1);
     171               ICNT=ICNT+1;
     172               ISW=ISW+1;
     173               GO TO CONTINUE;
     174               END;
     175            IF A(3)=22 THEN IMAX=MAX(A(1),A(2));
     176            END;
```

```
STMT


 177        IF IMAX=A(2) THEN DO;
 178           POS(J)=PROBLEM(J);
 179           SUB(J)=CLINE(PROBLEM(J));
 180           AVAIL(ICNT)=SUB(J);
 181           ICNT=ICNT+1;
 182           GO TO CONTINUE;
 183           END;

 184        IF IMAX=A(1) THEN DO;
 185           POS(J)=PROBLEM(J)-1;
 186           SUB(J)=CLINE(PROBLEM(J)-1);
 187           AVAIL(ICNT)=SUB(J);
 188           ICNT=ICNT+1;
 189           GO TO CONTINUE;
 190           END;

 191        IF IMAX=A(3) THEN DO;
 192           POS(J)=PROBLEM(J)+1;
 193           SUB(J)=CLINE(PROBLEM(J)+1);
 194           AVAIL(ICNT)=SUB(J);
 195           ICNT=ICNT+1;
 196           END;

 197   CONTINUE:
            END;

 198        INDEX=INDEX+ISW;

 199        INDEX=INDEX+1;
 200        ISTART=1;
 201        CPOS=60;

 202        DO L=1 TO ICOUNT;
 203        DO J=ISTART TO 100;

 204        DO KK=1 TO 2;
 205        IF KK=1  THEN K=CPOS-J;
 206                 ELSE K=CPOS+J;
 207        IF K < 1  THEN GO TO NX;
 208        IF REMOVE(L)=CLINE(K)   THEN GO TO FND_LETTER;
 209   NX:  END;
 210        GO TO NX_CHK;
 211   FND_LETTER:
            POS(INDEX)=K;
 212        SUB(INDEX)=CLINE(K);
 213        INDEX=INDEX+1;
 214        ISTART=J+1;
 215        IF L < ICOUNT & REMOVE(L)¬=REMOVE(L+1)   THEN ISTART=1;
 216        GO TO NX_REMOVE;
```

```
    STMT

     217  NX_CHK:
                 END;
     218  NX_REMOVE:
                 END;

     219         INDEX=INDEX-1;
     220         ICOUNT=INDEX;
    STMT


             /**********************************************/
             /*****  PUTTING THE RESULTS OF THE MIP   *****/
             /*****  ADDITION SECTION IN AVAIL ARRAY  *****/
             /**********************************************/

     221           DO IADD=1 TO 250;
     222           IF SUBSTR(CNAME(IADD),3,2)='AA' THEN GO TO FND_ADD;
     223           END;
     224     FND_ADD:
                   DO I=1 TO NUM WHILE(SUBSTR(CNAME(IADD),3,2)='AA');

     225           DO J=1 TO ACTIVE(IADD);
     226           AVAIL(ICNT)=SUBSTR(CNAME(IADD),1,2);
     227           ICNT=ICNT+1;
     228           END;

     229     CHK_NEXT_AA:
                   IADD=IADD+1;
     230           END;

     231           ICNT=ICNT-1;

     232           DO I=1 TO ICNT;
     233           DO K=1 TO NUM;
     234           IF AVAIL(I)=CODE(K)  THEN GO TO FND_KEY;
     235           END;
     236     FND_KEY:
                   SDIFF(I)=LOAD(K);
     237           END;
    STMT

         /****************************************/
         /*****  SORTING THE AVAIL ARRAY  *****/
         /****************************************/

     238        DO I=1 TO ICNT-1;
     239        DO J=I+1 TO ICNT;
     240        IF SDIFF(I) > SDIFF(J)  THEN GO TO SORTA;

     241        IF SDIFF(I) = SDIFF(J)  THEN DO;
     242           SIZE1=0;
     243           SIZE2=0;

     244           DO K=1 TO NUM;
     245           IF AVAIL(I)=CODE(K)  THEN GO TO S1;
     246           END;
     247     S1:   IF SIZE(K)='L'  THEN SIZE1=1;

     248           DO K=1 TO NUM;
     249           IF AVAIL(J)=CODE(K)  THEN GO TO S2;
     250           END;
     251     S2:   IF SIZE(K)='L'  THEN SIZE2=1;
     252           IF SIZE2 <= SIZE1  THEN GO TO SORTA;
     253           END;

     254        ITEMP=AVAIL(I);
     255        ITEMP2=SDIFF(I);
     256        AVAIL(I)=AVAIL(J);
     257        SDIFF(I)=SDIFF(J);
     258        AVAIL(J)=ITEMP;
     259        SDIFF(J)=ITEMP2;

