Manufacturing costs can be materially reduced by a production control system which assures that all demands for materials are promptly met with a minimum amount of inventory on hand. Here is a closed-loop control system, based on a computerized inventory model, which has been successfully applied in several industries. Last of a series on the Manufacturing Information and Control System (MIACS) developed by General Electric and now being integrated into Honeywell's FACTOR System (after the recent merger of the two companies' computer businesses).

A Computerized Inventory Model For Production Control

C.A. RENOUARD, Honeywell Information Systems, Inc., Phoenix

The first article of this series (Sept. '70, p. 78) described MIACS as an integrated control system using three computer models to completely describe, monitor, and control the operations of a manufacturing or processing facility: a material inventory model, a process model, and a resource inventory model. The second article (Dec. '70, p. 60) described the Integrated Data Store (IDS) file management technique, which permits efficient maintenance of and access to the data required by MIACS modules.

This third and last part will develop the concepts of the materials inventory model, which would logically be the first of the three modules to be implemented in a projected complete system. Called GEIMS (GEneralized Inventory Management System), it can by itself make enough contribution to economic control of the enterprise to pay back quite promptly the investment in it.

The GEIMS model calculates the demand for the final products (or replacement parts) on a manufacturing enterprise and converts that demand into detailed schedules of item requirements throughout the plant or factory. These requirements, the production schedules based on them, and the stored inventory data are continuously modified on the basis of reports of actual plant activity fed back to the computer.

The system is designed to reduce production costs and plant investment by:

(1) Reducing the amount and value of inventory required to meet customer delivery criteria, while

(2) Reducing the lead times required between customer order and delivery,

(3) Eliminating equipment and personnel idle time due to lack of raw materials or in-process inventory,

(4) Reducing shrinkage by close control of inventory and in-process materials,

(5) Timely notification to management of impending delays and out-of-stock conditions, and

(6) Maintenance of product documentation and orderly control of effectivity ("used when" data) when products are modified.

The central element of the system is a model based primarily on two descriptors: the indented parts list for each product, and the lead time required to produce the subassemblies called out at each level of this list. (Other, detailed parameters are such factors as cost data for economic order quantity calculations.)

The organization of a parts list, and the file structure used in MIACS to store such lists efficiently in a computer, were described in some detail in the second article of this series. Briefly, a parts list is a record of the structure of every item—subassembly and final product—inventoried within the enterprise (or the portion of it covered by the model). It has the general form of Figure 1, in which the parts list for item A shows that A consists of 2 of item B, 3 of item C, and 12 of item D. A separate callout chain shows that B, in turn, is made up of items E, F, and G in varying quantities. Links established in the Integrated Data Store create in effect an indented parts list showing the components of every item down to the lowest assembly and piece-part level, and listing all assemblies on which each part and subassembly is used.

Simplified model

The inventory model for a typical product is shown in

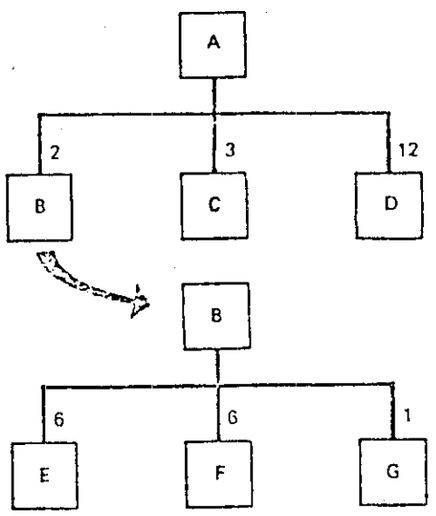


FIG. 1. Typical product structure: Finished product A consists of 2 of item B, 3 of item C, and 12 of item D. Subassembly B, in turn, is made up of items E, F, and G, in quantities of 6, 6, and 1, respectively.

Figure 2, where the simplifying assumption is that the item is not normally inventoried but is manufactured to order. (This assumption will be modified later.) The input to the model, a schedule of demand vs future date, is combined with the lead time required to assemble the final product using the first (highest) level of subassemblies. This generates a schedule of quantities vs time for the start of final assembly.

The latter schedule in turn is applied to the parts list, to generate a schedule of requirements of first-level subassemblies. This is applied to the lead times required to fabricate each first-level subassembly, which generates the starting schedules for those fabrications and the due dates for the next lower-level subassemblies.

This iteration is continued backward in time and downward in level of assembly until a schedule for procuring purchased materials and starting the lowest level of assembly or fabrication is generated. This process is referred to as the "explosion" of a demand into its many detailed intermediate demands.

As engineering changes are made, they are incorporated into the parts list by substituting new detail records at the appropriate levels. If required, both the old and the new parts can remain in the chain, with effectivity notes (either by part number or by effective date) for both. The computer will take effectivity into account when exploding the demand for a product.

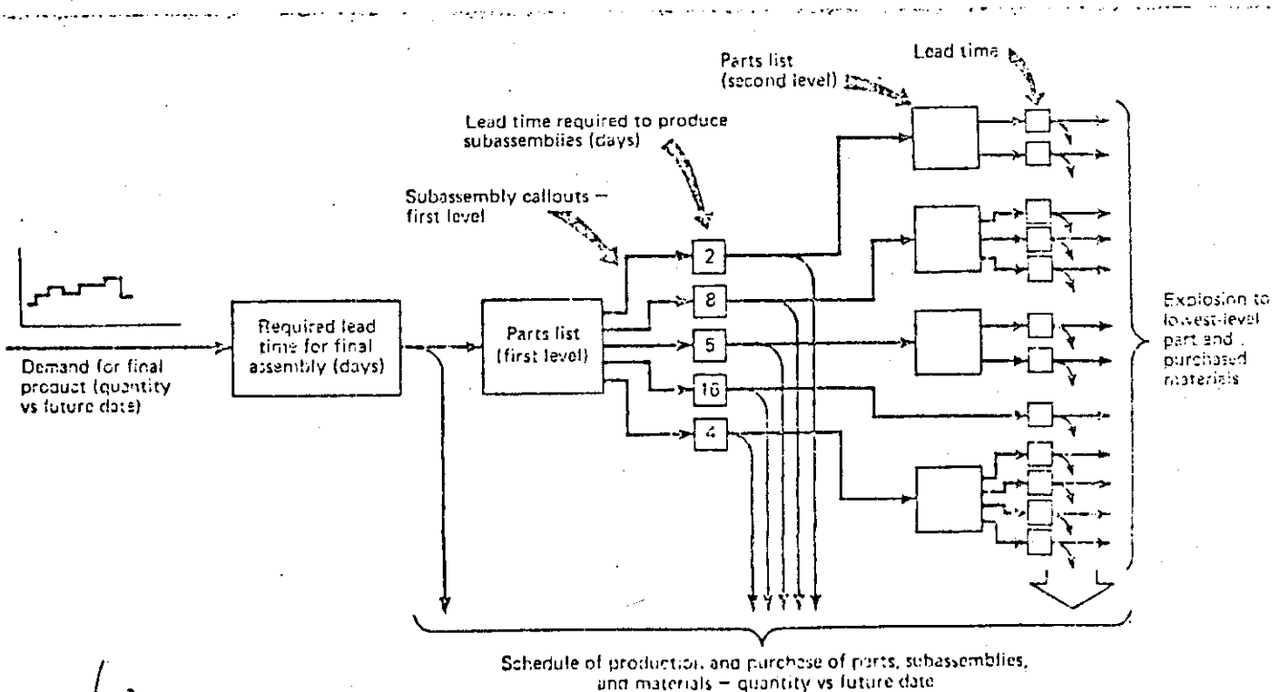
Supply orders

In the case of items which are inventoried, an additional step is introduced into this process: Demand is converted into "supply orders," as illustrated in Figure 3.

Here, "demand" is defined as the requirements (estimated or specifically ordered) from inventory for delivery or as input to production. "Supply" is defined as the amount actually produced by the plant in response to a supply order. Supply is a function of both demand and inventory on hand, as generated by an ordering rule.

This rule is calculated in terms of the economic goals of the enterprise, seeking the optimum tradeoff

FIG. 2. Inventory model generates exploded demand by (at each subassembly level) identifying the component parts from the stored parts list and offsetting the supply orders for each by the time required to make or procure the item.



between costs of manufacture in various lot sizes and the cost of ownership of inventory. Ordering rules may be stored in the inventory system in the simple form shown in Figure 3, in which case the only required inputs are demand and inventory status. Additionally, if the costs of manufacturing and of holding inventory are expected to vary significantly, the system can recalculate the ordering rules from the economic data as new costs are entered into the system.

The application of the ordering rule of Figure 3 for item B is indicated by Figure 4. Assume that two demands for product A are entered on January 10: a requirement for 50 on April 5 and for 30 on April 12. Assume that product A is not inventoried but is assembled in each order quantity for direct delivery to the customer. If final assembly of A requires 15 days, the ordering rule might call for the start of final assembly of each lot 15 days before the respective delivery dates. (The amount of lead time specified by the rule can be made to vary as a function of lot size, if required.)

All first-level parts and subassemblies must be available in inventory on the date final assembly is to start. This creates demands for parts B, C, and D on March 22 and 29. The demand for B is shown in Figure 4. As shown, the future inventory of B is calculated to be reduced below the reorder point by the demand of March 29. This finding triggers the calculation of a supply order for B, of the size and starting date called for in the ordering rule of Figure 3. Specifically, in the case illustrated, the calculation causes a lot of 560 of item B to be started on March 13, to meet a need for 60 on March 29 and to replenish inventory.

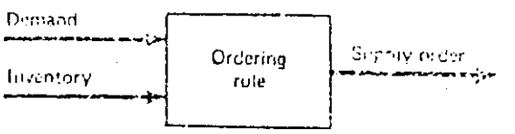
Note that supply orders will not automatically be generated for every item on the parts list of item A—no orders will be generated for items of which there is an adequate inventory to meet the projected demand.

And note also that the ordering rule calls for the consolidation of all demand over a 60-day period in calculating the supply order. The decision to consolidate, and the length of the consolidation period, would normally be based on the economics of lot size and cost of ownership.

Similarly, the ordering rule can incorporate provision for revising the order quantity up or down in response to new demands or cancellations after the order has been generated. Both the magnitude of the revision and the maximum allowable time between the initial order and the revision can be specified in the ordering rule.

Orders to the shop

The flow of information is illustrated in Figure 5. The plant or facility under control is divided into "administrative centers"—logical or functional groupings of resources (perhaps manufacturing departments such as a foundry or machine shop, or in some cases a single machine, such as an injection molding unit)—each assigned a number or code des-

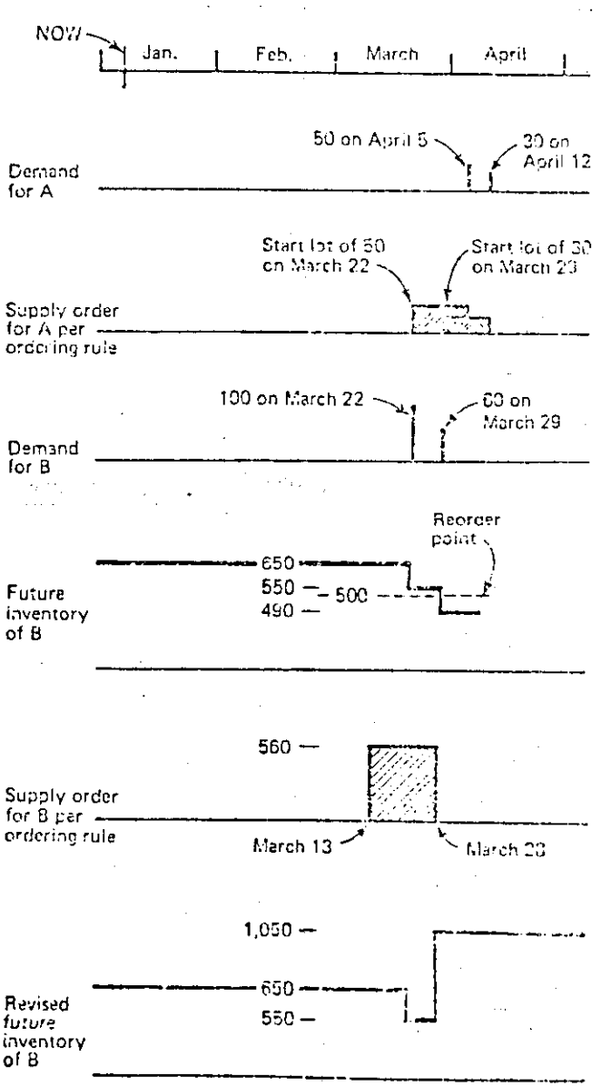


Ordering Rule for Item B

1. Reorder point: 500 items.
2. Due date of order: date inventory on hand is calculated to drop below reorder point, minus one.
3. Calculation of lot size: sum of demand for 30 days following due date, plus 500 items, and minus any items being made for inventory under existing orders in process. Minimum lot size: 500 items.
4. Starting date of work order: due date minus 15 days.

FIG. 3. Supply is a function of demand and inventory, as expressed by an ordering rule. Here is a hypothetical rule for item B of the example. If desired, the parameters determining lot size (step 3) can be recalculated within the model as costs undergo change.

FIG. 4. Demand for final product A generates corresponding demand for subassembly B, earlier in time. The model finds that the second demand will pull future inventory below the reorder point, so the ordering rule generates an order to start production of 560 of B on March 13, to be delivered to inventory on March 28.



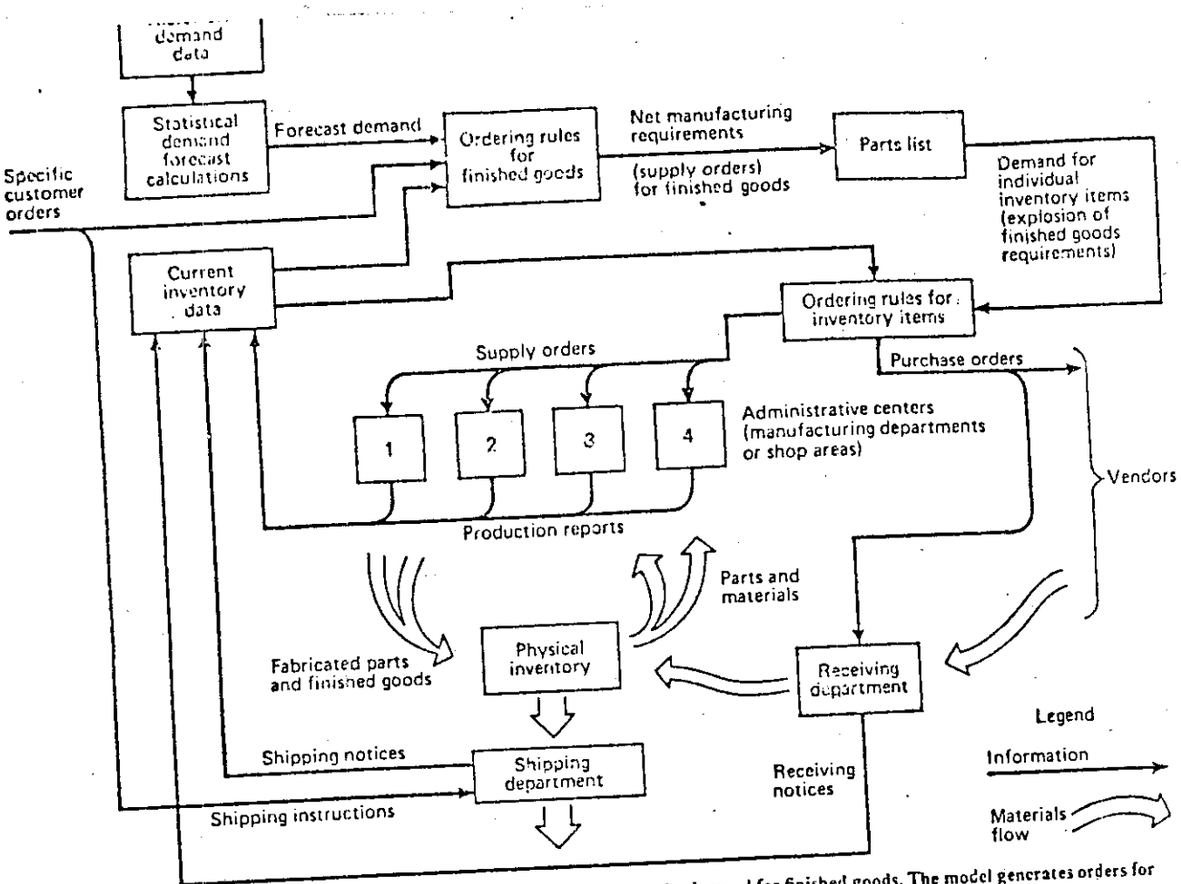


FIG. 5. Flow of information through the inventory control system starts with the demand for finished goods. The model generates orders for the continuous replenishment of inventory. Detail records within the inventory file enable the computer to direct the work orders to the administrative areas responsible for each task. Feedback from production, receiving, and shipping enables the computer to keep track of the inventory quantities used in the model.

ignation within the inventory system.

Using the IDS file structure, it is easy to link to the master for each inventory item the identification of the administrative center or centers responsible for making or procuring the item. Thus as the computer generates supply orders it can direct these orders to the appropriate administrative centers. This means that the entire plant is loaded in an economical and orderly way, with one center producing today the in-process inventory which another center will require tomorrow, without the development either of shortages or of excessive inventory.

As work progresses, actual plant performance is fed back to the computer: Production personnel report on supply orders completed, the receiving department enters arriving goods into the system, and the shipping department reports on goods dispatched out of the facility. This input may be collected manually and entered periodically as a batch, if the computer is a batch system or, if it is an on-line system with remote terminals in the production, receiving, and shipping areas, each transaction can be entered as it occurs.

This feedback makes it simple for the computer program to detect any emerging discrepancies between schedule and performance, and to generate timely expedite reports to management of impending delays or shortages.

The model can be made to accept inquiries by management or production supervisors and to provide the supply status of any inventory item or the shipping status of any customer order.

Users' experience

The inventory model has been implemented by several industrial users and is producing the expected economic gains. One interesting example is General Electric's Power Production and Conversion Div., Philadelphia, a manufacturer of power generation and distribution equipment. The division has implemented these inventory capabilities on a Honeywell 600 computer, with direct communication to a plant in Burlington, Iowa, via a satellite G-100 computer. Over a two-year period in which plant size doubled, customer delivery cycle has been brought down from 28-32 weeks to 10-12 weeks. The time required for a cycle of product design and manufacturing startup by the engineering department has been reduced by 75 percent—an increase, according to the user's calculations, of 60 percent in engineering productivity. □

United States Patent [19]

Gelder

[11] **3,725,650**

[45] **Apr. 3, 1973**

[54] **METHOD AND ARRANGEMENT FOR OPTICALLY REPRESENTING INDUSTRIAL MANAGEMENT DATA**

[76] Inventor: **Franz Gelder,** Gabelsberger Str. 36, Salzburg, Austria

[22] Filed: **Sept. 15, 1970**

[21] Appl. No.: **72,332**

[30] **Foreign Application Priority Data**

Sept. 15, 1969 Austria 8735

[52] **U.S. Cl.** **235/151.3**

[51] **Int. Cl.** **G06f 15/20**

[58] **Field of Search**.....340/324, 166 EL; 307/311; 313/108; 250/217; 324/114; 235/151.3

[56] **References Cited**

UNITED STATES PATENTS

3,327,163	6/1967	Blank	340/166 EL X
3,205,403	9/1965	Schwertz	313/108 X
3,531,795	9/1970	Gassler	340/324
3,376,452	4/1968	Lally	313/108
3,328,790	6/1967	Rhodes	313/108
3,469,252	9/1969	Bet	340/324
3,149,281	9/1964	Lieb	324/96
3,351,937	11/1967	Spens	313/108
3,565,606	2/1971	Carlson et al.	235/151.1 X
3,475,599	10/1969	Schwartzenberg et al.	235/151.35

Primary Examiner—Malcolm A. Morrison
Assistant Examiner—Edward J. Wise
Attorney—Edwin E. Greigg

[57] **ABSTRACT**

Data obtained in industrial production or office work, such as rate or quantity of output of a machine or a worker, are applied to computers for deriving electric signals characterizing the operation with regard to efficiency, profitableness, etc. Indicator boards are provided having electroluminescent strips or discs, or cathode ray tubes the extent of the luminous portions of which are controlled by the computers.