     260   SORTA:
                END;
     261        END;
```

```
PL/I OPTIMIZING COMPILER         SCHED:   PROC REORDER;

    STMT


          /*****************************************************/
          /*****  SORT THE CHANGE STRUCTURE BY POSITION   *****/
          /*****************************************************/

     262       DO I=1 TO ICOUNT-1;
     263       DO J=I+1 TO ICOUNT;
     264       IF ABS(POS(I)-CPOS) <= ABS(POS(J)-CPOS)  THEN GO TO SORTB;

     265       ITEMP=ADD(I);
     266       ITEMP1=SUB(I);
     267       ITEMP2=POS(I);
     268       ADD(I)=ADD(J);
     269       SUB(I)=SUB(J);
     270       POS(I)=POS(J);
     271       ADD(J)=ITEMP;
     272       SUB(J)=ITEMP1;
     273       POS(J)=ITEMP2;

     274  SORTB:
               END;
     275       END;
```

```
STMT

      /*********************************************/
      /*****  FINAL MOLD SEQUENCING STATEGY  *****/
      /*********************************************/

      FINAL_SCHED:
              DO I=1 TO ICNT;
              DO J=1 TO ICOUNT;
              IF ADD(J)¬=' '  THEN GO TO NOMATCH1;
              MSIZE=0;
              NSIZE=0;

      /***  STRAIGHT SUBSTITUTION OF AVAILABLE MOLDS  ***/

              IF POS(J+1)¬=POS(J)+1 & POS(J+2)¬=POS(J)+2  THEN DO;
              IF J > 1 & POS(J-1)=POS(J)-1  THEN GO TO CHANGE_MOLD;
              IF J > 2 & POS(J-2)=POS(J)-2  THEN GO TO CHANGE_MOLD;
                 A1=LWDIFF(POS(J)-2)+LWDIFF(POS(J)-1);
                 A2=LWDIFF(POS(J)+2)+LWDIFF(POS(J)+1);
                 A3=LWDIFF(POS(J)-1)+LWDIFF(POS(J)+1);
                 ITEMP2=MAX(A1,A2,A3);
                 NEED(J)=LWD-ITEMP2;

                 DO K=-1 TO 1 BY 2;
                 DO M=1 TO NUM;
                 IF CLINE(POS(J)+K)= CODE(M)  THEN GO TO R3;
                 END;
        R3:      IF SIZE(M)='L'  THEN MSIZE=1;
                 IF TAPE(M)='Y'  THEN NSIZE=1;
                 END;

                 DO K=1 TO ICNT;
                 IF SDIFF(K) > NEED(J)  THEN GO TO R5;

                 DO MM=-1 TO 1 BY 2;
                 IF CLINE(POS(J)+MM)='AN' & AVAIL(K)='MM'  THEN GO TO R5;
                 IF CLINE(POS(J)+MM)='MM' & AVAIL(K)='AN'  THEN GO TO R5;
                 END;

                 DO M=1 TO NUM;
                 IF AVAIL(K)= CODE(M)  THEN GO TO R4;
                 END;
        R4:      IF SIZE(M)='L' & MSIZE=1  THEN GO TO R5;
                 IF TAPE(M)='Y' & NSIZE=1  THEN GO TO R5;

                 LWDIFF(POS(J))=LOAD(M);
                 ADD(J)=AVAIL(K);

                 DO L=K TO ICNT-1 WHILE(ICNT>1);
                 AVAIL(L)=AVAIL(L+1);
```

```
 STMT

  311              SDIFF(L)=SDIFF(L+1);
  312              END;
  313              ICNT=ICNT-1;
  314              GO TO FINAL_SCHED;
  315    R5:       END;
  316           END;