25 Claims, 8 Drawing Figures

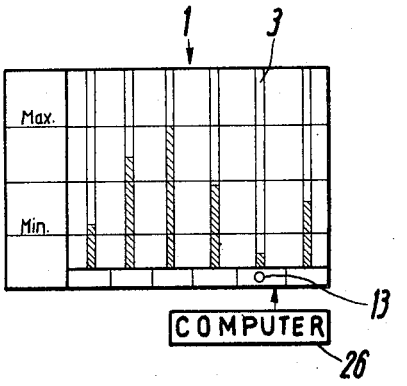


Fig. 1

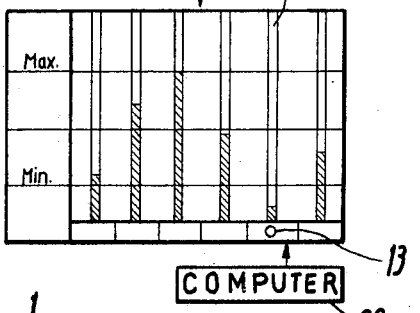


Fig. 2

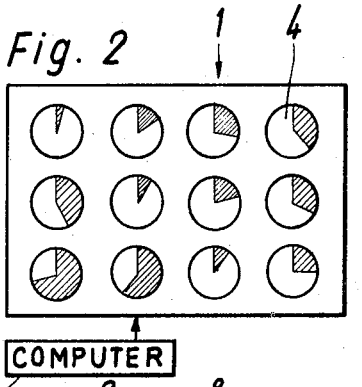


Fig. 3

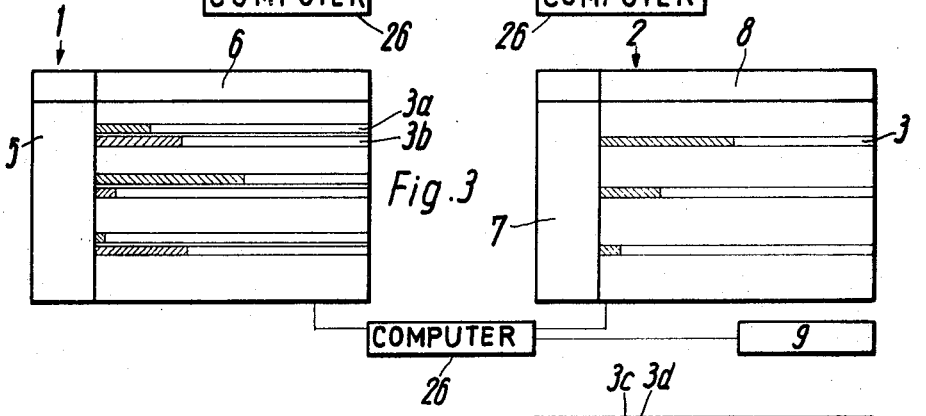


Fig. 4

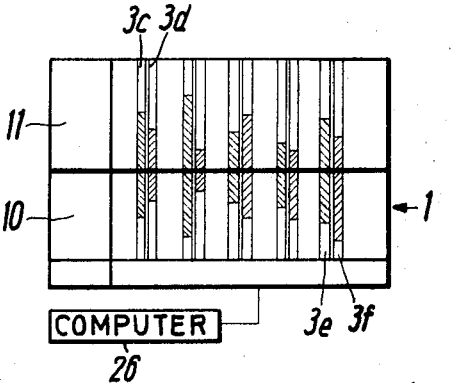
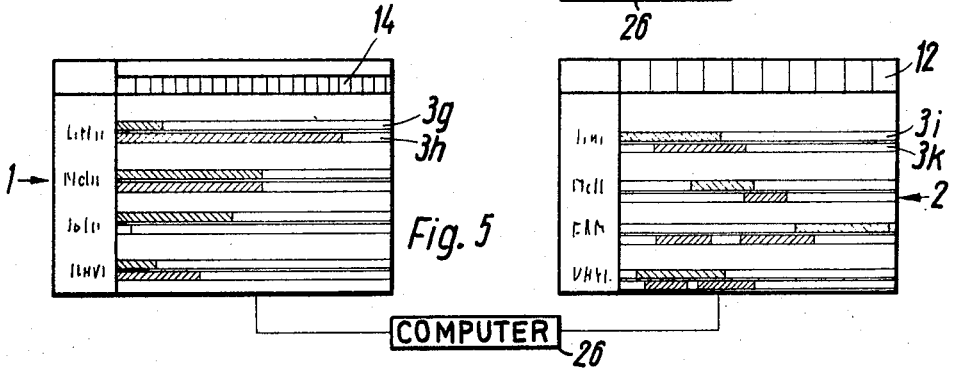


Fig. 5



Inventor:
Franz Gelder
by:
Edwin E. Greigg

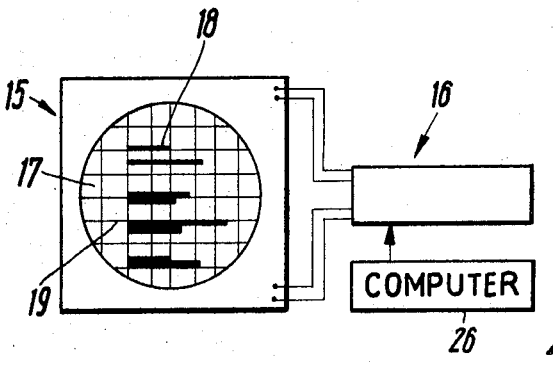


Fig. 6

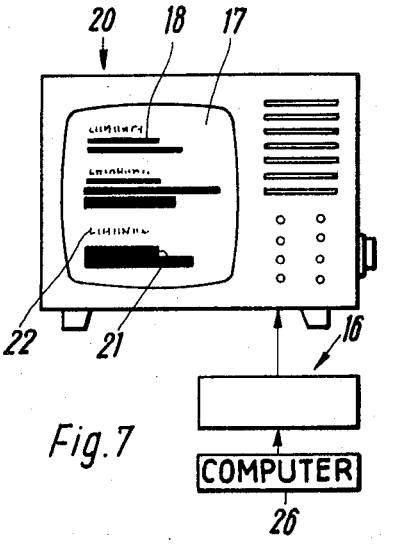


Fig. 7

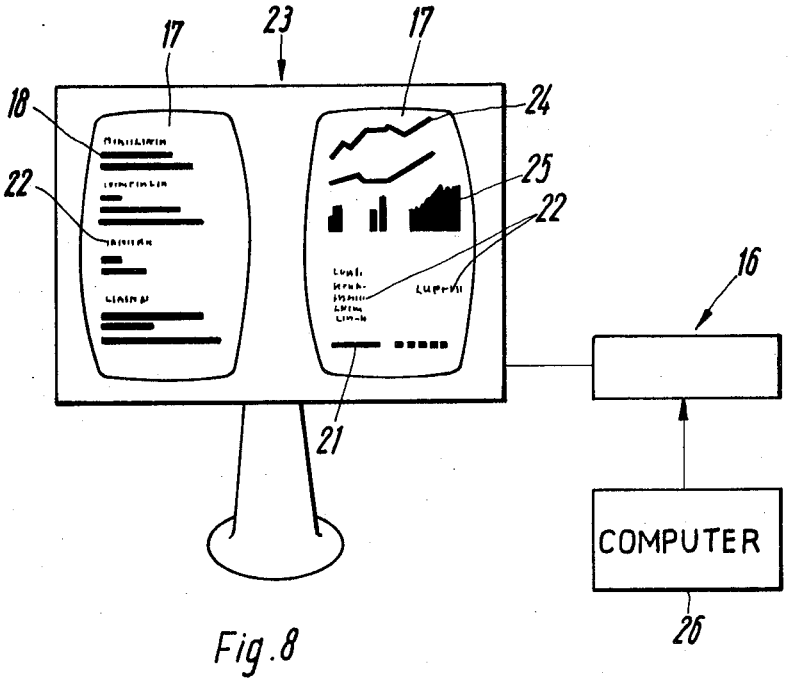


Fig. 8

METHOD AND ARRANGEMENT FOR OPTICALLY REPRESENTING INDUSTRIAL MANAGEMENT DATA

FIELD OF THE INVENTION

The invention relates to a method for optically representing data characterizing a production process and to an arrangement for performing the method.

BACKGROUND OF THE INVENTION

In offices as well as in industrial enterprises, factories, etc., it is imperative to have access to a number of production data, such as the quantities of planned, actually produced and sold products, prices, prime costs, number of work hours, and the like. These values are collected for storage and for further processing. In the interest of clarity, use is made for this purpose of graphic diagrams, tables, charts and the like.

It is previously known to utilize indicator boards, on which are placed strips of paper, which may be in different colors, or magnetic markers. Also known are boards provided with colored tapes which can be pulled off from reels. These methods of representing production data are complex and time-consuming and are not suitable for automatic processing, for instance, in computers.

OBJECTS AND SUMMARY OF THE INVENTION

It is an object of the invention to provide an arrangement and a method for handling industrial or other data, which makes it possible to represent them optically in a rapid and dependable manner with as little use of personnel and time as possible, and to facilitate storage and other processing of such data with the aid of a computer.

According to the invention, data characterizing a production process such as an industrial process or other operation or a quantity which has been derived from such data with the aid of calculators, preferably an electronically controlled calculator, such as a computer, is obtained and transformed into electrical signals. The signals are applied to a digitally controllable optical display unit such as an electroluminescent device, and cause the same to respond to and indicate the magnitudes of the quantities. This method can be carried out very simply and easily with the aid of a digitally controllable optical display unit comprising an electroluminescent device having at least one element which is responsive to produce an electroluminescent output signal of variable magnitude and which responds directly or indirectly to a digital signal-generating means, each element being provided with a scale for the corresponding quantity.

In a modified embodiment of the invention, the digitally controlled optical display unit is a luminescent screen device, such as a conventional TV image tube, the input circuit of which is connected, preferably via an encoder, to the calculator, the control circuits of the tube being arranged to supply electrical signals for displaying the magnitude of a quantity representing a characteristic of an industrial process or an optical signal corresponding to such magnitude.

An advantage of the use of electroluminescent strips consists in that they can be controlled digitally from a computer. Also, adjacent strips produce indications that are easily compared, and the range of electroluminescent response is capable of relatively fine subdivision.

It is also possible to use as indicators counter-tubes or cathode ray oscillographs, as well as combinations thereof. Since it is generally preferred in office equipment to use analog representation, the invention in its preferred form uses devices including electroluminescent strips.

The representation may also be in the form of strips on a luminescent screen, i.e. on a TV image-reproducing tube. This provides, in addition, a possibility of recording and displaying diagrams, charts, symbols and characters and to illustrate the time variations of processes.

BRIEF DESCRIPTION OF THE DRAWING

The invention will be described below with reference to the drawing, which shows schematically by way of example various embodiments thereof.

FIG. **1** shows an arrangement for representing productivity,

FIG. **2** is a modification thereof,

FIG. **3** shows an arrangement adapted for production planning,

FIG. **4** shows an arrangement for representing production data in connection with cost calculation and marketing,

FIG. **5** is an arrangement adapted for capacity planning,

FIG. **6** is an arrangement comprising an oscillograph,

FIG. **7** is a modified form including a TV apparatus, and

FIG. **8** is an arrangement with two image screens.

DESCRIPTION OF EMBODIMENTS

FIG. **1** shows schematically an arrangement for representing the characteristic quantity "productivity" of an industrial production process. The productivity is a characteristic data or quantity which is derived from other data such as the types and quantities of products as well as the number of working hours used for their production, and it is usually indicated as a percentage. The arrangement comprises an indicator board which may have a number of electroluminescent devices, or cathode ray tubes, such as conventional TV tubes thereon. In the embodiment shown, there are provided a number of electroluminescent strips **3**. The luminescent strips may be of the type referred to in the magazine "Flugwelt" **19** (1967), No. 9, page 638, published by Krausskopf-Flugwelt-Verlang GmbH., Mainz, Germany, or in the magazine "Luftfahrttechnik-Raumfahrttechnik", Vol. 13 (1967), No. 11, pp. 276–279, published by VDI Verlag, Dusseldorf, Germany. The strips are responsive to a calculator or computer **26**, preferably of the electronic type, and the output signals of which are applied as input signals to the strips. Provided at the lower edge of the board are markings indicating fields or categories of production, such as departments of an enterprise, premium or bonus groups, accounting information, e.g. accounting sections, etc. for the various strips **3**. There are supplied manually or automatically to the calculator or computer a number of data and the computer calculates therefrom the corresponding maximum,

minimum, or average productivity values over a predetermined period of time and indicates them on the board. A scale may be provided on the board for enabling reading of numerical values therefrom. Optical alarm signals **13** are provided, which are activated when a quantity decreases below a predetermined value, such as an average or a minimum. The instantaneous values may be correlated with values stored in the calculator and derived from indications of the indicating device during the foregoing comparison intervals. In addition to the optical representation of the production data, it is also possible to provide digital representation, for instance, by means of digital counting tubes.

FIG. **2** shows a modification of the arrangement, in which the indicating devices comprise disc-type elements. According to the magnitude of the quantity to be represented, a greater or lesser sector is rendered luminescent in response to the output signal of the computer.

FIG. **3** shows the combination of two arrangements adapted for production planning. An indicator board **1** is provided with a number of pairs of strips **3***a*, **3***b* and with a pair of marginal fields **5**, **6**. Field **5** contains a tabulation of the machines used in the company or plant and field **6** contains a time scale. Strip **3***b* may show the total running time of a machine, whereas strip **3***a* shows the time during which the machine was loaded or not idling. A second board **2** with electroluminescent strips **3** is located adjacent to the first one and is provided in a marginal field **7** with designations of the products produced by the machines and in a second marginal field **8** also with a time scale. The indicating devices of both boards are controlled by a computer **26** to which there may be applied input signals indicative of the quantities of different products via a digital signal generator **9** in combination with an encoder.

FIG. **4** shows an arrangement serving to facilitate cost calculation or analysis and to make it possible to perform such operations more rapidly than was heretofore possible, and to facilitate taking steps to modify marketing procedure. An indicator board **1** is divided into two regions **10** and **11** having corresponding scales for indicating production volume, turnover, sales or the like and for indicating marginal cost **11**. The electroluminescent strips **3** are continuous but preferably separate for the two fields and are paired together, each lower pair **3***e*, **3***f* being connected to the output side of the corresponding upper pair **3***c*, **3***d*. The lower scale may be provided with symbols designating representatives' or agents' districts corresponding to the lower pairs. The strips serve to indicate desired (nominal or theoretical) values as well as actual values of production volume, turnover etc., **3***c*, **3***d*, as well as of the marginal cost **3***e*, **3***f* and are controlled by a computer. It is obvious that a greater number than two of coordinated strips could be provided, which might differ from each other in width and/or color or the like to facilitate distinguishing them from each other.

FIG. **5** shows another combination of two arrangements which facilitates the capacity planning of an enterprise in a very much simplified and rationalized manner. In this case, also, a pair of indicator boards **1**, **2** are provided, one of which has a plurality of pairs of strips **3***g*, **3***h*, as well as a time scale **14**. The pairs of strips correspond to locations or operating units or the like, and the strips themselves indicate the desired (nominal or theoretical) and the actual value of the degree of utilization, such as the number of persons or machines assigned to a work, rate of output, or the like of the operating equipment or personnel of the locations or units. The strips are preferably controlled by a computer **26**, as are also the strips **3***i* and **3***k* of the second board, there being applied to computer **26** the required input values, which are processed therein and transformed into output signals. The second board **2** is also provided with a time scale **12**, which may indicate chronologically dates and/or weekdays and/or months. Provided in one field of board **2** are designations of various products corresponding to strips **3***i*, **3***k*, the strips showing the desired (nominal) and the actual amount of time spent in production. The embodiment shown possesses only pairs of strips **3***i* and **3***k*. However, it is just as possible, if several shifts are operating, to provide a corresponding number of strips for each product designation as well as an arrangement for indicating the desired (nominal) and the actual value for each shift. The strips may be of different construction, such as of different widths, colors or the like. For instance, all strips indicating actual values may be of the same type. Preferably, there is integrally combined with each indicator board a decoder for transforming the output signals of the computer, which may be in machine code, into a form suitable for application to the electroluminescent strips. Owing to the storage capability of a computer, it is also possible to represent on the boards values corresponding to arbitrary past periods in time. Furthermore, it is possible to monitor a continuously running machine, if the quantity to be measured is derived immediately from the machine itself. The arrangements described are therefore suitable for obtaining all sorts of values or quantities that may be of interest in connection with planning or monitoring the production of a plant or an office and for processing and indicating the same as rapidly as possible, there being no limits in this regard other than those depending on the size and storage capacity of the computer.

FIG. **6** shows an arrangement in which the display unit for optically displaying data or quantities representing characteristic features of a production process comprises an oscillograph **15** and a computer **26** connected thereto. The connection may be direct, but in the embodiment shown there is inserted between the computer and oscillograph **15** an encoder **16**. The computer has stored therein all data that may be of interest, such as production and other quantities and amounts, work hours, machine or equipment capacities, prices, prime costs and the like together with corresponding programs and instructions which may represent the organization of a production process, time planning, pre-planning and so on. To represent or display the data of interest on the image screen **17** of the oscillograph, they are processed in the computer and transformed into electric signals, such as pulse sequences, which are supplied to the control circuits of the oscillograph. However, in the preferred embodiment shown, the computer supplies its output to an encoder **16**, which performs a further processing and sup-

plies its output signal to the control circuit of the image screen device. Decoder **16** is preferably provided with a repeating device, such as an endless magnetic tape storage device, to make it possible to repeat indefinitely the input signals supplied to the control circuits. The electric signals which may be pulse sequences, are rendered visible on image screen **17** as optical signals, which may take the form of lines **18** of a length corresponding to the measure of the corresponding data or quantity to be represented. Depending on the optical inertia of the image screen, several lines can be displayed simultaneously to the human eye. Since most oscillograph screens are provided with a raster pattern **19**, this can be used as a reference scale for measuring the displayed quantities.

FIG. **7** shows a modified arrangement, in which a T.V. apparatus **20** is used as an image screen device, on the screen **17** of which a measure of the data or quantity characterizing an industrial or other process is displayed in the form of optical signals such as lines **18**, beams **21** or the like. It is also possible to represent a quantity as a number or numeral **22** or to bring a corresponding legend to luminescence. The computer or the encoder **16**, which is preferably connected thereto may in this case also be provided with a repeating unit and gives off electrical signals, such as pulse sequences, forming a video signal which is applied to the input control circuit of the apparatus. The pulse sequence is put together by the computer in accordance with the desired type of T.V. representation. In most cases, it will be sufficient to represent the optical signals as corresponding points of the image screen, which are brought to luminescence, and to leave all the remaining points dark. The reverse process, where the optical signals are represented as dark points on an otherwise luminous screen, would also be possible.

If the T.V. apparatus is capable of reproducing colors, data or quantities of particular interest, such as monthly balances, total productivity figures, optimal prices or the like could be made to stand out by means of differently colored optical signals. If no composite colors are used, this, of course, simplifies the control process of the apparatus.

It is also possible to arrange for predetermined sections of the screen to emit optical signals of a corresponding color. The incoming electrical signal may be automatically supplied to the electron gun corresponding to a preselected color to be used and to activate the gun to produce the optical signal when the corresponding image point falls in the desired section of the screen, such as the last hundred lines thereof. In a simplified modification, the tube may be provided with a screen having at least one section thereof provided with different phosphors for emitting different colors.

If production data are to be displayed in correlation with months or years, the optical signals corresponding to months could be displayed on a section of the screen which has a phosphor of conventional TV type and makes the corresponding signals appear in white or slightly bluish color, whereas the data corresponding to years are represented by points, for instance, in the section comprising the last **50** lines, which may comprise a phosphor of the type used in oscillographs, so as to make these image points appear green.

In any case, there is obtained a clear and easily evaluated representation of a number of quantities or corresponding optical displays, which are distinguished by different colors and are therefore easily grouped together.

However, the representation of data is not limited to the use of lines, beams, numbers or letters, as will be explained in the following.

FIG. **8** shows an embodiment having two image screens **17** and which is particularly suitable for conference purposes. The computer **26** and/or encoder **16** applies to the control circuits of an image screen device **23**, electric signals such as pulse sequences, in the manner already described. The representation of data may be in the form of lines **18**, beams **21**, numbers, numerals, letters or symbols **22**. However, it is also possible to represent the time variation of data as a linear graph **24** or a shaded area **25**. If the image device is adapted for reproducing different colors, displays of particular interest can be either repeated once more or be represented separately on the second screen. Both screens can be of the type used in color television receivers. The two screens can be adapted to reproduce different unicolored optical signals or the screens may be of the type described in connection with FIG. **7**, or a combination of both is possible.

The arrangement, therefore, makes it possible to obtain a modern, easily interpreted display of data representing production or other processes to the exclusion of human error. Owing to the great storage capacity and calculating speed of the computer, it is possible to represent to the viewer a large number of data in a minimum of time. These possibilities will be illustrated by the following.

The production planning of an enterprise gives rise to a problem of the following type:

The enterprise produces n products $X_1, X_2 \ldots X_n$ with the aid of m production facilities $V_1, V_2 \ldots V_m$. The term "production facilities" is to be taken in its broadest sense and includes machines, personnel, raw materials, intermediate products, electric energy, storage space, transport vehicles, etc.

Normally, the most important facilities are machines and workers, which may form organized production units. The capacity of such a unit is the maximum amount produced in a certain interval.

Usually, it is desired to maximize the overall profit of the enterprise, which is a function of the quantities and types of products. For different products, the price includes different values of the components which together constitute the price quotation: variable cost, fixed cost and profit. The first two together form the prime cost and the last two the marginal cost.

Variable cost includes, for example, the cost of electric energy, raw materials, proportionate wages, etc. Fixed cost, which includes, for example, depreciation of equipment, rents, insurances, fixed wages, etc., remains unchanged regardless of whether equipment or labor is used productively or not.

As a simple example, assume that two products X_1 and X_2 are manufactured in quantities (so far unknown) of x and y and the corresponding profits per unit are a and b, respectively. The total profit is then expressed by the linear equation $Z = ax + by$; since the production capacity is limited, both x and y are limited

and if further factors are taken into account, such as the fixed and variable costs of the products, marketing considerations, etc., additional conditions are to be met and new variables are introduced, such as the degree of utilization of different facilities. These additional conditions may also be expressed by equations, e.g. of linear character.

The totality of these equations constitutes an optimum model, especially a maximum profit model or minimum cost model indicating how the production should be planned. It is possible to arrive at values for the several variables by solving the usually very large number of equations of the optimum model obtained by known linear programming methods, preferably by means of a computer, to which there is fed the mathematical program for solving the system of equations as well as the input data representing the known properties of the production, such as capacities, profits of products, fixed costs, marketing considerations, etc.

The methods of obtaining the optimum models and solving the same by means of computers may be, for instance, of the types described in one of the following well-known textbooks:

Dorfman-Samuelson-Solow: *Linear Programming and Economic Analysis*, New York, 1958

Ferguson-Sargent: *Linear Programming*, New York, 1958

Manne-Markowitz: *Studies in Process Analysis*, New York, 1963

The theoretical value of the quantity of a product which has been obtained by the computer may now be multiplied with the corresponding marginal cost. This operation can be performed by the computer, which has already been supplied with information about the marginal costs of different products. The resulting value may be represented on a luminescent strip.

The marginal cost value supplied to the computer is, on the one hand, the so-called planned marginal cost obtained from the planned cost accounting or from a roughly estimated production program and, on the other hand, the actual marginal cost which is obtained as the difference between the effective net product sales and the product-variable costs. This is done since the quotation prices of the product arrived at by the cost accounting (with the product costs calculated by the cost accounting) usually do not correspond to the established actual product market prices which may differ also because of competitive grounds. Since, as a general rule, the product line formed of a number of different products is not permanent but, because of numerous influencing factors may be, and rationally should be, varied, it is advantageous — as contemplated by the invention — to indicate the actual marginal cost per product as well as its planned marginal cost not only as a mathematical but also as an optical (i.e. visual) magnitude. If the actual marginal costs are shown visually to be larger than the planned marginal cost of the products, an immediate overall visual survey meaningfully indicates not only the particularly profitable products, but also those products which, as a general rule, may be sold cheaper by the difference between the actual marginal cost and the planned marginal cost or those which, with their already well-established prices, particularly contribute to the profit buildup of the enterprise and therefore their production and sale should be encouraged. The same, but converse considerations apply for those products whose actual marginal cost is smaller than their planned marginal cost; a condition which is also visually recognizable according to the invention. These indications, which may thus also be represented visually, have a great significance for the determination of the product line and for market and price policy considerations. This significance of such indications is even more intensified by the fact that — based on the marginal cost values per product supplied to the computer and based on the work plan per product and further, based on data which are also fed to the computer and which relate to available production capacities pertaining to equipment and labor per work station — according to the invention, it may also be visually shown next to the visual indication of the production capacity, whether and to what measure does in a planned production program the production capacity fulfill the requirements in its entirety and also per operating work stations for both a single shift and a multiple shift operation (it is noted that a work plan discloses which work station is used to make the product and what average production times are needed for equipment and labor). In this manner, again, important indications are obtained for further investment policies since the operational bottlenecks are immediately visually recognizable in a positive manner. Since on the opposite side of the exemplarily described arrangement of the invention, not only the planned and actual marginal cost per product (and thus its meaningful difference) may be recognized, but there are also visually shown the absolute magnitudes of each planned product quantity or actual marginal costs derived from product quantities calculated and proposed by the computer based on a given optimal model. Such indications may furnish the management with further important keys for decision making. Since previously the computer also calculated from the given program the entire sum of the fixed costs relating to the operation (that is, the so-called actual total fixed cost), there is immediately shown not only digitally but also in a meaningful manner visually, with what type of product and product quantities (and the marginal costs resulting therefrom) will the actually present total fixed costs be fully covered. As a further result, the apparatus described by way of example, visually and digitally shows to what extent does the total profit of the enterprise increase if the available production capacities are fully balanced. Such result is obtained principally because, subsequent to covering the total actual fixed cost, the excess of the marginal cost values of the products gives the actual profit of the enterprise for the planned or computer-calculated optimal production program. These data are also optically and digitally visible. Thus, as a result, the invention assists in a particular manner the product line planning and production planning for a maximum profit.

The aforenoted considerations thus show the importance of a rapid indication of the actual and planned marginal cost as well as the magnitude obtained by multiplying the product quantity by its marginal cost. It is similarly important to show rapidly the utilization of operational capacity of the production program which is either planned in a conventional manner or is calculated by a computer based on an optimum model.

If the computer, based on the optimum model, calculates the quantities of the individual products, then, since the manufacturing steps of the individual products and the production factors necessary therefor are fixed, the utilization of the production capacity is determined. The speed of calculation and also a rapid visual indication of the results is of great importance since it is then possible to obtain various simulated programs which are close to reality. In practice, the determination of the production program is never left solely to the computer, not even if it works with mathematical optimum models. The reason is that there are constantly changes in the magnitudes of the influencing variables so that only a cooperation of the human spirit (usually taking effect as a result of teamwork of management), sales and operational leadership, cost accounting and further operational units, may result in a determination of the optimum product line and production program by utilizing the calculating speed and capacity of the computer and further utilizing suitable rapid visual indicator devices to illustrate the results automatically which is the purpose and the subject of the present invention.

In practice, there are thus preponderantly variable simulated situations which are "played through" mostly as a result of the teamwork of the departments in charge around the conference table utilizing a continuously changing quantity of operational data and accordingly, supplying different data quantities to the computer (which may be located on or off the premises). Thus, continuously changing products are often preplanned for production in determined quantities, because the sales department may have already fixed orders which have to be fulfilled in any case irrespective of whether these products are particularly profitable or not. Since such fixed planned product species and quantities (which may be determined for example in the course of a conference) are supplied as a so-called condition in an optimal model, the problem is thus always to optimize the remainder of the production program, that is, to optimize the still undecided product species and product quantities. In such cases, from the product line some preplanned individual products, based on definite sales considerations are included in the mathematical model as conditions and thus have an effect on the remainder of the optimized calculated results.

Thus, since such product line planning and production planning take place mostly as a teamwork and usually around a conference table, it is particularly important that a rapid calculating process and also a rapid optical indication of the results take place. The speed of obtaining results is not only dependent upon the calculating speed of the computer, but first of all, upon the type and extent of the program which, in turn, is dependent upon the number of the individual calculating steps which are necessarily tied with the basic mathematical optimum model. By weighing all circumstances, it may be said that a larger tolerance in the accuracy of the results is of lesser importance than a smaller tolerance which latter, however, gives rise to unproportionately longer calculating times of the computer.

For a solving process in case of a practically still representable, somewhat at greater tolerance of the calculator results, the so-called "transporting method" and thereamong preferably the so-called "Hungarian Method" is very rapid; it requires only one-tenth of the calculating period compared with other solving processes. The so-called Hungarian Method, as a variant of the so-called "transporting method" is described in detail particularly in H. C. Joksch: *Lineares Programmieren*, Tubingen, **1965**, pp. 159–167.

These considerations illustrate the importance of a rapid display of nominal and actual marginal costs.

If the computer calculates optimal values of the quantities of different products, it may be programmed to derive therefrom the degree of utilization of the corresponding production facilities. This makes it possible to ascertain if adjustment of the running program should be made, e.g. it may turn out that one production unit can do the job originally envisaged for two units.

A visual inspection of the board may indicate that a correction should be made in the data or the program supplied to the computer, and the result of such an alteration will then be immediately visible.

It is clear from the above that not only the maximum profit but other optimal conditions may be obtained in similar manner. In many enterprises, such as base material industries, oil shipping and other transport companies, the counterpart of maximum production would be optimum transportation of goods. Assume, for instance, that a company possesses m mines $W_1, W_2 \ldots W_m$ and n blast furnaces $H_1, H_2 \ldots H_n$ at different locations, and the maximum output of the mines and the capacities of the furnaces are known, as well as the cost of transport from any mine to any furnace. It is then of importance to find a transportation plan representing a minimum of cost.

The solution of a model of this type of problem can be found preferably by means of the aforenoted Hungarian Method, described in the cited reference. The result is a set of output values from the computer that may be represented on the indicator board of the present invention.

The term "production process", as used in the present specification and claims, is, therefore, to be taken in its broadest sense to include the production not only of material products but of any kind of effort or work, such as transportation work.

That which is claimed is:

1. A method of monitoring a production process by optically representing industrial management data characterizing the production process, comprising the steps of:

obtaining a first quantity derived from data characterizing quantity and cost figures related to individual products of an industrial process and a second quantity derived from data characterizing the manufacturing steps on the products during operation of said process

deriving electrical signals representing said quantities, and

applying said signals simultaneously to digitally controllable optical display units to cause said units to respond and provide a visual indication corresponding to the magnitudes of the quantities and the utilization of production capacity.

2. A method according to claim 1, in which said quantities are derived from production data by means of a computer.

3. A method according to claim 1, in which said signals are applied to electroluminescent devices for visually comparing the results corresponding to the first and the second quantities.

4. A method according to claim 1, in which said signals are applied to cathode ray tubes.

5. A method according to claim 1, in which electric pulses are derived from one display unit for controlling the combining of the optically displayed quantities and applying them to another optical display unit.

6. A method according to claim 1, in which at least one of said quantities is derived from the movement of an object.

7. A method according to claim 6, in which said object is a machine.

8. A method according to claim 6, in which said object is the product obtained from a process.

9. A system for monitoring a production process by optically representing industrial management data characterizing the production process, comprising:

a general purpose computer including means responsive to data characterizing quantity and cost figures related to individual products and manufacturing steps on the products during the operation of a process for obtaining a quantity derived from said data,

signal-deriving means responsive to said data responsive means for deriving an electrical signal representing said quantity.

a digital signal generator responsive to said electrical signal, and

a digitally controllable optical display unit responsive to said electrical signal to produce an optical signal and further including an additional digitally controllable optical display unit responsive to a switching unit for transforming said electrical signals into secondary signals representing a re-arrangement of said optical signals to provide a visual display of utilization of production capacity.

10. An arrangement according to claim 9, in which the display comprise units an electroluminescent element having a luminescent area of controllable extent and a scale for measuring said extent.

11. An arrangement according to claim 10, in which said element is an electroluminescent strip.

12. An arrangement according to claim 10, in which said scale is exchangeable.

13. An arrangement according to claim 10, in which said element is a strip forming a closed configuration.

14. An arrangement according to claim 13, in which said strip is circular.

15. An arrangement according to claim 13, in which a plurality of parallel strips is provided.

16. An arrangement according to claim 10, in which said element is a circular disc and said luminescent area is a sector of said disc.

17. An arrangement according to claim 10, comprising a plurality of electroluminescent elements of different shape.

18. An arrangement according to claim 17, in which said elements are of different widths.

19. An arrangement according to claim 10, comprising a plurality of electroluminescent elements of different colors.

20. An arrangement according to claim 9, in which said computer is connected via an encoder to each said display unit.

21. An arrangement according to claim 9, in which said digitally controllable optical display units are luminescent screen devices having an input circuit connected to a computer means for producing an electric output signal representing the measure of a quantity characterizing a production process.

22. An arrangement according to claim 21, in which said luminescent screen device is a T.V. image tubes.

23. An arrangement according to claim 21, wherein said luminescent screen devices are connected via electrical switching means for deriving and displaying additional quantities characterizing a production process.

24. An arrangement according to claim 23, in which said luminescent screen devices are adapted for displaying a plurality of signals in different colors.

25. An arrangement according to claim 24, in which the screen have different sections displaying said signal in different colors thereon.

* * * * *

United States Patent [19]

Kelley

[11] **4,019,027**

[45] **Apr. 19, 1977**

[54] **APPARATUS AND METHOD FOR SCHEDULE MONITORING AND CONTROL**

[76] Inventor: **William L. Kelley,** 45 Ashbrook Place, Moraga, Calif. 94556

[22] Filed: **May 9, 1975**

[21] Appl. No.: **576,221**

[52] **U.S. Cl.** ..... **235/89 R;** 35/24 A; 40/19.5; 58/151

[51] **Int. Cl.²** **G09B 19/18;** G09F 3/18; G06C 3/00

[58] **Field of Search** ..... 35/24 A, 24 B; 40/19.5; 58/149, 151; 116/135; 235/89

[56] **References Cited**

UNITED STATES PATENTS

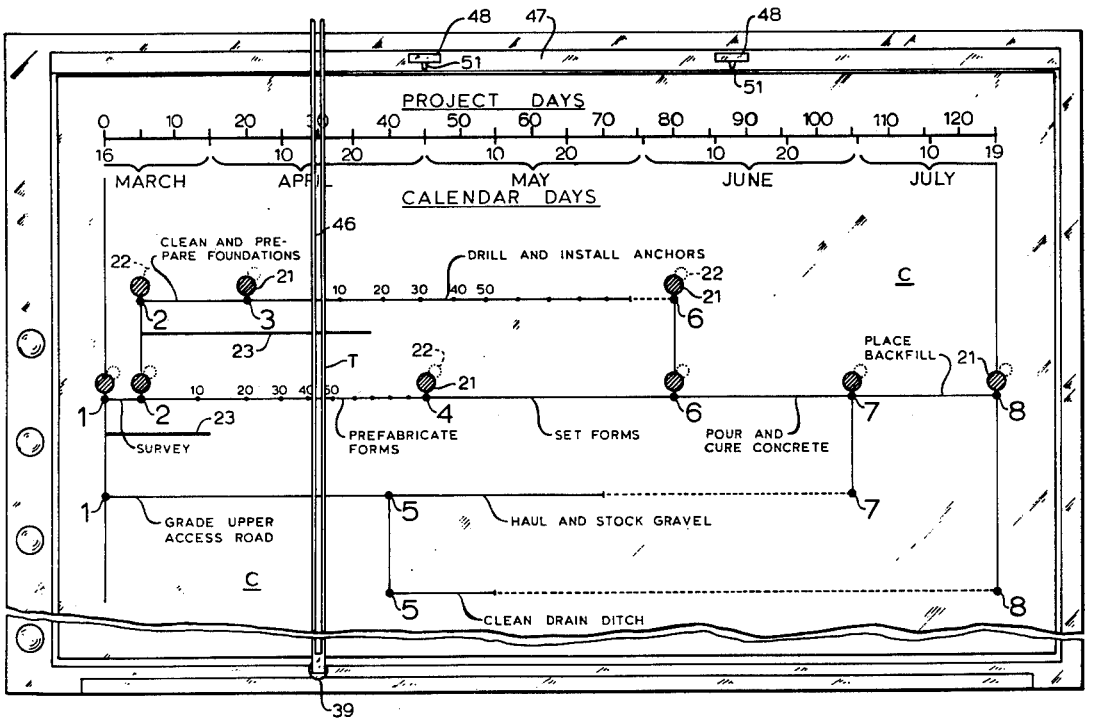
2,647,328	8/1953	Ostrander	35/24 A
2,994,296	8/1961	Waldin	116/135
3,270,709	9/1966	Berge	116/135
3,492,812	2/1970	Cimbal	58/149

Primary Examiner—E. S. Jackmon
Attorney, Agent, or Firm—Stanley Bialos; Alvin E. Hendricson

[57] **ABSTRACT**

For the scheduling of activities, particularly construction activities by the Critical Path Method, an apparatus is provided with a mechanical simulation of the network associated with the method, and is adapted to support a network chart of activities and events. It has a progress bar for monitoring such activities and events, which is power driven at a constant rate for indicating the scheduled progress of the entire project at any given time, whereby the scheduler can determine at a glance at the board what adjustments, if any, need be made in the timing and progress of the various activities of the project.

11 Claims, 14 Drawing Figures



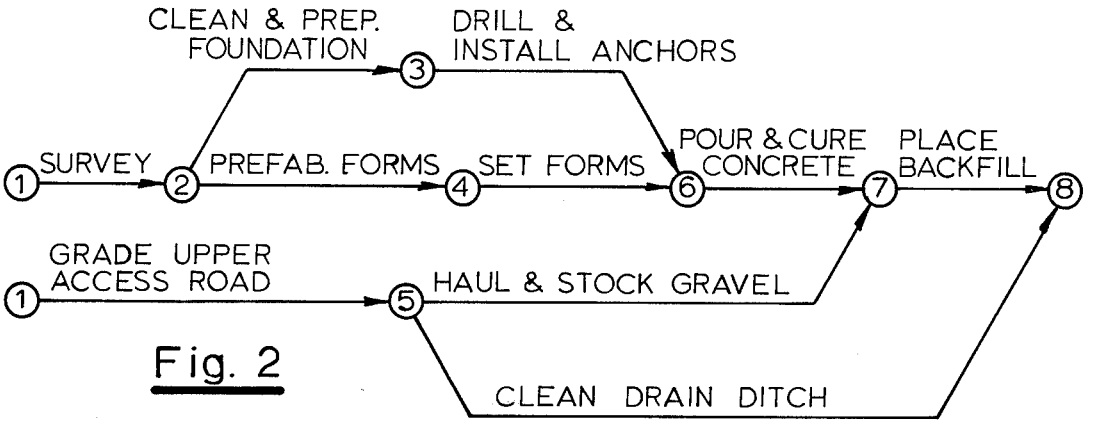
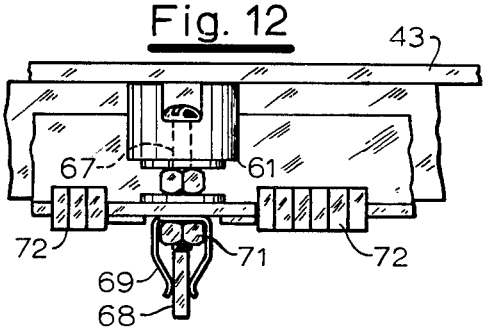


Fig. 2

Fig. 12



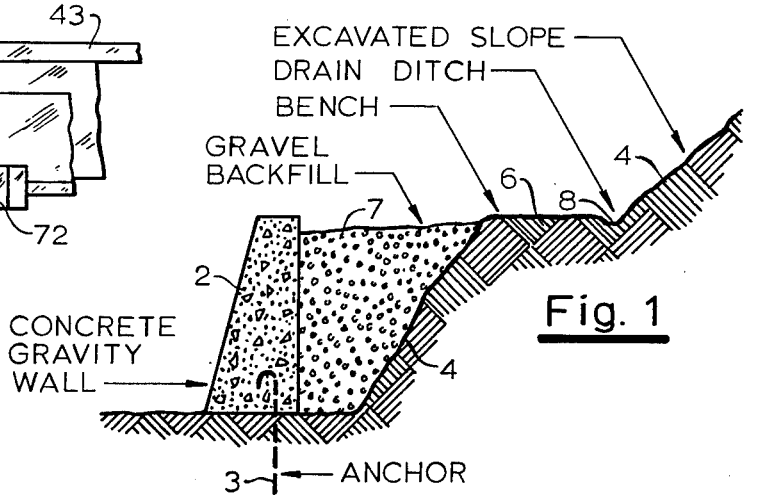


Fig. 1

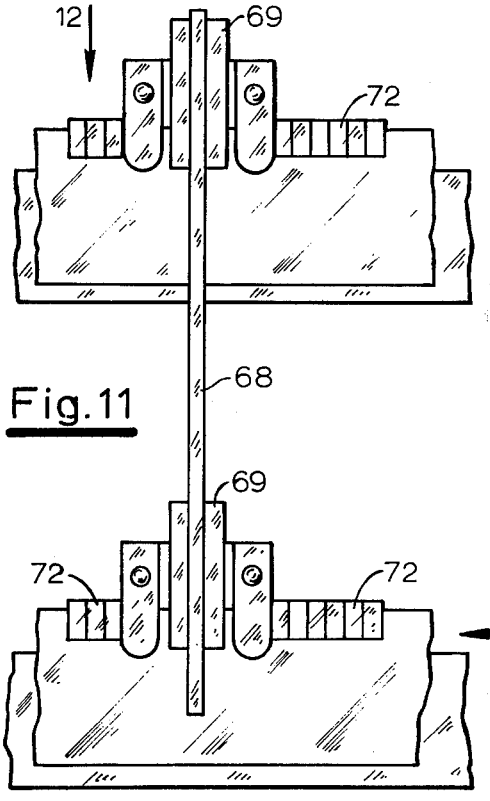


Fig. 11

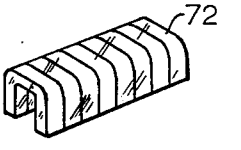
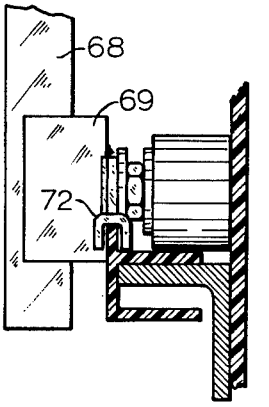