        /***  SUBSTITUTION WHERE MANY MOLDS ARE CHANGING  ***/

  317           ELSE DO;

  318     CHANGE_MOLD:
                   A1=LWDIFF(POS(J)-2)+LWDIFF(POS(J)-1);
  319              IF J > 2 & ADD(J-2)='  ' & POS(J-2)=POS(J)-2
                       THEN A1=LWDIFF(POS(J)-1);
  320              IF J > 1 & ADD(J-1)='  ' & POS(J-1)=POS(J)-1
                       THEN A1=LWDIFF(POS(J)-2);
  321              IF J > 2 & ADD(J-1)='  ' & ADD(J-2)='  ' &
                      POS(J-1)=POS(J)-1 & POS(J-2)=POS(J)-2  THEN A1=0;
  322              A2=LWDIFF(POS(J)+2)+LWDIFF(POS(J)+1);
  323              IF ADD(J+2)='  ' & POS(J+2)=POS(J)+2
                       THEN A2=LWDIFF(POS(J)+1);
  324              IF ADD(J+1)='  ' & POS(J+1)=POS(J)+1
                       THEN A2=LWDIFF(POS(J)+2);
  325              IF ADD(J+1)='  ' & ADD(J+2)='  ' &
                      POS(J+1)=POS(J)+1 & POS(J+2)=POS(J)+2  THEN A2=0;
  326              A3=LWDIFF(POS(J)-1)+LWDIFF(POS(J)+1);
  327              IF J > 1 & ADD(J-1)='  ' & POS(J-1)=POS(J)-1
                       THEN A3=LWDIFF(POS(J)+1);
  328              IF ADD(J+1)='  ' & POS(J+1)=POS(J)+1
                       THEN A3=LWDIFF(POS(J)-1);
  329              IF J > 1 & ADD(J-1)='  ' & POS(J-1)=POS(J)-1 &
                      ADD(J+1)='  ' & POS(J+1)=POS(J)+1  THEN A3=0;
  330              ITEMP2=MAX(A1,A2,A3);
  331              NEED(J)=LWD-ITEMP2;
  332              IF SDIFF(I) > NEED(J)  THEN GO TO NOMATCH1;

  333              DO MM=-1 TO 1 BY 2;
  334              IF CLINE(POS(J)+MM)='AN' & AVAIL(I)='MM'
                      THEN GO TO NOMATCH1;
  335              IF CLINE(POS(J)+MM)='MM' & AVAIL(I)='AN'
                      THEN GO TO NOMATCH1;
  336              END;

  337              DO K=-1 TO 1 BY 2;
  338              IF CLINE(POS(J)+K)=CLINE(POS(J+K)) & ADD(J+K)='  '
                      THEN GO TO R110;
  339              DO M=1 TO NUM;
  340              IF CLINE(POS(J)+K)= CODE(M)  THEN GO TO R11;
```

```
STMT


 341              END;
 342    R11:      IF SIZE(M)='L'  THEN MSIZE=1;
 343              IF TAPE(M)='Y'  THEN NSIZE=1;
 344    R110:     END;

 345              DO M=1 TO NUM;
 346              IF AVAIL(I)= CODE(M)  THEN GO TO R12;
 347              END;
 348    R12:      IF SIZE(M)='L' & MSIZE=1  THEN GO TO NOMATCH1;
 349              IF TAPE(M)='Y' & NSIZE=1  THEN GO TO NOMATCH1;

 350              LWDIFF(POS(J))=LOAD(M);
 351              ADD(J)=AVAIL(I);

 352              DO L=I TO ICNT-1 WHILE(ICNT>1);
 353              AVAIL(L)=AVAIL(L+1);
 354              SDIFF(L)=SDIFF(L+1);
 355              END;
 356              ICNT=ICNT-1;
 357              GO TO FINAL_SCHED;
 358           END;

 359   NOMATCH1:
           END;
 360   NEXT_SCHED1:
           END;

 361       PUT PAGE EDIT(' 4 ICNT= ',ICNT,'  AVAIL  SDIFF') (A,F(3),A);
 362       DO I=1 TO ICNT;
 363       PUT SKIP EDIT(AVAIL(I),SDIFF(I)) (X(5),A,X(3),F(3));
 364       END;

 365       PUT SKIP(2) EDIT('4 POS  ADD  SUB  NEED') (X(2),A);
 366       DO I=1 TO ICOUNT;
 367       PUT SKIP EDIT(POS(I),ADD(I),SUB(I),NEED(I))
              (X(5),F(3),X(5),A,X(5),A,X(5),F(3));
 368       END;

 369       IF (LWD<=29 & ICNT<=5) | (LWD<=30 & ICNT<=4) | ICNT<=3 THEN DO;
 370          DO I=1 TO ICNT;
 371          DO J=1 TO ICOUNT;
 372          IF ADD(J)¬=' '  THEN GO TO NX_LOOK;
 373          IF LWD < 34 & LWD+SDIFF(I)-NEED(J) > 34  THEN GO TO BEGIN_PGM;

 374          ADD(J)=AVAIL(I);
 375          GO TO NX_CNT;

 376       NX_LOOK:
              END;
```

```
  STMT


   377        NX_CNT:
                 END;
   378        ICNT=0;
   379        END;
```

```
    STMT


          /*****************************************/
          /*****  CHECK FOR PROBLEM SOLUTION  *****/
          /*****  THEN UPDATE MOLD POSITIONS  *****/
          /*****************************************/