Fig. 13

Fig. 14



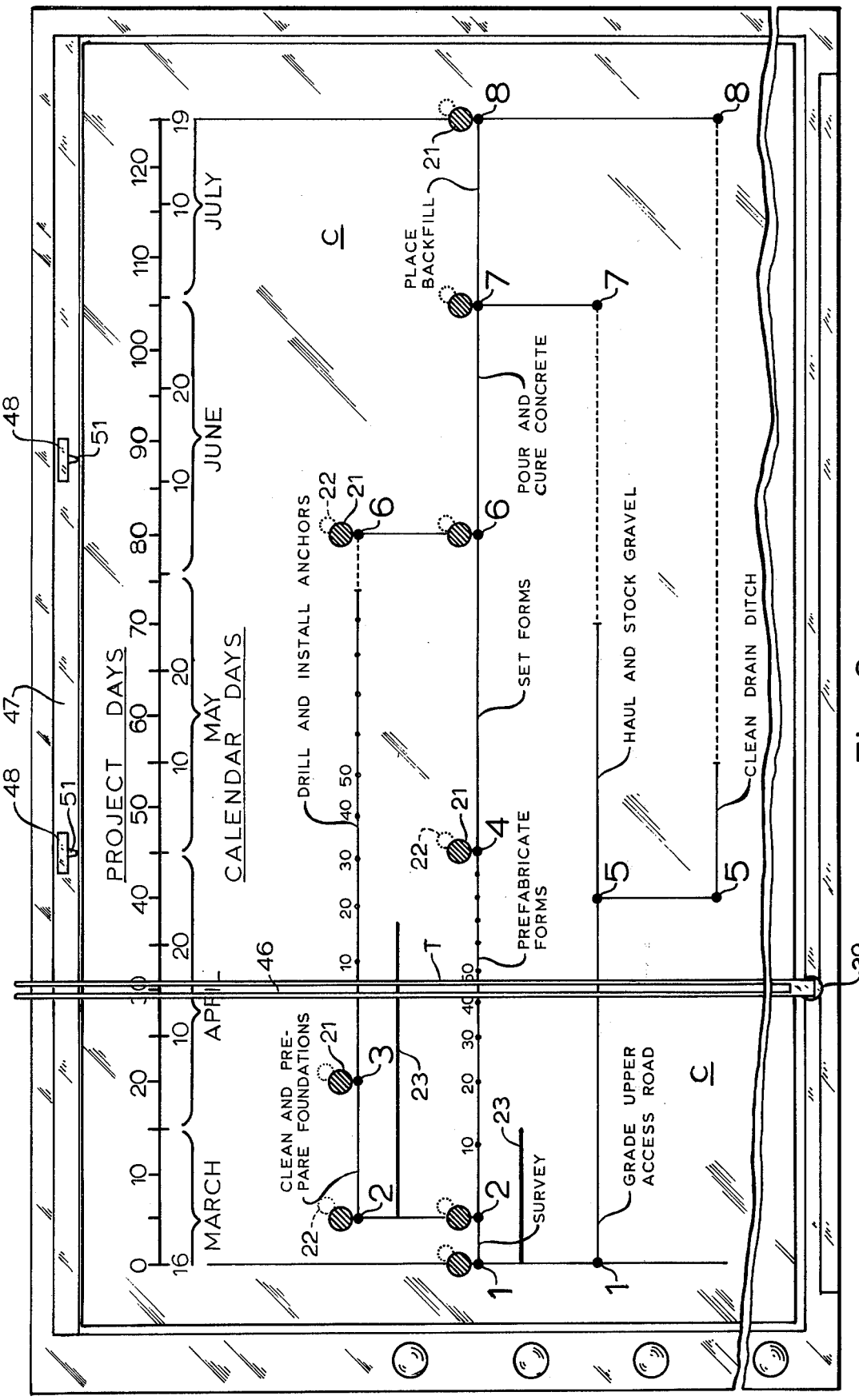

PROJECT DAYS
CALENDAR DAYS
MARCH
APRIL
MAY
JUNE
JULY
CLEAN AND PRE-PARE FOUNDATIONS
DRILL AND INSTALL ANCHORS
PREFABRICATE FORMS
SET FORMS
POUR AND CURE CONCRETE
PLACE BACKFILL
HAUL AND STOCK GRAVEL
CLEAN DRAIN DITCH
GRADE UPPER ACCESS ROAD
SURVEY
C
T

Fig. 3

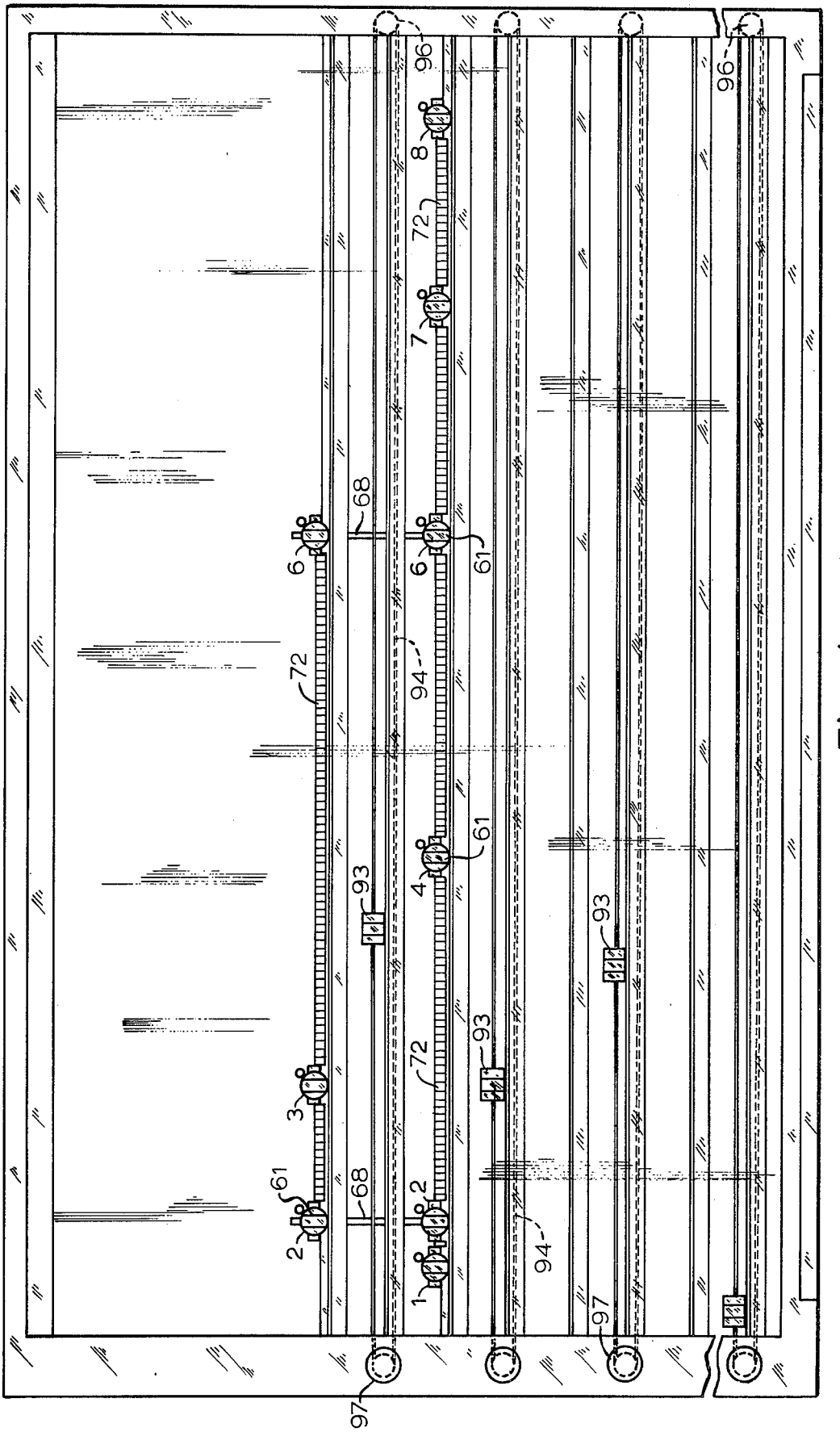


Fig. 4

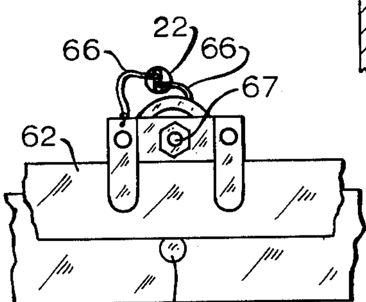


Fig. 7

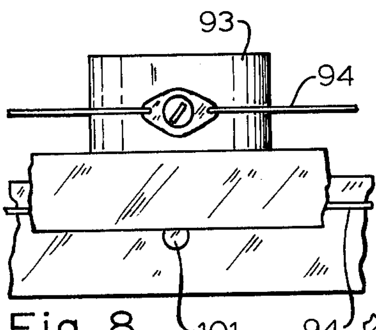


Fig. 8

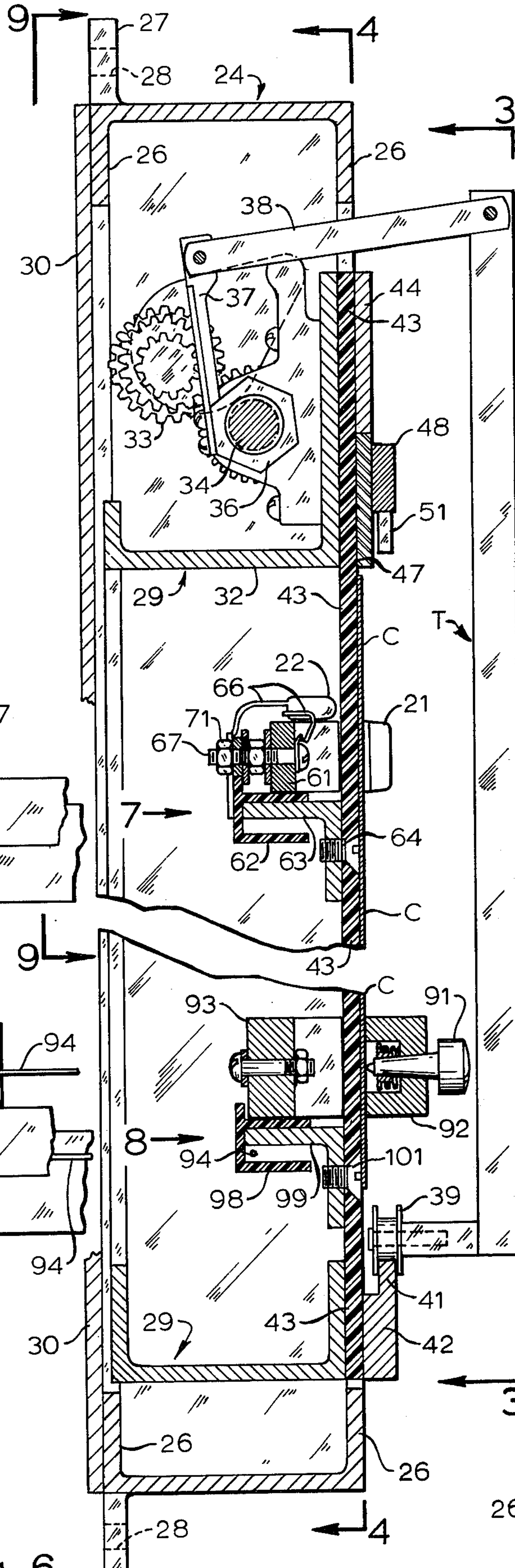


Fig. 6

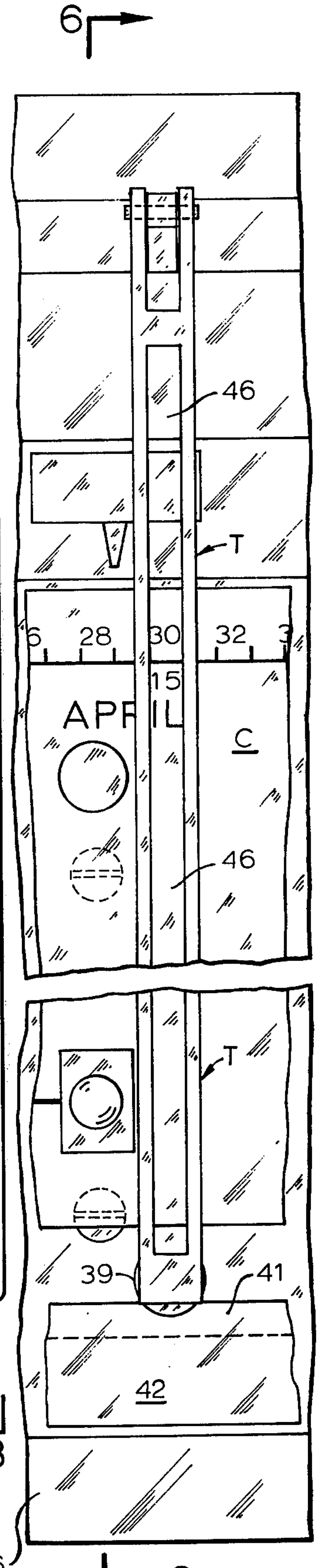


Fig. 5

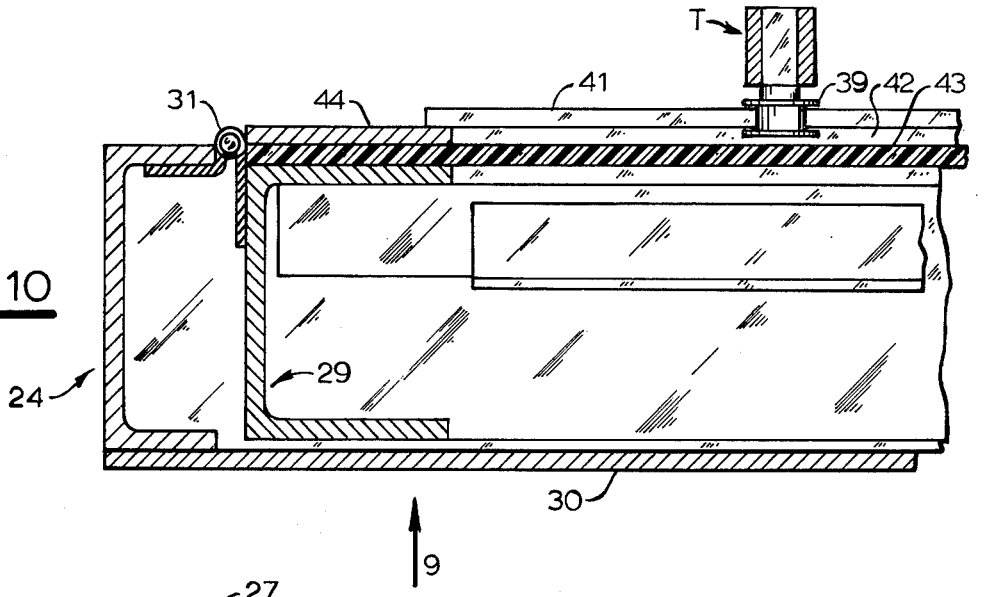


Fig. 10

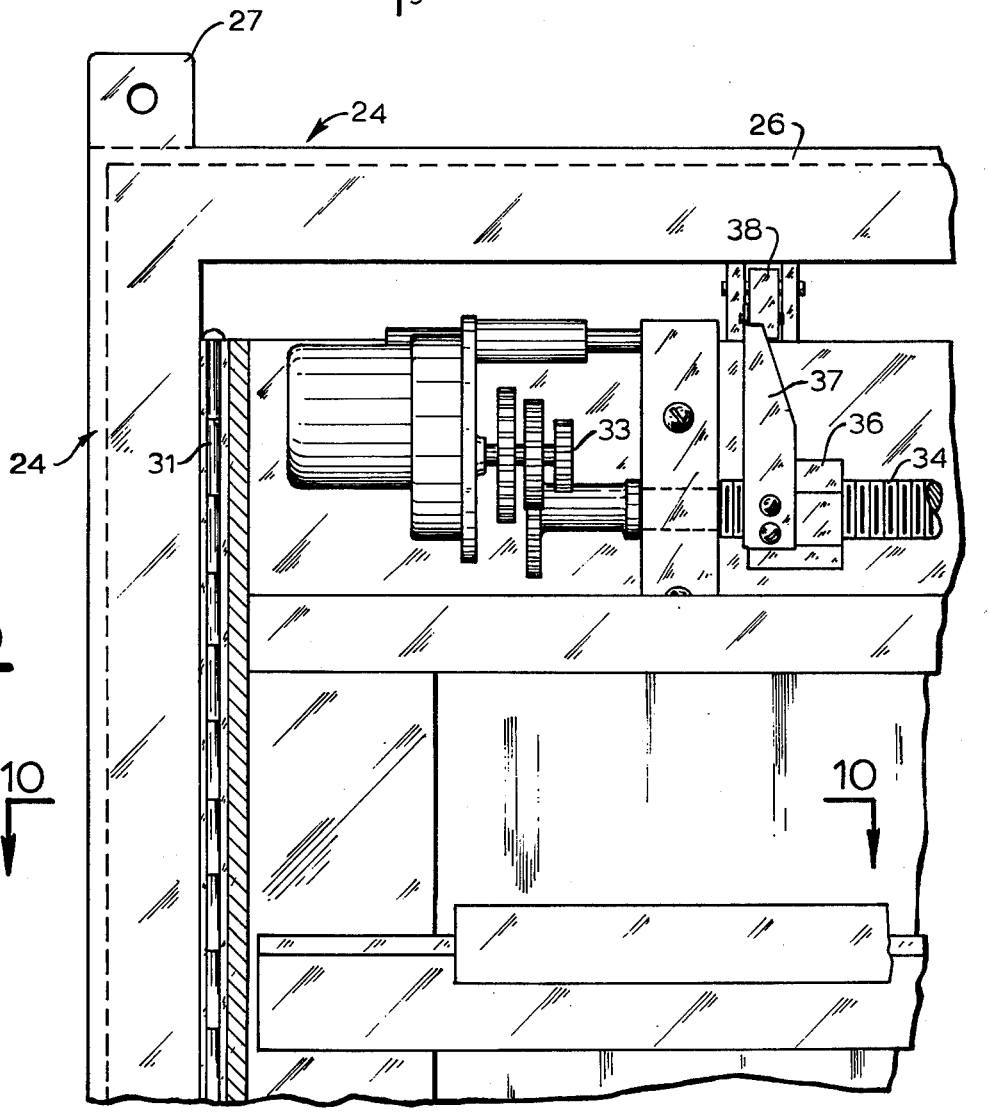


Fig. 9

APPARATUS AND METHOD FOR SCHEDULE MONITORING AND CONTROL

BACKGROUND OF THE INVENTION

The critical path method (CPM) is a comparatively recent engineering development particularly adapted for use in the construction industry for the planning and scheduling of construction acitivities. For a detailed analysis of CPM, reference is made to the book entitled, CPM IN CONSTRUCTION MANAGEMENT, Second Edition, by James J. O'Brien, published by McGraw-Hill Book Company, Copyright 1971.

In construction projects there are activities which may be scheduled to start at certain events (times) and end at other events later on. An acitivity is a work item leading from or to an event; and an event is a point of time for completion or starting or an activity. In the CPM method, the activities are usually drawn on a chart indicated by lines or arrows and which start or terminate where two or more activities meet.

The engineers responsible for the particular scheduling of a project usually have a network chart of events and activities made up beforehand which is based on estimates. When reports from the field are received after the project has started, this network chart is examined regularly (which may be daily or weekly) to determine the progress of the entire project and of its individual activities, which is noted on the chart. If a critical activity is behind schedule, the Project Manager may employ additional resources (e.g. personnel or equipment), and from the chart he can tell whether the entire schedule has to be updated in order to complete it on time. This is a tedious job; and quite a complex one in situations where the project is complex.

In the invention hereof, a monitoring apparatus which will be described in detail hereinafter is provided whereby the project engineers or management in charge of the project may at a glance determine the status thereof so that they may take appropriate action when required.