     380        IF LSW=0  THEN DO;
     381           LSW=1;
     382           PUT PAGE;
     383           END;

     384        IF ICNT > 0 & LWD < 33  THEN DO;
     385           PUT SKIP(2) EDIT('  THE SCHEDULING PROBLEM CANNOT BE SOLVED ',
                      'WITH AN AVERAGE WIRE LOAD DIFFICULTY OF ',LWD) (A,A,F(3));
     386           LWD=LWD+1;
     387           PUT SKIP EDIT('  LWD IS NOW BEING SET TO ',LWD) (A,F(3));
     388           GO TO BEGIN_PGM;
     389           END;

     390        DO I=1 TO ICOUNT;
     391        CLINE(POS(I))=ADD(I);
     392        END;

     393        PUT SKIP(2) EDIT('  THE SCHEDULING & SEQUENCING ALGORITHM WAS ',
                   'SOLVED WITH AN AVERAGE WIRE LOAD DIFFICULTY OF ',LWD)
                   (A,A,F(3));
     394        LWD=27;
    STMT

              /*****************************************************/
              /*****  SORT THE CHANGE STRUCTURE BY POSITION  *****/
              /*****************************************************/

     395            DO I=1 TO ICOUNT-1;
     396            DO J=I+1 TO ICOUNT;
     397            IF POS(I) <= POS(J)  THEN GO TO SORTC;

     398            ITEMP=ADD(I);
     399            ITEMP1=SUB(I);
     400            ITEMP2=POS(I);
     401            ITEMP3=NEED(I);
     402            ADD(I)=ADD(J);
     403            SUB(I)=SUB(J);
     404            POS(I)=POS(J);
     405            NEED(I)=NEED(J);
     406            ADD(J)=ITEMP;
     407            SUB(J)=ITEMP1;
     408            POS(J)=ITEMP2;
     409            NEED(J)=ITEMP3;

     410      SORTC:
                    END;
     411            END;

     412      FINISH:
                    END SCHED;
```

```
STMT


        /**********************************************************************/
        /***  PRINT THE TOTAL NUMBER OF MOLDS ON LINE & MOLDS AVAILABLE   ***/
        /**********************************************************************/

  97          PUT SKIP EDIT('TOTAL NUMBER OF MOLDS') (X(5),A);
  98          IF ISIM > 0 & ICOUNT > 0 THEN PUT EDIT('MOLD','ADD','REMOVE')
                 (COL(36),A,2(X(10),A));
  99          IF LNUM=0  THEN PUT EDIT('TAPE AFTER POST CURE - @@') (COL(90),A);
 100          IF LNUM ¬= 0 THEN
                 PUT EDIT('PROBLEM AREAS') (COL(93),A);
 101          PUT SKIP EDIT('ON LINE') (X(11),A);
 102          IF ISIM > 0 & ICOUNT > 0 THEN PUT EDIT('POSITION','MOLD','MOLD')
                 (COL(34),A,X(8),A,X(10),A);
 103          IF LNUM=0  THEN PUT EDIT('PLATFORM WORK - $$') (COL(94),A);
 104          IF LNUM ¬= 0 THEN
                 PUT EDIT('POSITION','TOTAL WIRE DIFFICULTY')
                      (COL(84),A,X(5),A);

 105    NEXT:
              DO I=1 TO NUM;
 106          IF I>LNUM THEN GO TO NEXT1;
 107          PUT SKIP EDIT(CODE(I),EQ,MTOT(I),LPOS(I),LDIFF(I))
                 (X(11),A(2),A,F(3),X(67),F(4),X(13),F(4));
 108          IF ICOUNT >= I THEN PUT SKIP(0) EDIT(POS(I),ADD(I),SUB(I))
                 (COL(37),F(3),X(11),A(2),X(12),A(2));
 109          GO TO LIST;

 110    NEXT1:
              PUT SKIP EDIT(CODE(I),EQ,MTOT(I))
                 (X(11),A(2),A,F(3));
 111          IF ICOUNT >= I THEN PUT SKIP(0) EDIT(POS(I),ADD(I),SUB(I))
                 (COL(37),F(3),X(11),A(2),X(12),A(2));

 112    LIST:
              END;

 113          DO I=NUM+1 TO ICOUNT WHILE(ISIM >0);
 114          IF ICOUNT < NUM THEN GO TO FINISH;
 115          PUT EDIT(POS(I),ADD(I),SUB(I))
                 (COL(37),F(3),X(11),A(2),X(12),A(2));
 116          END;

 117    FINISH:
              ISIM=ISIM+1;
 118          END LINEUP;
```