TYPICAL EXAMPLE OF CPM

As background for explanation of CPM, the following typical example is illustrative. A scale network chart therefor, for the apparatus hereof will be explained later.

With reference to such example, FIG. 1 is a schematic sectional-elevation of a hypothetical project; and FIG. 2 is a network and activity schedule chart drawn for such example. The project of the sample comprises construction of a concrete. gravity wall 2 anchored at 3 near the toe of an excavated slope 4 on which a bench 6 exists, and back filled with gravel 7 between the slope and wall 2 up to the elevation of bench 6 in order to provide a widened road at such elevation. Drain ditch 8 is provided adjacent to the slope and the bench.

Conditions require that the project must be completed within a maximum period of 140 calendar days, including a 15 day contingency allowance for foul weather. Excluding the 15 day allowance for such foul weather, the schedule is 125 calendar days. With these limitations in mind, the management develops a network with activity data for the project on a calendar day basis. The activity data are hand calculated but complex networks often justify computer assistance.

FIG. 2 illustrates the CPM activity and event network for the example of FIG. 1, which is drawn after the activities are determined and their durations estimated by the personnel in charge. In the FIG. 2 network, the activities indicated by straight lines with arrows and the events indicated by numbered circles, are not drawn to a time scale. This is done later when a time bar chart or schedule used in this invention is made up. It will be described later.

The following is a table of activities and events of the FIG. 2 network, as estimated by the management:

Event Interval	Activity	D	ES	EF	LS	LF	F
1 – 2	Survey	5	0	5	0	5	0
1 – 5	Grade Upper Access Rd.	40	0	40	35	75	35
2 – 3	Clean & Prep. Foundation	14	5	19	11	25	6
2 – 4	Prefabricate Forms	40	5	45	5	45	0
3 – 6	Drill & Install Anchors	55	19	74	25	80	6
4 – 6	Set Forms	35	45	80	45	80	0
5 – 7	Haul & Stock Gravel	30	40	70	75	105	35
6 – 7	Pour & Cure Concrete	25	80	105	80	105	0
5 – 8	Clean Drain Ditch	15	40	55	110	125	70
7 – 8	Place Backfill	20	105	125	105	125	0

D - Duration
ES - Early Start
LS - Late Start
F - Float
EF - Early Finish
LF - Late Finish

The activities are listed in a column and are indicated between event numbers. The duration of the entire project is estimated at 125 days as was indicated previously. The estimated duration of each activity in project days is indicated in column D. ES designates teh estimated earliest possible start which can be made for any particular activity determined from the time the project is commenced; LS the latest possible estimated start without delaying the entire program; EF the estimated earliest possible finish of an activity; LF the latest possible estimated finish without delaying the entire program; and F represents float time in project days.

With respect to float, it should be kept in mind that certain activities are not critical, and therefore they will not interfere with completion of the project if the activity is not started or finished within a reasonable time. However, certain activities are critical and have no spare time or float. For example, the "Survey" (1 - 2) must be completed before the two activities "Prefabricate Forms" (2 - 4) and "Clean and Prepare Foundation" (2 - 3), which lead from (1 - 2), can be started. Also, as a further example the activity "Prefabricate Forms" (2 - 4) must be completed before the activity "Set Forms" (4 - 6) can be started. If critical activities, such as 1 - 2, 2 - 4, or 4 - 6 are completed after latest finish (LF) times, the entire project will be delayed.

There are various ways in which "Float" can be calculated. It can be determined by the formula F - LF - EF; or LS - ES; or LF - ES -D.

Those activities which have no float time (zero float) are critical because they must be completed in the scheduled time if the project is not to be delayed. Those which have very little float in days are considered near critical. A near critical acitivity may be considered one in which the float is within about 10% of the calculated duration of the entire project. The more float there is in an activity, the less critical such activity

becomes; and as previously explained, the entire project may be delayed if a critical activity is not started or completed in time. Near critical activities are usually considered the same as critical to provide a factor of safety.

From the above, it will be noted that the critical and near critical activities of the illustrative Example are 1 - 2; 2 - 4; 3 - 6; 4 - 6; 6 - 7; and 7 - 8. Of these, activities 1 - 2; 2 - 4; 4 - 6; 6 - 7; and 7 - 8 comprises the critical path which is defined as the longest route through the CPM network.

SUMMARY AND OBJECTS OF THE INVENTION

Summarizing the invention hereof, it comprises performing the usual CPM estimates and calculations for a particular project as explained above, and making a scaled network chart or schedule in time-bar form, on which the events are plotted at their calculated early start (ES) positions. In cases in which two or more activities share the same total float, this float may be distributed among the affected activities. Thus in the example herein shared float is allocated equally to Activities 2 - 3 and 3 - 6. Therefore the scheduled start of Activity 3 - 6 is 3 days later than its early start, or Day 22.

An apparatus is provided which has a hinged front panel, advantageously transparent, on which the network schedule is mounted. Markers in the form of discs or buttons for various events are removably positioned at the front of the schedule at the various event positions. The markers are of magnetic material, and are held in position by magnets movably supported at the back of the panel so that the markers can be shifted by manually shifting the magnets.

A vertical indicator bar is mounted for movement along the panel, and is moved continuously by a clock driving mechanism which is timed for the entire calculated project. The chart has a horizontal scale drawn thereof which indicates in calendar days as read by the position of the indicator bar, the number of days the project has progressed as scheduled. Horizontal scales are drawn parallel to various activity lines and indicate the estimated percent completion of the respective activities as determined by the position of the indicator bar. Thus, by noting the position of the indicator bar with reference to the activity scales, the Project Manager in charge (who usually must depend on information conveyed to him by subordinates or have to wade through a mass of written reports or data) can by himself determine at a glance the overall progress of the project as compared to the scheduled progress. In this connection, the actual progress of any particular activity can be estimated and recorded on the schedule at any particular time by means of a marking pen. Desirably, such marking pen is provided for each critical or near critical activity on the critical path; and means is provided to hold the marking pen and move it manually along the chart.

Should it be found at any particular time that a critical or near critical activity is behind schedule, job management can readily recognize this from the position of the indicator bar of the apparatus with reference to the estimated actual progress, and take appropriate action such as by adding more personnel to work on the activity. For example, should it be noted that activity "Prefabricate Forms" (2 - 4) is substantially behind schedule even after increased personnel has been employed to work on this activity, the entire project can be rescheduled by rescheduling one or more of the succeeding critical activities for shorter duration or durations, or accept the lost time as probably unrecoverable. If the latter is opted, means is provided on the apparatus for prolonging and updating this schedule, for example, by increasing the overall duration time.

For the purpose of updating or adjusting the event markers, means is provided at the rear of the schedule control panel for manually simultaneously shifting all of the critical event markers at one time so that they need not be individually shifted. This is done by interconnecting such event markers so that when one of the critical markers is shifted, the ones interconnected therewith are also shifted. Shifting of the event markers to update the schedule results in altering the "Float" of the non-critical activities.

From the preceding, it is seen that the invention has as its objects, among others, the provision of an improved simple method and apparatus for determining at a glance the status of any particular project which is programmed by the critical path method, which is simple to operate and perform, and economical. Other projects will become apparent from the following more detailed description and accompanying drawings, in which:

DESCRIPTION OF DRAWINGS

FIGS. **1** and **2** have been previously described.

FIG. **3** is a more or less schematic front elevation of the apparatus with the CPM scale chart thereon, drawn from the information illustrated in FIGS. **1** and **2**; the view looking in a direction of line **3 — 3** in FIG. **6**;

FIG. **4** is a front elevation of a light transmitting panel for mounting the CPM chart, and which is mounted on a hinged frame of the apparatus, with the chart, the indicator bar and event marker discs of magnetic material omitted to illustrate the construction more clearly; the view looking in the direction of line **4 — 4** in FIG. **6**, with parts o the apparatus in the same relationship as shown in FIG. **3**;

FIG. **5** is a fragmentary front elevational view of the apparatus looking at an indicator bar portion thereof;

FIG. **6** is a transverse vertical section taken in a plane indicated by line **6 — 6** in FIG. **5**;

FIG. **7** is a fragmentary rear elevation of the magnet arrangement for holding or retaining event marker buttons; the view looking for holding or retaining event marker buttons; the view looking in the direction of arrow **7** in FIG. **6**;

FIG. **8** is a fragmentary rear elevation of a magnet structure for holding recorder mechanism, looking in the direction of arrow **8** in FIG. **6**;

FIG. **9** is a fragmentary rear elevation of the apparatus looking ir. the direction of arrow **9** in FIG. **10**, with a back cover panel removed to illustrate the support frame arrangement and drive mechanism for the indicator bar, and with a portion of the structure shown in section to illustrate the construction more clearly;

FIG. **10** is a fragmentary horizontal section taken in a plane indicated by line **10 — 10** in FIG. **9**;

FIG. **11** is a fragmentary rear elevation view illustrating a form of vertical connection between magnet mechanism, looking at the marker **6** portion of the apparatus in FIG. **3**;

FIG. **12** is a fragmentary top plan view of the portion of the mechanism shown in FIG. **11**, looking in the direction of arrow **12** in FIG. **11**;

FIG. 13 is a isometric view of a type of spacer bar between magnet mechanism; and

FIG. 14 is a fragmentary end elevation, looking in the direction of arrow 14 in FIG. 11.

DETAILED DESCRIPTION

As was noted previously, after the activities have been determined and outlined as indicated in FIGS. 1 and 2, and by the table for FIG. 2, the time bar chart or schedule of the entire project is drawn to scale for various duration times. A suitable scale may be:

Scale	Schedule Duration Range
0.1 in. = 1 day	1 to 2 yrs.
0.2 in. = 1 day	6 mos. to 1 yr.
0.4 in. = 1 day	6 mos. or less

For schedules longer than two years' duration or should it be desired to use a larger scale, two or more monitoring boards or panels may be employed, or the schedule may be divided into two or more successive networks.

A suitable paper size for the schedule is about 40 by 84 inches; and it has been found that lines of activities which extend horizontally should be spaced about 3 inches apart vertically. The board may be of any suitable vertical height to accommodate the activity lines spaced apart vertically; a suitable number being about 10 to 12. Shared float among the activities where this occurs is distributed by allocation.

FIG. 3 is a schedule of the project previously explained, drawn from the data depicted in FIG. 1 and 2. Events for critical and near critical activities are indicated by circles but non-critical activities are shown on the chart solely to complete the network and to enable overview thereof. It will be noted from FIG. 3 that the activities follow the order indicated in the aforementioned table. Where more than one activity terminates or originates from an event, such events are separated vertically on the schedule. Float times for non-critical activities are indicated by dotted lines. A scale in calendar days is drawn on the chart indicating the date of termination (duration). This same scale also indicates the estimated duration of the project (project days), commencing with 0 start and ending at the estimated end of the project (125 days).

The activity lines between events are scaled to indicate the estimated duration of the respective activities divided into estimated percentage of scheduled completion of the particular activity at any particular time during its estimated duration determined with reference to the position of the indicator bar. Thus, it will be noted for example that the activity "Drill and Install Anchors" (3 - 6) having an original estimated duration of 55 days is divided into increments indicating percent completion. These are estimated for all activities by the project management as previously related; and it will be observed that the spacing for the percentages decreases with time on the theory that as work on the activity progresses, the personnel become more familiar with it and can work more efficiently at the end than at the start of the project. A time bar *T*, to be described later, on the apparatus is provided which is moved by clock drive mechanism to indicate scheduled progress of the project in calendar days.

As will be described later in greater detail when the apparatus structure is described, the prepared schedule or chart is mounted on a panel; and means is provided for removably positioning critical or near critical event markers 21, which are in the form of discs, at the points such events had been calculated to occur. For example, event 2 occurs after the "Survey" (1 - 2) has been completed, which is 5 days after commencement of the project, and the marker therefor is positioned at such point. The marker for event 3 is placed at a position 14 days after event 2, etc. For clarity, these event markers 21 are schematically illustrated in FIG. 3 in offset relationship with reference to the event points, and are desirably illuminated as indicated by the dotted circles 22 in FIG. 3. As will be described later, the markers are held by magnetic means at the back of the panel.

Also, means is provided for recording the overall progress of each of the critical or near critical activities by providing a recorder which is moved manually along the panel and draws a recorder line 23 adjacent the activity line. This record is made by the management based upon an estimate of how the work for such activity has progressed. Instead of providing such recorder on the schedule itself, the same effect may be obtained by the Project Manager, by drawing the line on the schedule by a hand held pencil or pen.

As illustrative of how the schedule is used in the apparuats, assume that on Apr. 15, 30 days after the project start, the panel and board control schedule appear after examination as shown in FIG. 3. It will be noted that the project is in difficulty because critical activity 2 - 4 is behind schedule 15 days. This is ascertained from the fact that the time bar *T* is at 30 project days at which the activity should be about 45% completed, but the actual completion, as indicated by record line 23 just below activity line 2 - 4, is only about 10% which should have occurred about 15 days after the project commenced instead of 30 days. Therefore, management has to make a decision whether to prolong the completion date of the project or to recover lost time.

Assuming that a decision is made to add a carpenter crew in order to increase the production rate of prefabricating the forms (activity 2 - 4), and after this is done the activity is still behind schedule by 10 days, then management must make another decision of whether to reschedule one or more of the succeeding critical activities for shorter duration or durations. or accept the lost time as probably unrecoverable. The desirable form of apparatus will now be described, including its means for rearranging the schedule or timing by prolonging of updating it.

Referring to FIGS. 3 through 14 which illustrate the apparatus structure, it comprises a support frame 24 of rectangular box shape having vertical flanges 26 which form sides channel-shaped in cross section. Outwardly extending lugs 27 having apertures 28 provide means for attaching the frame to a suitable support such as a wall or easel. Frame 24 houses an inner frame structure 29 which carries apparatus components and which it pivotally or hingedly connected along one side edge thereof by means of hinge 31 to a side flange of frame 24. Frame 29 provides a support for the chart and also for driving means for indicator bar T which is mounted on frame 29. The rear of frame 24 is covered by detachably connected backing panel 30.

As can be seen best from FIGS. 6 and 9, frame 29 has an upper horizontal channel 32 above which is mounted clock drive gearing 33 connected to drive a screw 34. A nut 36 on screw 34 is attached to an up-

standing link 37 to which is pivotally connected a link 38 which is turn is pivotally connected to the top end of indicator bar T. As the clock gearing 33 is driven, indicator bar T is moved longitudinally along the chart at a constant rate. In this connection, gearing 33 comprises replaceable speed changed gearing so that the rate may be fixed in accordance with the scale of the chart desired.

The lower end of indicator bar T carries a freely journalled rotatable roller 39 which rides on a rail 41 forming part of a bar 42 secured to the bottom of frame 29. Bar 42 also attaches a light transmitting support panel 43 (desirably of plastic material, such as "Plexiglas") to inner frame 29; suitable additional bars 44 also being provided to secure light transmitting panel 43 to frame 29. The aforementioned pivotally connected link 38 spaces indicator bar T away from panel 43 and away from the schedule chart C which is attached to panel 43 by any suitable means such as "Scotch Tape" along its edges. The chart is desirably of light transmitting material, such as tracing paper or the like.

For ease of reading, indicator bar T is formed with a longitudinally extending open narrow space 46 through which indicia on the scales may be more readily seen. A longitudinally extending bar 47 of magnetic material is provided above chart C upon which so-called milestone event markers are removably positioned, each comprising a magnet 48 and a pointer 51. Milestone events are those of special importance.

As was previously related, event markers 21 are of magnetic material. They are removably held over the desired critical or near critical events. Means for movably retaining markers 21 for freely slidable movement along the chart C when desired, comprises magnets 61 each of which is supported on an insulating U-shaped track member 62, desirably of plastic material, in turn mounted on an angle member 63 secured to light transmitting panel 43 by countersunk screws 64. Because each magnet 61 is supported for slidable movement along the track, it may be readily positioned at a desired event point on the chart at which an event marker 21 may be held by the magnet.

Each magnet 61 supports the aforementioned light source 22 which illuminates the associated marker 21 by light transmitted through transparent support panel 43. Light transmitting panel 43 also serves to allow the scheduler to see where to set up the magnets 61 in desired positions to hold markers 21. All the light sources, desirably electric light bulbs 22, are connected by any conventional circuity including wires 66 suitably insulated from the magnets and held in position by bolts 67.

As was previously noted, where two or more activities start from the same event, a plurality of rows of vertically displaced event markers is provided such as 2 - 2 and 6 - 6 shown in FIGS. 3 and 4. In order that the vertically spaced magnets for such events may be shifted in unison together, they are rigidly connected together by a connecting bar 68, as can be seen from FIGS. 11, 12 and 14. For attaching the bar to a magnet, a U-shaped clip 69 is fastened to the magnet structure by a nut 71 which cooperates with bolt 67. The clip rigidly clamps the bar so that two of the magnet structures, one above the other, can be manually moved laterally in unison when desired.

In revising (updating) the schedule for reasons previously explained, a plurality of magnet structures is manually shifted or moved to the right or left with reference to FIGS. 3 and 4. They are so interconnected that all of them that are to be shifted can be simultaneously moved in unison merely by pushing one of the magnet structures from left to right or vice versa, thus updating the schedule by prolonging or shortening it, respectively. Means for effecting such shifting includes readily convenient structure for initially properly spacing the magnets apart when the apparatus is first set up, so that the event markers 21 retained by magnets 61 will be properly spaced apart.

For such purpose, a premanufactured conventional bundle 72 of individual staples adhesively secured together edge to edge is most suitable because the bundle can be readily adjusted to the desired length by removing staples or by combining bundles. When so adjusted to space two event markers apart for example 4 – 6 as shown in FIG. 4, it will maintain the desired spacing whereby all of the markers in advance of a selected marker may be moved in unison merely by manually pushing a selected magnet structure which retains the marker. The magnet structures which are connected together by an upright bar 68 will also be shifted simultaneously. Not only do the bundles 72 act as spacers, but they are desirably of metal so that they can form part of the illumination circuity which also includes aforementioned wires 66.

The structural mechanism has been described which relates to monitoring the actual progress of critical or near critical activities so as to relate them to the scheduled progress as forecast at the beginning when plans were initially made by the management. Accordingly actual progress as recorded on the chart from field reports can be compared readily at any time with scheduled progress shown by the indicator bar T.

Desirably, the aforementioned record line 23 should be drawn indicating the actual progress of the respective particular activities at the time. Line 23 may be drawn from time to time on the chart by hand held pencil or pen. However, recording means is provided forming part of the apparatus to enable drawing of such line conveniently.

Such recording means, as can be seen best from FIGS. 6 and 8, comprises a spring pressed writing instrument 91 slidably mounted on a button magnet 92 which is pressed against chart C by the spring. Magnet 92 is fixedly held against the chart by means of a cooperating horeshoe magnet 93 which is secured to an endless flexible strand 94 which runs about a pulley 96 at one end and about a pulley 97 at the opposite end having a manually controllable knob by which pen 91 may be moved across the chart. Magnet 93 can slide along insulating track 98 adhered to angle member 99 which supports the track; angle member 99 being secured to the transparent panel 43 by means of countersunk screws 101.

I claim:

1. Schedule control and monitoring apparatus for use in network type scheduling of a project having a plurality of interrelated activities with events at the start and termination thereof and wherein a succession of activities of the longest time path for the project are critical, said apparatus comprising a support frame, means for mounting on said frame a prepared chart on which a network schedule of such activities and events is written chronologically with the distance of consecutive critical activities drawn to scale representing scheduled times based on estimates of the durations to complete

the respective activities, a plurality of magnets movably disposed on said frame behind said chart whereby the positions thereof can be changed, a plurality of markers of magnetic material adapted for disposition on the front face of said chart at said event points and supported by said magnets, an indicator bar mounted on said frame over said chart, and means for moving said indicator bar at a constant rate past said markers from a position indicating the commencement of the project to provide a comparison between the actual progress of any given activity and the scheduled progress thereof.

2. The apparatus of claim 1 wherein means interconnects a plurality of said magnets for simultaneously shifting said magnets together along said chart when the schedule is to be changed.

3. The apparatus of claim 1 further comprising a plurality of recording instruments for various respective activities; each of said instruments being mounted on a body of magnetic material at the front of the chart, a magnet for each of said bodies disposed at the back of the chart for retaining said respective body and pressing the instrument thereof against chart, a track on which each magnet is slidably mounted, and manual means including a control knob for sliding said magnet along said track.

4. The apparatus of claim 3 wherein said manual sliding means includes an endless flexible member to which the magnet is attached.

5. The apparatus of claim 1 wherein the support frame includes a panel of light transmitting material to which the chart is attached at the front side of the panel, the magnets are slidably mounted on tracks at the rear side of the panel, and means carried by respective magnets is provided to illuminate and respective marks supported thereby of events at the start and termination of critical activities.

6. The apparatus of claim 2 wherein said support frame includes a panel to which the chart is attached at the front side of the panel, means is provided for spacing the indicator bar to clear said markers as the indicator bar moves, means is provided for mounting said panel for hinged movement along one edge about an upright axis to allow the panel to be swung for access to the interconnected magnets at the rear of said panel, and the means spacing said indicator bar to clear said markers includes linkage connected to the top of the bar for pivotal movement about a horizontal axis.

7. The apparatus of claim 6 wherein a track at the bottom of the panel is provided, and a roller is journalled at the bottom bar rides on said track.

8. Schedule control and monitoring apparatus for a critical path method project which has a succession of activities some of which are critical whereby they must be completed on time if the project is not to be delayed and also comprising events which are times at which activities start or terminate, said apparatus comprising a support frame including a transparent panel, means mounting on the front face of said panel a prepared chart on which a network schedule of such activities and events is written chronologically with the distance between consecutive critical activities drawn to scale representing scheduled times based on estimates of the durations to complete the respective activities, magnets slidably mounted at the rear face of said panel for respectively supporting a plurality of markers of magnetic material on the front face of said chart at event points, means interconnecting a plurality of said magnets for simultaneously shifting said magnets along said chart when the schedule is to be changed, an indicator bar mounted on said frame over said chart, and means for moving said indicator bar at a constant rate past said markers from a position indicating the commencement of the project to provide a comparison between the actual progress of any given activity and the scheduled progress thereof.

9. A schedule control and monitoring method for use in network type scheduling wherein a plurality of interdependent activities culminating in events of a project are to be scheduled and monitoring during progress of the project comprising the steps of:

providing a chart on which a network schedule of activities and events of the project are presented chronologically with the distance between consecutive events drawn to a time scale respresenting scheduled times for performance of activities based on estimates of the durations thereof,

mounting said networks chart upon a front face of a panel,

forming a mechanical counterpart of the network of said chart on the back face of said panel by placing magnets at the locations of said events on the chart and mechanically linking the magnets between the events as related on the chart,

placing a magnetically responsive marker on said chart at each event and supporting said markers by said magnets,

drawing lines indicating the actual progress of said activities on the chart to the scale thereof, and

moving an indicator bar at a constant rate related to the scale of the chart across the front face of said panel over said chart to provide a visual comparison of actual progress of the project to estimated times of completion of activities thereof.

10. The method of claim 9 further defined by

providing illumination at each magnet, and

electrically and mechanically linking said magnets in the manner events are connected on the chart and connecting said electrical linkage across a power supply whereby only markers located at events of a critical longest path of activities of the project are illuminated to visually depict the critical path of the project.

11. The method of claim 10 further defined by

elongating the linkage between magnets corresponding to markers at the beginning and end of activities which are not to be completed in estimated times to thus reposition magnets and markers on said chart for illuminating the markers of events of an adjusted longest critical path of activities to provide a visual indication of adjusted network schedule and comparison to original network schedule and actual activities progress.

* * * * *

United States Patent [19]

Smith et al.

[11] **4,336,589**

[45] **Jun. 22, 1982**

[54] **WAREHOUSING MONITOR AND CONTROL SYSTEM**

[75] Inventors: **Charles A. Smith; Robert T. Danevicz,** both of Grand Rapids, Mich.

[73] Assignee: **Rapistan Division, Lear Siegler, Inc.,** Grand Rapids, Mich.

[21] Appl. No.: **138,250**

[22] Filed: **Apr. 7, 1980**

[51] **Int. Cl.**[3] **G06F 15/24**

[52] **U.S. Cl.** **364/403;** 198/347; 198/418; 364/478; 364/900

[58] **Field of Search** ..... 364/403, 478, 200, 900; 235/432, 385, 91 L; 340/150; 198/347, 418

[56] **References Cited**

U.S. PATENT DOCUMENTS

2,995,729	8/1961	Steele	364/403
3,144,958	8/1964	Gumpertz	235/385
3,637,989	1/1972	Howard et al.	235/91 L
3,651,478	3/1972	Shandlay	364/900
3,688,087	8/1972	Howard et al.	235/385
3,737,631	6/1973	Harris	235/385
3,899,775	8/1975	Larsen	340/150
4,024,380	5/1977	Gunn	235/432
4,135,241	1/1979	Stanis et al.	235/385
4,141,078	2/1979	Bridges et al.	364/403
4,180,204	12/1979	Koenig et al.	235/385

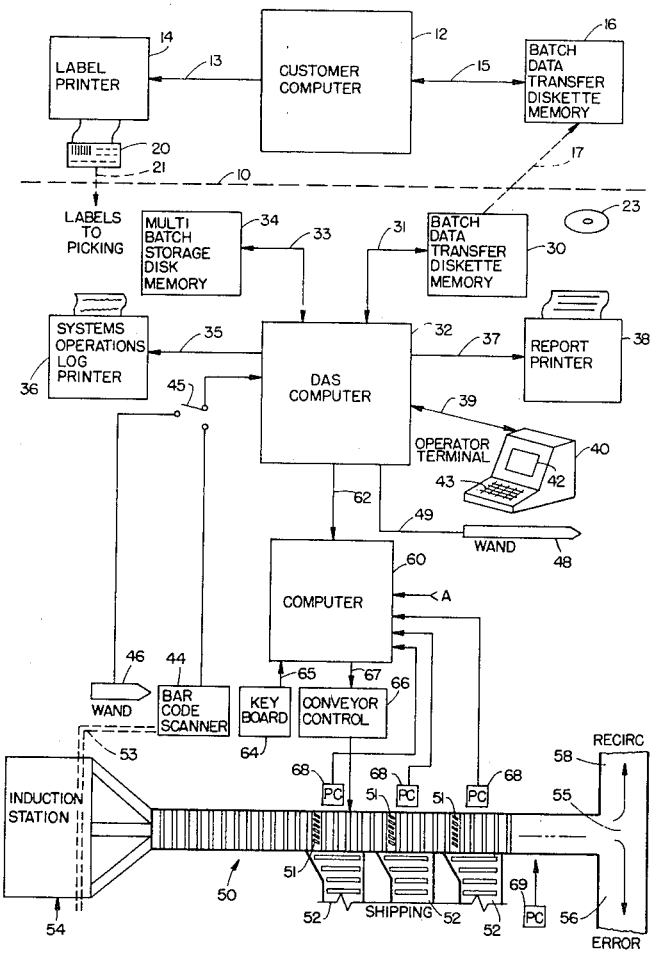
Primary Examiner—Errol A. Krass

Attorney, Agent, or Firm—Price, Heneveld, Huizenga & Cooper

[57] **ABSTRACT**

A warehouse product monitoring and control system includes a distribution audit system computer which receives batch picking information on a recording medium identifying each transaction or article to be picked with a unique number. Associated with each unique number is processing information for controlling the sorting of the article within the system. The same number, unique for each article, is contained on a label positioned on each article as it is picked. The computer control provides continuous monitoring of the article as it is scanned by one or more label reading devices such that the picking and sorting status of each article to be picked can be continuously monitored. The system includes operator interface terminals and printers for providing a variety of status reports to the operating personnel such that the operational status of the system can be continuously monitored.

17 Claims, 9 Drawing Figures



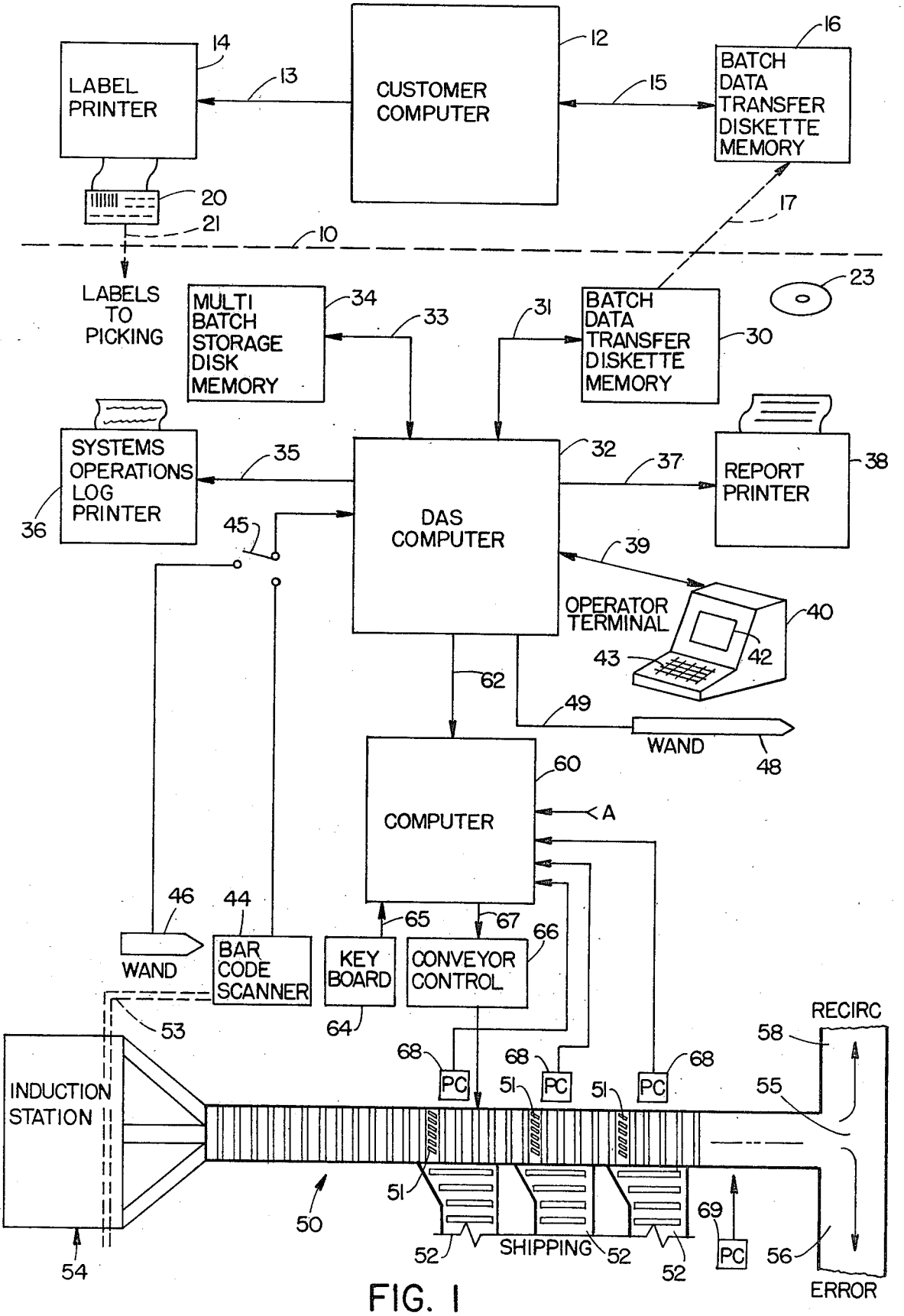

LABEL PRINTER
14
13
CUSTOMER COMPUTER
12
15
BATCH DATA TRANSFER DISKETTE MEMORY
16
17
20
21
10
LABELS TO PICKING
MULTI BATCH STORAGE DISK MEMORY
34
33
31
BATCH DATA TRANSFER DISKETTE MEMORY
30
23
SYSTEMS OPERATIONS LOG PRINTER
36
35
DAS COMPUTER
32
37
REPORT PRINTER
38
45
39
OPERATOR TERMINAL
40
43
42
62
49
WAND
48
COMPUTER
60
A
46
44
65
67
66
WAND
BAR CODE SCANNER
KEY BOARD
CONVEYOR CONTROL
64
53
RECIRC
58
68
PC
51
55
INDUCTION STATION
50
69
56
54
52
SHIPPING
ERROR

FIG. 1

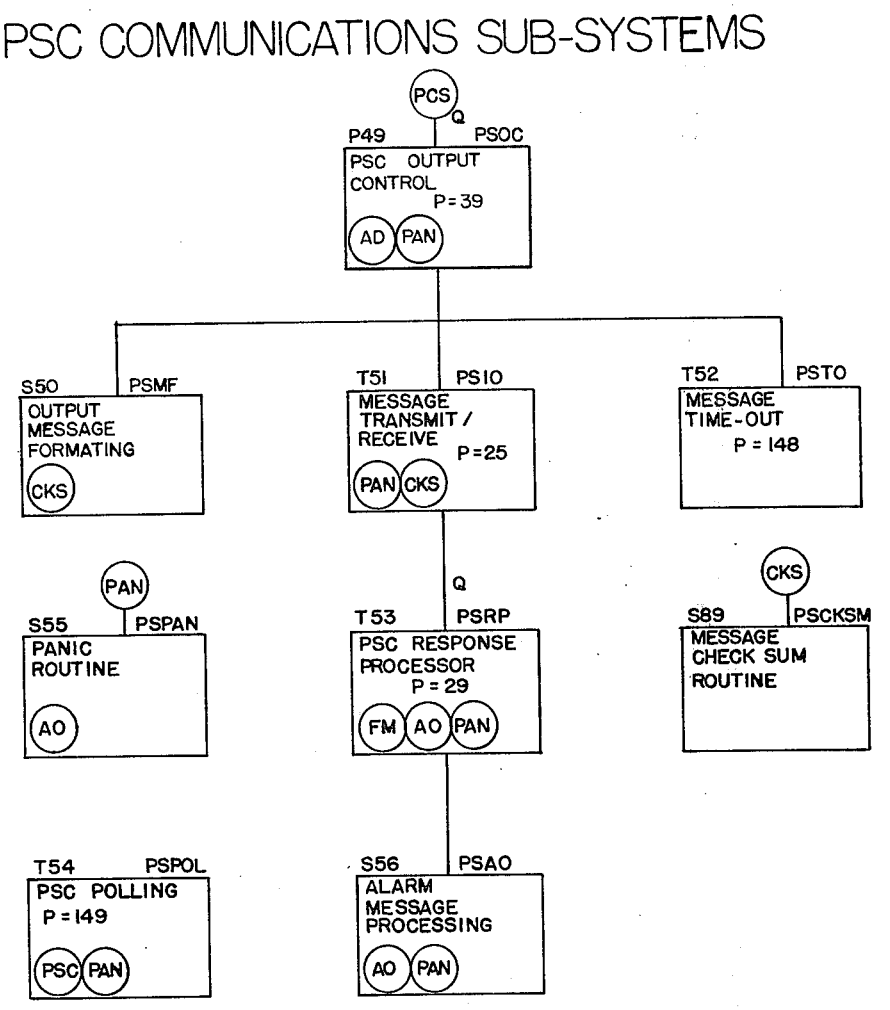

PSC COMMUNICATIONS SUB-SYSTEMS
PCS
Q
P49
PSOC
PSC OUTPUT CONTROL
P=39
AD
PAN
S50
PSMF
OUTPUT MESSAGE FORMATING
CKS
T51
PSIO
MESSAGE TRANSMIT / RECEIVE
P=25
PAN
CKS
T52
PSTO
MESSAGE TIME-OUT
P=148
PAN
S55
PSPAN
PANIC ROUTINE
AO
Q
T53
PSRP
PSC RESPONSE PROCESSOR
P=29
FM
AO
PAN
CKS
S89
PSCKSM
MESSAGE CHECK SUM ROUTINE
T54
PSPOL
PSC POLLING
P=149
PSC
PAN
S56
PSAO
ALARM MESSAGE PROCESSING
AO
PAN

FIG. 7

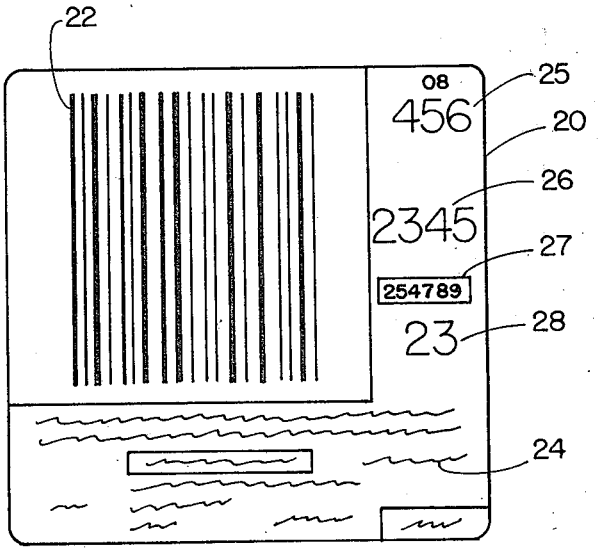

22
08
456
25
20
26
2345
27
254789
23
28
24

FIG. 2

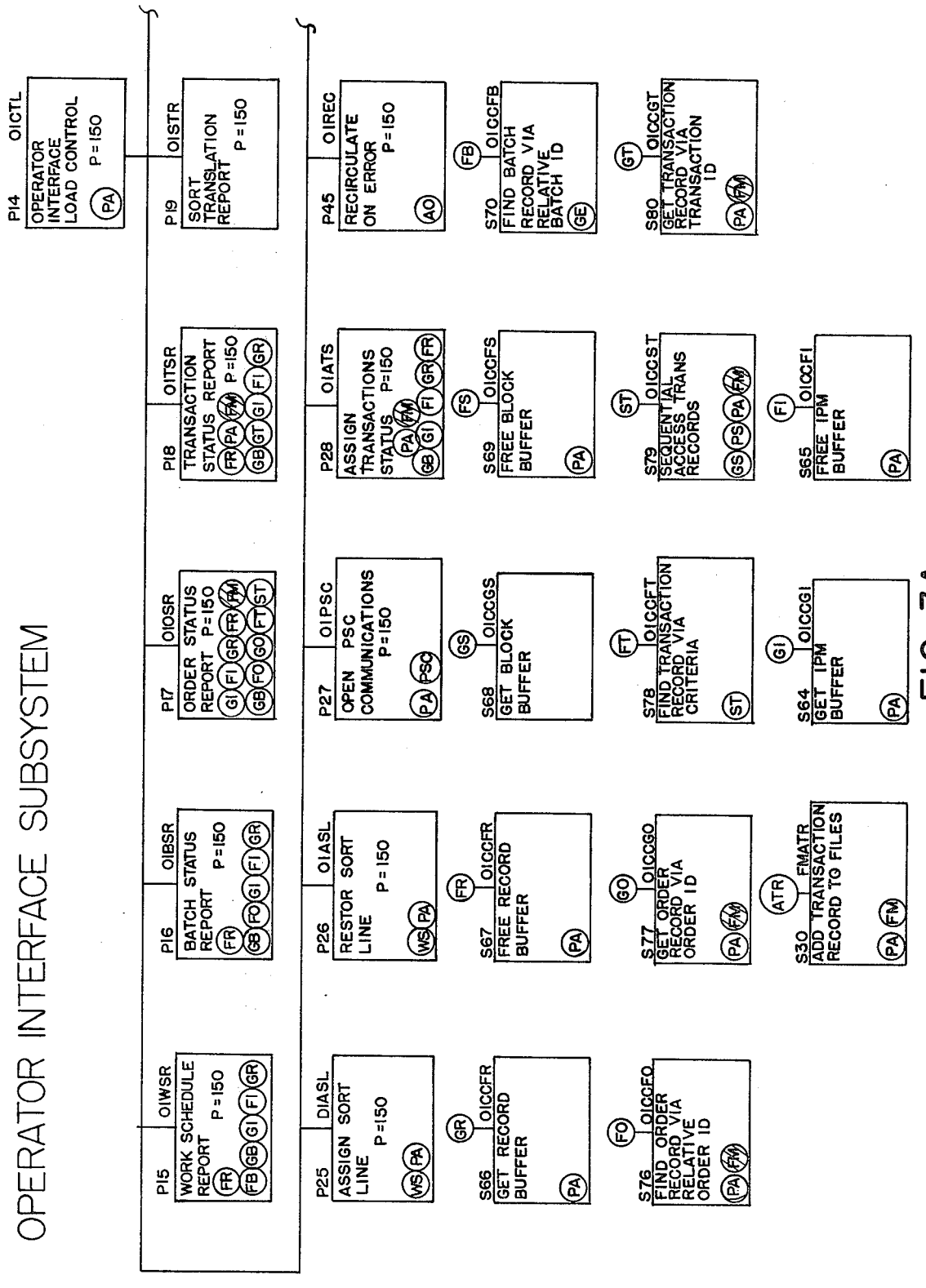

OPERATOR INTERFACE SUBSYSTEM
P14 OICTL
OPERATOR INTERFACE LOAD CONTROL
P = 150
PA
P19 OISTR
SORT TRANSLATION REPORT
P = 150
P15 OIWSR
WORK SCHEDULE REPORT
P = 150
FR FB GB GI FI GR
P16 OIBSR
BATCH STATUS REPORT
P = 150
FR GB FO GI FI GR
P17 OIOSR
ORDER STATUS REPORT P = 150
GI FI GR FR FM GB FO GO FT ST
P18 OITSR
TRANSACTION STATUS REPORT
P = 150
FR PA FM GB GT GI FI GR
P45 OIREC
RECIRCULATE ON ERROR
P = 150
AO
P25 DIASL
ASSIGN SORT LINE
P = 150
WS PA
P26 OIASL
RESTOR SORT LINE
P = 150
WS PA
P27 OIPSC
OPEN PSC COMMUNICATIONS
P = 150
PA PSC
P28 OIATS
ASSIGN TRANSACTIONS STATUS
P = 150
PA FM GB GI FI GR FR
GR
S66 OICCFR
GET RECORD BUFFER
PA
FR
S67 OICCFR
FREE RECORD BUFFER
PA
GS
S68 OICCGS
GET BLOCK BUFFER
FS
S69 OICCFS
FREE BLOCK BUFFER
PA
FB
S70 OICCFB
FIND BATCH RECORD VIA RELATIVE BATCH ID
GE
GT
S80 OICCGT
GET TRANSACTION RECORD VIA TRANSACTION ID
PA FM
FO
S76 OICCFO
FIND ORDER RECORD VIA RELATIVE ORDER ID
PA FM
GO
S77 OICCGO
GET ORDER RECORD VIA ORDER ID
PA FM
FT
S78 OICCFT
FIND TRANSACTION RECORD VIA CRITERIA
ST
ST
S79 OICCST
SEQUENTIAL ACCESS TRANS RECORDS
GS PS PA FM
ATR
S30 FMATR
ADD TRANSACTION RECORD TO FILES
PA FM
GI
S64 OICCGI
GET IPM BUFFER
PA
FI
S65 OICCFI
FREE IPM BUFFER
PA

FIG. 3A

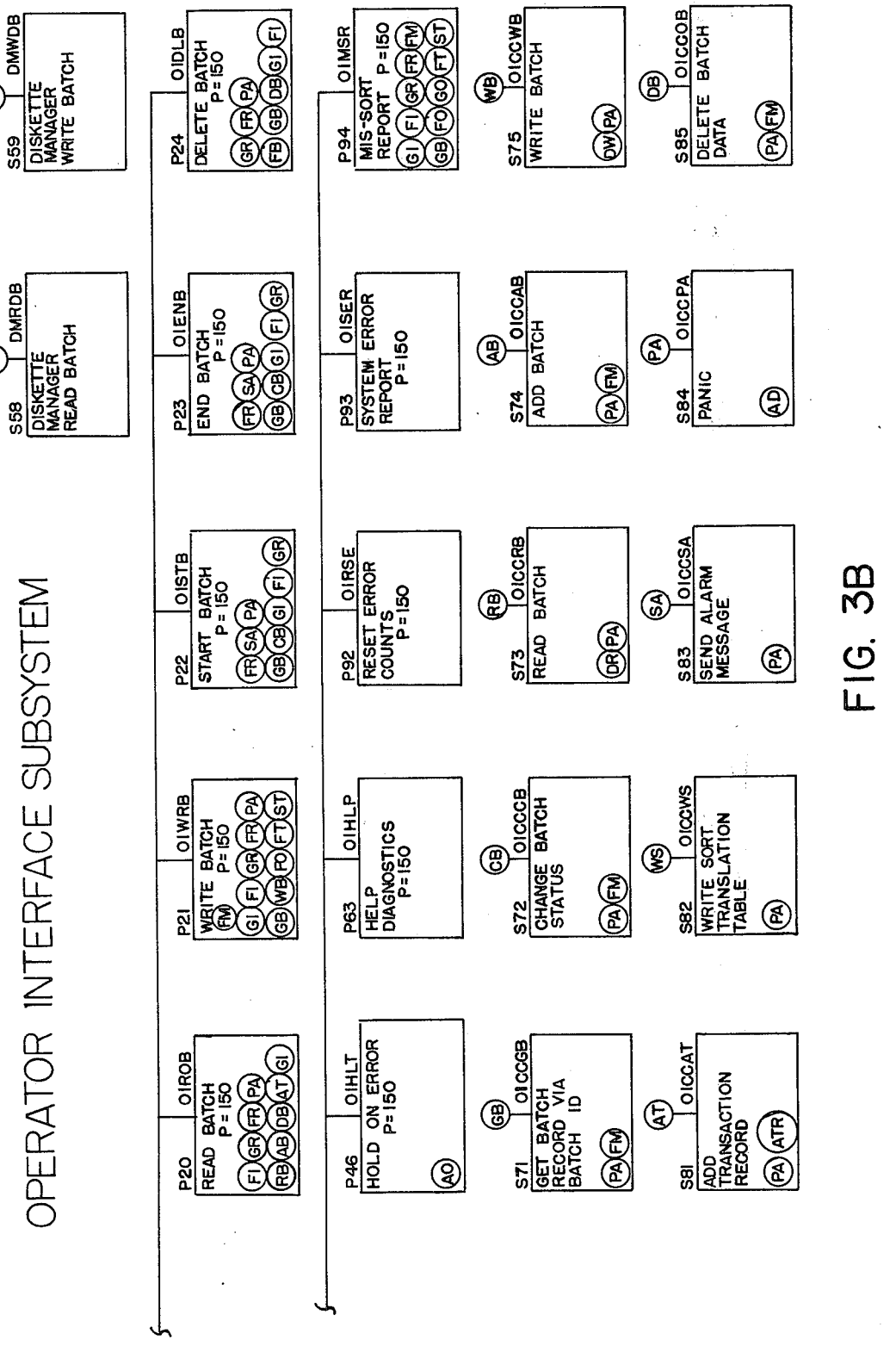

OPERATOR INTERFACE SUBSYSTEM
S58 DMRDB
DISKETTE MANAGER READ BATCH
S59 DMWDB
DISKETTE MANAGER WRITE BATCH
P20 OIROB
READ BATCH P=150
P21 OIWRB
WRITE BATCH P=150
P22 OISTB
START BATCH P=150
P23 OIENB
END BATCH P=150
P24 OIDLB
DELETE BATCH P=150
P46 OIHLT
HOLD ON ERROR P=150
P63 OIHLP
HELP DIAGNOSTICS P=150
P92 OIRSE
RESET ERROR COUNTS P=150
P93 OISER
SYSTEM ERROR REPORT P=150
P94 OIMSR
MIS-SORT REPORT P=150
S71 OICCGB
GET BATCH RECORD VIA BATCH ID
S72 OICCCB
CHANGE BATCH STATUS
S73 OICCRB
READ BATCH
S74 OICCAB
ADD BATCH
S75 OICCWB
WRITE BATCH
S81 OICCAT
ADD TRANSACTION RECORD
S82 OICCWS
WRITE SORT TRANSLATION TABLE
S83 OICCSA
SEND ALARM MESSAGE
S84 OICCPA
PANIC
S85 OICCDB
DELETE BATCH DATA

FIG. 3B

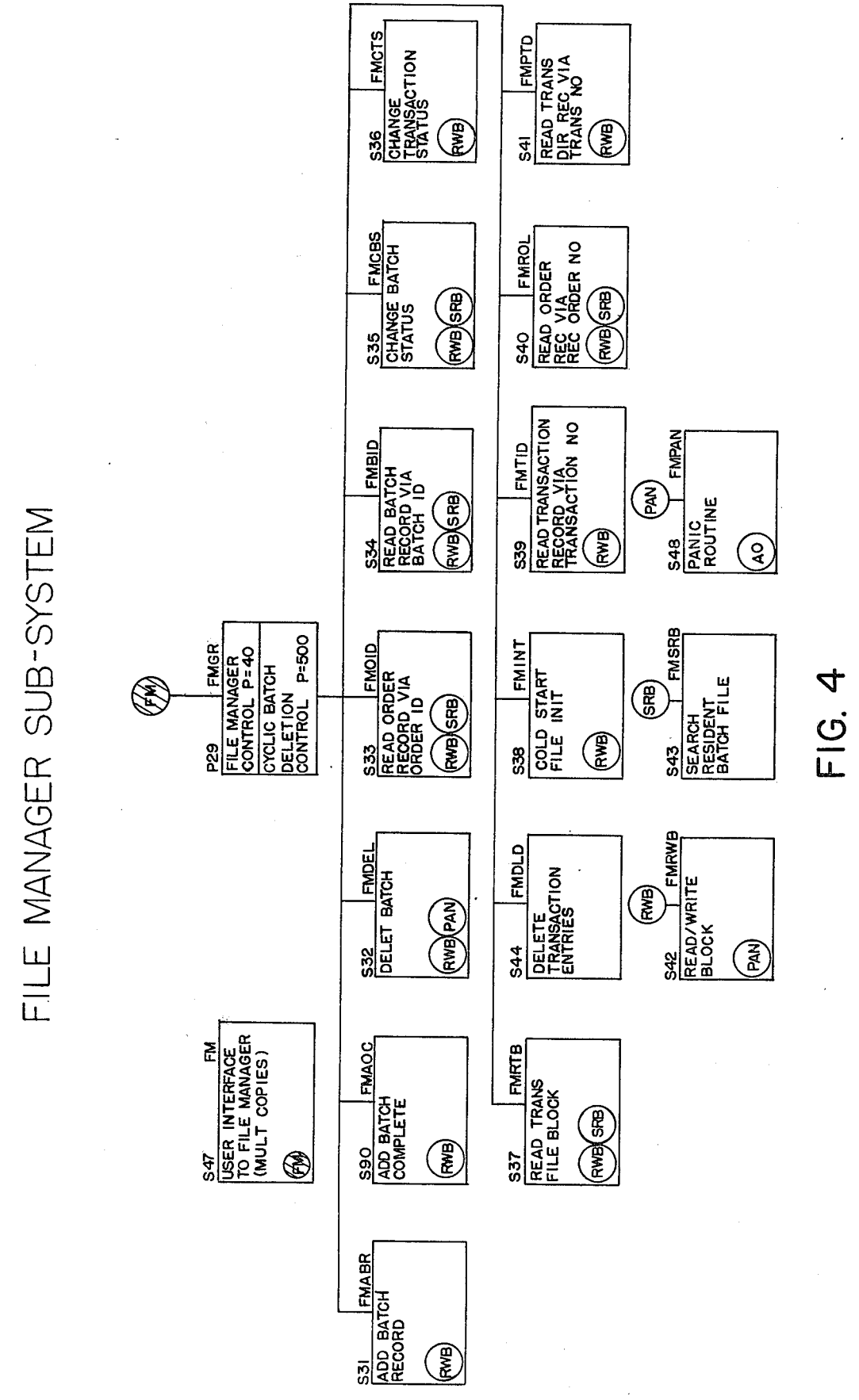

FILE MANAGER SUB-SYSTEM
FM
FMGR
P29
FILE MANAGER CONTROL P=40
CYCLIC BATCH DELETION CONTROL P=500
S47
USER INTERFACE TO FILE MANAGER (MULT COPIES)
FM
FMABR
S31
ADD BATCH RECORD
RWB
FMAOC
S90
ADD BATCH COMPLETE
RWB
FMDEL
S32
DELET BATCH
RWB PAN
FMOID
S33
READ ORDER RECORD VIA ORDER ID
RWB SRB
FMBID
S34
READ BATCH RECORD VIA BATCH ID
RWB SRB
FMCBS
S35
CHANGE BATCH STATUS
RWB SRB
FMCTS
S36
CHANGE TRANSACTION STATUS
RWB
FMRTB
S37
READ TRANS FILE BLOCK
RWB SRB
FMDLD
S44
DELETE TRANSACTION ENTRIES
RWB
FMRWB
S42
READ/WRITE BLOCK
PAN
FMINT
S38
COLD START FILE INIT
RWB
SRB
FMSRB
S43
SEARCH RESIDENT BATCH FILE
FMTID
S39
READ TRANSACTION RECORD VIA TRANSACTION NO
RWB
PAN
FMPAN
S48
PANIC ROUTINE
AO
FMROL
S40
READ ORDER REC VIA REC ORDER NO
RWB SRB
FMPTD
S41
READ TRANS DIR REC VIA TRANS NO
RWB

FIG. 4

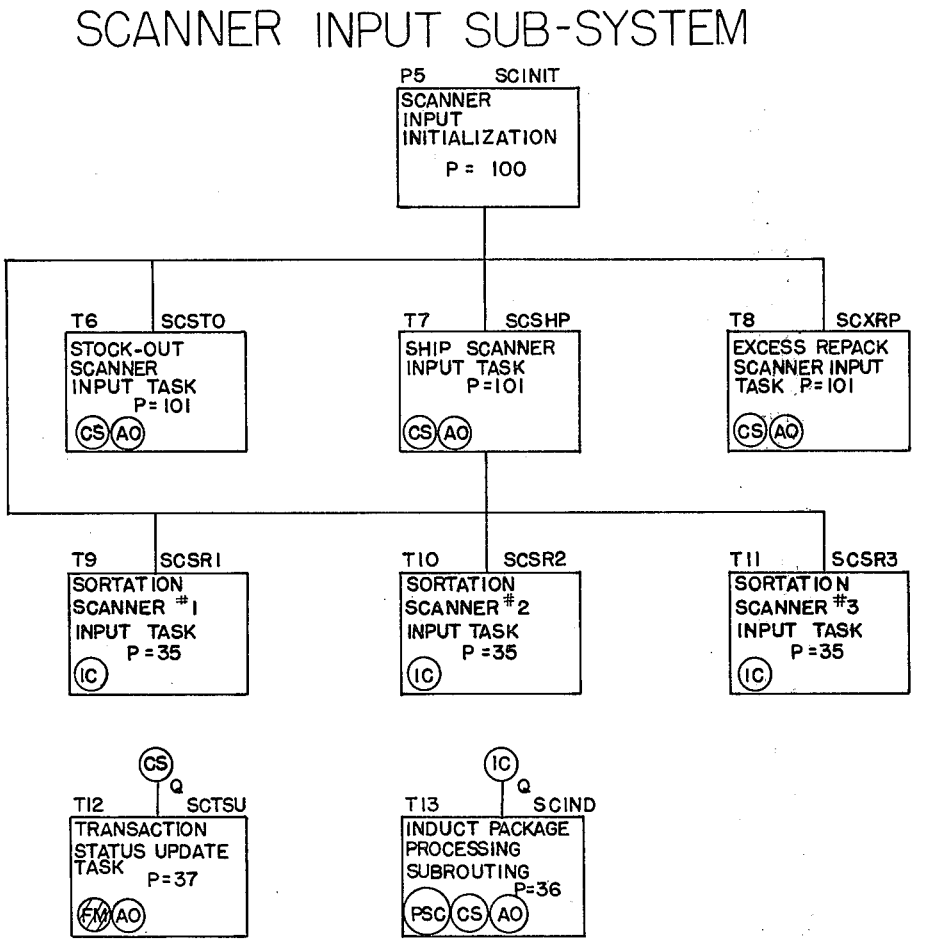

SCANNER INPUT SUB-SYSTEM
P5 SCINIT
SCANNER INPUT INITIALIZATION
P = 100
T6 SCSTO
STOCK-OUT SCANNER INPUT TASK
P=101
CS AO
T7 SCSHP
SHIP SCANNER INPUT TASK
P=101
CS AO
T8 SCXRP
EXCESS REPACK SCANNER INPUT TASK P=101
CS AO
T9 SCSR1
SORTATION SCANNER #1 INPUT TASK
P=35
IC
T10 SCSR2
SORTATION SCANNER #2 INPUT TASK
P=35
IC
T11 SCSR3
SORTATION SCANNER #3 INPUT TASK
P=35
IC
CS Q
T12 SCTSU
TRANSACTION STATUS UPDATE TASK
P=37
FM AO
IC Q
T13 SCIND
INDUCT PACKAGE PROCESSING SUBROUTING
P=36
PSC CS AO

FIG. 8

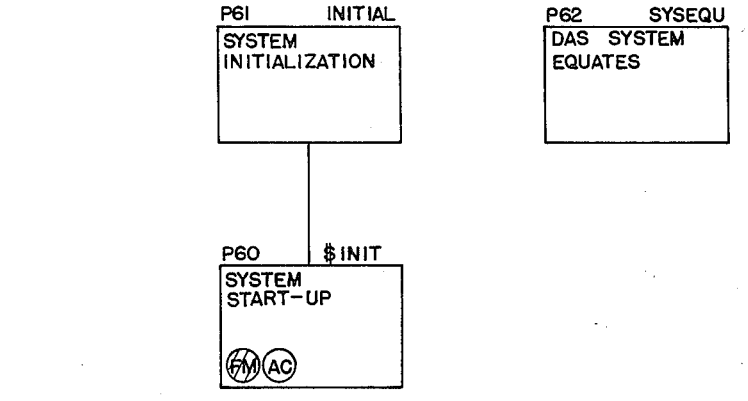

AUXILIARY FUNCTIONS SUB-SYSTEM
P61 INITIAL
SYSTEM INITIALIZATION
P62 SYSEQU
DAS SYSTEM EQUATES
P60 $INIT
SYSTEM START-UP
FM AC

FIG. 5

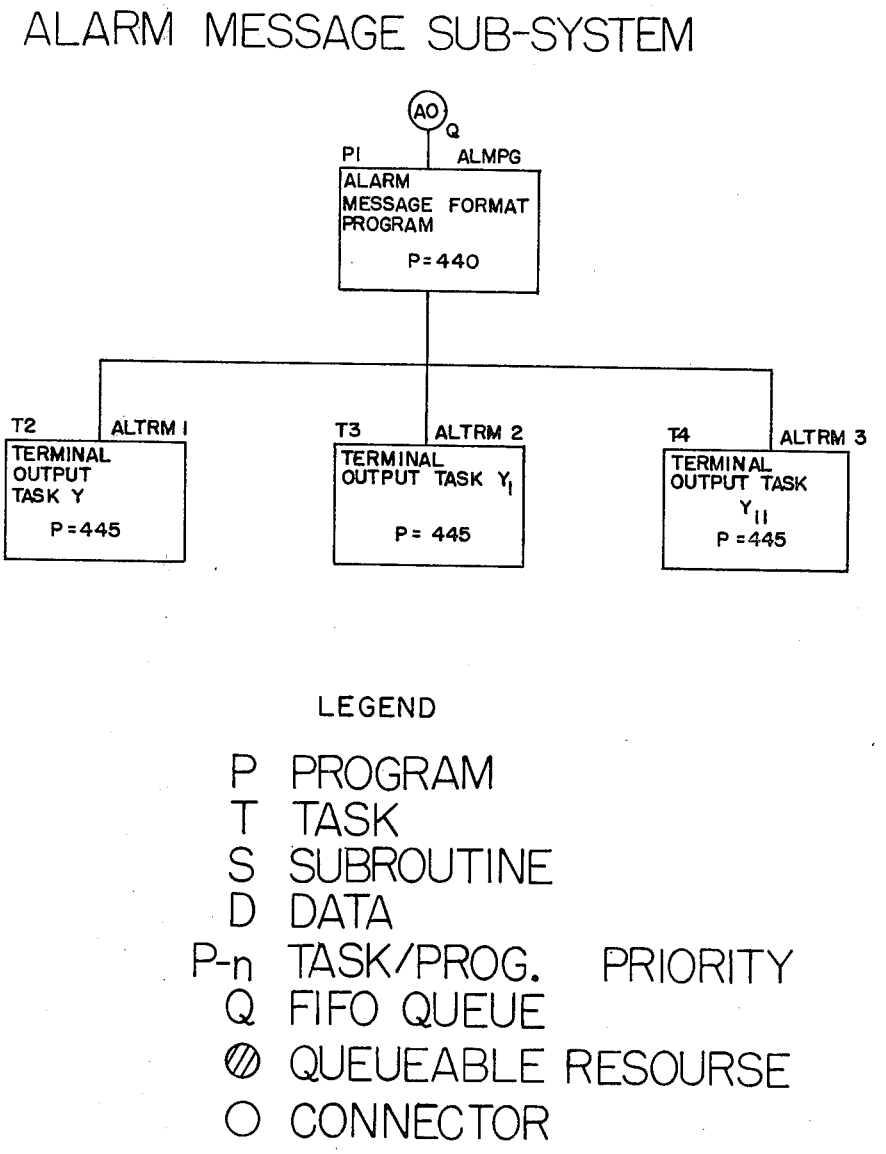

ALARM MESSAGE SUB-SYSTEM
AO
Q
PI
ALMPG
ALARM MESSAGE FORMAT PROGRAM
P=440
T2
ALTRM 1
TERMINAL OUTPUT TASK Y
P=445
T3
ALTRM 2
TERMINAL OUTPUT TASK Y_I
P= 445
T4
ALTRM 3
TERMINAL OUTPUT TASK Y_{II}
P=445
LEGEND
P PROGRAM
T TASK
S SUBROUTINE
D DATA
P-n TASK/PROG. PRIORITY
Q FIFO QUEUE
QUEUEABLE RESOURSE
CONNECTOR

FIG. 6

WAREHOUSING MONITOR AND CONTROL SYSTEM

BACKGROUND OF THE INVENTION

The present invention relates to a method and apparatus for monitoring and controlling the flow of articles in a warehouse or other installation.

There exists a great variety of systems in which orders are filled by, for example, a central warehouse where items to be shipped to a given destination are manually, semiautomatically, or automatically picked, sorted, and routed to a shipping dock for shipment to, for example, retail stores. Typically, labels identifying the shipping destination are applied to articles as they are picked which labels are subsequently employed at a sorting location for the diverting of articles to a particular shipping area of the installation. U.S. Pat. No. 4,181,947 illustrates a sorting system which can be employed with such a system.

Where, however, a warehouse system is of relatively large size and it is capable of simultaneously handling a great number of orders, maintaining accurate information as to the status of individual orders being filled as well as the operational status of the entire system, while maintaining a maximum throughput efficiency, is virtually impossible with the systems of the prior art.

SUMMARY OF THE PRESENT INVENTION

It is an object of the present invention to provide a method and apparatus for maintaining an accurate account of each transaction occurring within a warehousing system such that at any given time during an operating day, the status of each transaction and order being filled can be monitored and controlled. In order to achieve this objective, each transaction, which in the preferred embodiment of the invention pertains to a given article to be transferred from storage to a dispatching area, is assigned a unique identification number. This number is subsequently applied to an article as it is picked in the form of a machine readable coded label. The number is also entered into a control system memory, and as the article travels through the system, the label is read such that at any given time, the status of each transaction and the operation of the entire system is accurately known and can be controlled.

In the preferred embodiment of the invention, the system incorporates a computer controlled sorting and control system in which each article to be removed from storage and shipped to a destination, is assigned a unique code which identifies a single transaction within the system. Orders to be filled by batch picking and transfer of articles from storage to a given dispatch location at the warehouse, are grouped together in the computer memory to form batches of orders with the customer identification and discharge location being associated with each transaction number. By reading only the transaction number from a encoded label on an article at one or more locations within the system, the computer can continuously update the order status information and provide the operators of the system with current status information as well as provide control information to the sorting system employed. By providing operator interface circuits, reports can be generated either in soft or hard copies to monitor the operational status of each trasaction, order or batch of orders to facilitate the accurate and fast movement of articles within the system. Such information can be employed

to quickly identify and correct failures and breakdowns within the system or to rapidly reassign divert locations for the sorting system in the event of a breakdown of a sortation line.

These and other objects, features and advantages of the system will become apparent upon reading the following description thereof together with reference to the drawings in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a pictorial representation of a warehousing system embodying the present invention including an electrical circuit diagram in block form of the control system;

FIG. **2** is typical label associated with each article and identifying a unique transaction within the system;

FIGS. **3A** and **3B** are block diagrams showing the module layout of the software employed for the operator interface portion of the control system;

FIG. **4** is a block diagram of the module layout of the software for the file manager portion of the control system;

FIG. **5** is a block diagram of the module layout of the software for the auxiliary functions of the control system;

FIG. **6** is a block diagram of the module layout of the software for the alarm message portion of the control syste including the legend employed for FIGS. **3** through **8**;

FIG. **7** is a block diagram of the module layout of the software for the interface between the sorting control system and the main control system; and

FIG. **8** is a block diagram of the module layout of the software for the code reading portion of the control system of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** represents a physical installation as well as the control system located within the installation. At the upper portion of the figure above dashed line **10** there is shown a central control area corresponding typically to an office area of a customer's facility, while the portion below line **10** represents the actual warehouse installation including the sortation conveyer equipment, and in which the actual articles to be transferred from storage to a dispatching dock or the like are stored in a conventional warehousing storage system. Such a system may include, for example, tiers of vertically and horizontally arranged storage bins with access aisles extending between adjacent tiers permitting either manual, semiautomatic, or automatic picking of articles from the storage bins to be achieved through the use of stacker cranes or other picking systems. Articles removed from storage are placed on conveying systems which ultimately connect with a sorting conveyer. Not shown in FIG. **1** are the storage bins, access aisles, or the conveyers leading to the induction station and sorting conveyer system. In some installations, the warehouse will be physically separate from the central control area while in other installations the two locations may share a common building at separate areas. Having briefly described the environment in which the present invention pertains, a description of the overall system is now presented.

In FIG. **1**, the customer computer **12** typically will be a relatively large computer used by the customer for all

of its business operations. Such a computer may be, for example, a commercially available IBM System 3 type computer which is interfaced with a label printer 14 by means of a data link 13 for the printing of labels 20, such as shown in FIG. 2. The label printer can be any one of a number of dot matrix-type printers which are commercially available such as the Printronix Model 600. Also, interfaced with the customer computer 12, is a batch data transfer diskette memory 16 coupled to the computer by a data link 15 for the generation of diskettes containing information corresponding to one batch of orders. The unit 16 may be, for example, a commercially available IBM floppy diskette model 3540. The customer computer 12 is employed in connection with label printer 14 and disketter unit 16 through conventional programming techniques to arrange orders to any one of the several retail outlets of the customer in a logical picking sequence such that the articles can be batch picked. Thus, for example, if the customer is a large grocery chain, with several grocery stores located in the geographical area served by the warehouse, one batch of orders to be filled may include 50 different grocery items with the batch including orders for 15 different retail outlets. The orders are arranged such that each item which is common to the group of articles to be picked will be simultaneously picked by the warehousing picking system. Thus, the computer 12 will be programmed, for example, to print labels in order with all of the labels for example, pertaining to a case of a certain type of produce such as beans to be simultaneously picked for all of the orders. This organization provides for efficient batch picking of articles.

Assigned to each transaction, which corresponds to a single case of a given item, is a unique six digit number which is employed within the control system shown in FIG. 1, to uniquely identify that article such that its progress within the system can be monitored and controlled. This number is applied in machine readable form to the label 20 as seen in FIG. 2 in the form of a bar code 22 which occupies a large portion of the label. The label 20 also includes man readable information 24 at the bottom of the label indicating for example the product identification, pricing information, and shipment location for the customer to which the article being picked is to be transferred. Further, the label will include man readable indicia 25 indicting the storage location of the article within the warehousing system which is contained within the memory of computer 12. Also on label 20 is indicia 26, corresponding to the store number to which the article is to be shipped, indicia 27 showing the unique six digit number (which in the example shown in FIG. 2 is the number 254789); and finally, indicia 28 pertaining to the discharge chute of the sortation conveyor system to which the article is to be sent for shipment.

Each article to be picked corresponding to a single transaction thus receives a label 20 and each label printed by printer 14 in a given day has a unique number 22 as compared to all other labels printed. Information corresponding to the label information as well as additional information is simultaneously recorded on a floppy disc by unit 16 which includes the label information for a plurality of labels forming several orders forming, in turn, a single batch for processing within the control system. Each diskette therefore includes stored data corresponding to each label of a given batch including the six digit label number, the sort lane number, the store number, the product code, which is assigned by the customer to any particular type of product, and information pertaining to where the product is stored in the warehouse. Further, each diskette includes a header record which provides an identification of the particular batch, as well as a description of the batch which can be any 50 character description desired by the customer. Thus, each group of labels provided by printer 14 will have associated with them a single diskette corresponding to a batch of items to be picked. Each batch will typically include several orders, each of which include several transactions with each transaction having a single label associated with it. The system of the present invention is capable of processing simultaneously up to 10 batches with each batch having a capability of 35 orders per batch and total of 9,000 transactions for each batch. Typically, however, only one or two batches will be picked simultaneously with the remaining batch information contained in memory in either a pending status or possibly a completed status, depending upon the operational status of the picking and sorting of the batches. The generation of the labels, as well as the diskette in a batch picking sequence, with the exception of the utilization of a unique number assigned to and associated with each label and transaction is conventional and achieved by computer 12. The labels are manually carried to the warehouse as indicated by dashed line 21 as are the diskettes 23 generated and associated with each batch as indicated by dashed line 17.

Each diskette from the customer computer 12 generated by the diskette unit 16 is read by a batch data transfer diskette memory unit 30 associated with and coupled to a distribution audit system (DAS) computer 32 by means of a data link 31. The diskette unit 30 may comprise for example a commercially available IBM diskette unit model number 4964, while computer 32 may for example be an IBM model 4955D computer. The function of unit 30 is to read the information from the diskette hand carried from unit 16 into a multi-batch storage disc memory unit 34 coupled to computer unit 32 through data link 33. Unit 34 is capable of storing not only the control program for computer 32, but also the data stored on each of the diskettes associated with each batch of articles to be picked. Unit 34 may, for example, comprise a commercially available IBM 4962 disc storage unit. Hard copies of alarm, status, and control information are provided by a systems operations log printer 36 coupled to computer 32 by means of a data link 35. Printer 36 may, for example, comprise a Digital Equipment Corporation matrix printer type LA 120. A second report printer 38 is also coupled to computer 32 through data link 37 and is employed for providing a variety of status reports as discussed below and may be for example an IBM model 4974 matrix printer. Also coupled to computer 32 by means of a data link 39 is an operator terminal 40. Terminal 40 includes a CRT display 42 and a digital keyboard 43 and may be a commercially available IBM model 4979 display station. Several such operator terminals and printers may be positioned at different locations within the system as conveniently desired. Other interface inputs to the computer 32 include one or more bar code scanners 44 selectively coupled to the computer 32 by means of a switch 45 which either couples the scanners 44 which are Accusort model number M scanners to an input of computer 32 or label wand scanners 46 to the computer 32. The wand scanners 46 can be Accusort model number

4600 scanner for detecting the bar code **22** on labels **20** of FIG. **2**. Scanners **44** optically read labels on articles prior to transfer or induction onto the sortation conveyor **50** as shown by lines **53** in FIG. **1**. Also coupled to computer **32** by means of a data line **49** is a second wand scanner **48** which is employed for scanning stock-out labels as described below and which can also be an Accusort model 4600 wand scanner. An additional wand scanner (not shown) is employed for hand scanning excess repack labels as also described below.

The DAS computer **32** interfaces with a programmable sort control (PSC) computer **60** by means of a data interface coupling **62**. Associated with the PSC computer **60** is a backup keyboard **64** coupled thereto by means of a serial data link **65**, and a conveyer control circuit **66** coupled thereto by means of a data link **67**. A plurality of article detecting photo cells **68** are spaced along the sorting conveyer **50** downstream of each article diverter **51** for detecting whether an article has been properly diverted to its assigned discharge or shipping chute **52** associated with the sorting conveyer **50**. Further, a line full photo cell detector **69** is also coupled to the PSC computer **60** and is associated with each of the discharge chutes **52** such that a control signal is applied to computer **60** in the event any of the discharge chutes are filled with articles and thereby requiring a different discharge chute to be assigned to a particular order being sorted and staged for shipment.

At the input end of the sorting conveyer **50**, there is provided an induction station **54** for receiving articles from several feeder conveyers (not shown) extending throughout the warehousing system and converging the articles onto the sorting conveyer **50**. The PSC computer and its associated interface and control circuits to provide the induction and sorting of articles is described in detail in the above identified U.S. Pat. No. 4,181,947, the disclosure of which is incorporated herein by reference.

Sorting conveyer **50** terminates in a T **55** leading to either an error chute **56** or a recirculation loop **58** such that articles not sorted can either be recirculated through the sortation control system as selectively controlled as described below or fed to a storage area associated with the error chute for manual attention to a nonreadable label.

The control of the hardware elements shown in FIG. **1** corresponding to the control system of the present invention is achieved by the programming of computer **32** which program is stored in memory **34** and which is organized according to the module layout chart shown in FIGS. **3** through **8**. With reference to these figures, FIGS. **3A** and **3B** disclose the layout of the programming for the interface between computer **32** and printer **38** and computer **32** and the operator terminal **40**. The programming represented by FIG. **4** pertains to the internal handling of data between computer **32** and memory **34** as does the programming diagrammed in FIG. **5**. FIG. **6**, relates to the alarm messages provided by printer **36** and accordingly the programming for providing such alarms. The programming represented by FIG. **7** pertains to the programming which interfaces computer **32** with computer **60**. Finally, the programming represented by FIG. **8** pertains to the intercoupling of computer **32** with the scanning wands **46, 48**, and the bar code scanners **44**. The module layout of the program as represented by FIGS. **3** through **8** is directly correlated with the pseudo code and structural English printout of the software organization disclosed in appendix A which is attached hereto, made part of the specification and incorporated herein by reference. In FIGS. **3** through **8**, each block represents a module which performs a specific operation within the system. Each block is identified by the letters P, S, T, or D (described in the legend accompanying FIG. **6**) followed by a number identifying a particular portion of the program employed. In addition, each block includes a label name constituting an acronym which is correlated with the programming information disclosed in appendix A. As can be seen, for example in FIGS. **3A** and **3B**, many of the modules call up additional modules indicated by an encircled two letter connector. Thus, for example, the panic subroutine shown in FIG. **3B** which serves to stop the program in the event of a major failure, is identified by the two letters PA which is called up by many of the modules in the event the module is unable to perform its particular programming function. The programming comprising the source code for the PSC computer is also incorporated herein by reference as appendix B which is attached hereto and made a part of this specification.

Having described the hardware and the broad correlation between the program organization and individual elements of the hardware, a description of the operation of the system to achieve its desired monitoring and control of each transaction, as well as the various status reports available employing the unique transaction number identification of each article to be picked, is now presented.

Before any of the articles to be picked can be recognized and sorted by the DAS control system, the batch and transaction information must first be transferred to the DAS system. This is achieved by the operator control terminal **40** entering a READ command which transfers the information from a diskette **23** placed in unit **30** into the computer memory **34**. The program for this is shown in FIG. **3B** indicated as READ BATCH. Once the data base has been constructed by the operator for each of the diskettes being processed during a day's operation, the actual picking, sorting, and monitoring functions available are commenced by a START command which changes the batch status from pending to active and enables the DAS computer to sort the product.

The picker applies labels to the correct products and places the unique six digit bar code containing label on each article as they are picked and transferred onto conveyers within the system which feed to the induction station **54**. Each article is scanned by scanner **44** for its unique number and this information is supplied to the DAS computer which correlates it with sort location information for that number, and communicates with the PSC computer the sort destination assigned to the particular transaction. Depending upon the selected mode of operation, the article can be stopped and hand wanded by wand **46**, recirculated through recirculation loop **58** or shunted into the error chute **56** upon failure of scanner **44** to read the article's number. The status of the article is changed from "not-picked" to "in-sortation" and the PSC performs the physical sorting of the article and informs the DAS computer of the actual status of the package on the sorting conveyer. When the label is transferred to its assigned destination chute **52**, (FIG. **1**) the status of the article is changed from "in-sortation" to "staged for shipment". If the article is unable to sort to its assigned sorting lane, the PSC informs the DAS of the actual disposition of the package.

Once all of the articles of a batch have been picked and sorted, including articles for several different orders typically, the operator who monitors the progress of the batch picking as described below, enters an END command in the operator terminal **40** which changes the batch status to "complete". At this time, the operator can enter a WRITE command in terminal **40** which generates a new diskette which is returned to the customer computer and which has information as to the completed status of each transaction in the batch which can be employed by the customer for inventory, billing, and other functions such the printing of shipping manifests and other documents. Once the batch has been processed by the DAS control system, it can be deleted from the DAS data base, memory **34** by entering a DELETE command.

There are a variety of system batch and sortation control commands which are employed by the system of the present invention. The system control commands include a listing of system commands for status reports which is identified by the command HELP entered in keyboard **43** to generate a listing of available reports on CRT **42**.

By entering the command ERROR a system error report as shown in appendix A is generated by printer **38** which permits the operator to visualize the number of missorts and transfer failures and take corrective action. Another system control command is RESET which is employed by the operator for resetting error counts which may correspond to missorts where an item is discharged to the wrong discharge chute **52** inadvertently, or transfer failures where the PSC computer fails to divert an article to any of the desired discharge chutes. The final system control command is the time setting function which is employed to provide a real time display or printout to all of the various reports and is entered by entering the command $ T employed by the operator to set the system time.

In addition to the system control commands, there are several batch control commands which are entered by keyboard **43** some of which have been previously discussed. The first of these, is the READ command which effects the reading of the customer's diskette **23** by the DAS computer diskette unit **30** reading the information into the memory **34**. The START command starts the batch processing monitor and control while an END command as previously discussed, ends the batch processing. The WRITE command causes the diskette to be reprogrammed with the updated status of the orders and transactions therein reported back to the main customer computer through a reprogrammed diskette while a DELETE command then is employed to delete the completed batch information from memory **34**. The ASSIGN command is employed for example in the event a given product is out of stock and with the batch processing system, this could affect several transactions associated with several different orders. To provide a quick update of the status of the particular product and therefore the transactions in which it is involved, the operator can assign to the transaction numbers affected, a stockout status so that this condition can be instantaneously displayed at any of the report printers or operator terminals once the condition has been reported.

Finally, there are several sortation control commands entered into keyboard **43** to control the sorting of articles. One mode of operation is identified and controlled by the command HOLD. With the HOLD command, the induction station **54** is stopped when a no-read occurs on scanner **44** and the operator must manually scan the label using wand **46** (FIG. **1**) to read the label. He does that by actuating switch **45** such that the wand **46** output information is supplied to the DAS computer **32**. The induction station **54** then restarts, and the article is sorted.

With the RECIRC command, articles with labels not read by scanner **44** are automatically directed to recirculation lane **58**. Upon initial startup of the system, the command OPEN is employed to open the communications interface **62** between the DAS computer **32**, the PSC computer **60**. The command REROUTE is employed to assign an alternate sortation line for articles with a common destination contained within all batches being processed. This is employed for example, in the event a sort line **52** has a mechanical failure and it is temporarily shut down. The command RESTORE will automatically reassign rerouted articles not yet processed to the original sort line once it has been repaired.

In addition to the interface between the operator terminal **40** and the system employing the above identified commands, the system will automatically provide two different types of printouts available to management and control personnel for the monitoring and operational control of the system. The first type of printout is provided by the system's operational log printer **36**, and provides indications of the status of the system as well as missorts and the like. The following is a typical printout during a short period of time representing a variety of operational conditions of the system.

04/01/80 14:30:34 System Started
04/01/80 14:30:46 PSC Communications Line Open
04/01/80 14:30:59 Start of Batch T02
04/01/80 14:31:04 Hold on Scanner Error
04/01/80 14:31:43 Scanner Error At Induction #1 Scanner
04/01/80 14:31:54 Transaction 984561 at Induction #1 Scanner Not On File
04/01/80 14:32:17 Transaction 411589 at Induction #1 Scanner Out of Batch
04/01/80 14:32:44 Sortation Line 1 Full
04/01/80 14:32:49 Transaction 598402 Missorted To Sort Line 3
04/01/80 14:33:24 Unsuccessful Transfer At Sort Line 1 for Transaction 632368
04/01/80 14:33:27 Scanner Error at Induction #2 Scanner
04/01/80 14:34:29 Transaction 581419 Has Been Lost From Tracking
04/01/80 14:34:46 Recirculate On Scanner Error
04/01/80 14:35:37 End of Batch T02
04/01/80 14:36:08 Transaction 581419 on Sort Line 2 Not On File
04/01/80 14:36:27 PSC Communications Failure
00/00/00 00:00:10 Power Fail/Restart

As can be seen from the above report, the status of the system and its communications with the PSC is indicated as well as events such as scanning errors at the induction station, sortation lines being full, as detected by the line full photo detector **69** (FIG. **1**), missorts and the like. This record is automatically printed without a specific operator command such that monitoring personnel can take corrective action as required. Naturally, many of the messages printed simply indicate the status of a particular batch, for example, at 14:30:59, batch **T02**

was beginning to be processed and at 14:35:37, batch T02 was completed.

The second type of printed information, is specifically requested results from the entry of the system control command HELP, which provides a variety of selectable status reports. These reports are called up using keyboard **44** selectively as indicated by the following table:

Schedule—Work Schedule Batches
Batch—Batch/Order Status
Order—Order/Transaction Status
Transact—Transaction Status
Missort—Missort Exceptions
Route—Sort Line Translations

The reports generated by the entry of the commands at keyboard **43** are provided by the report printer **38** (FIG. **1**) and an example of each such report is shown in the attached appendix A which also includes a data dictionary which identifies the manner in which the modules shown in FIGS. **3** through **8** are called up in the program. Basically each report provides a complete current status of batches being processed by the system, orders included in each batch, and particular transactions included in each order. Further, these reports provide an indication as to the percentage of completion, the number of items which are out of stock, and an indication of items not picked, missorted, or in processing. This information is used by the supervisory personnel to monitor on a regular basis the operation of the picking and sorting system, as well as to provide particular commands through terminal **40** for effecting the efficient operation of the system.

In the event that a stockout condition exists where items to be picked are not available, this information can, as previously indicated, be entered through an operator terminal, but also wand **48** can be employed to scan the labels which have not been applied to articles to be picked and the system automatically assigns a stockout status to the transactions and permits the batch to be completed noting the stockout status in the batch status report. An additional wand scanner (not shown) is provided for excess labels to cover the situation where incomplete case loads of materials are ordered and the efficiency of the picker in filling the cartons is unknown. To cover this situation, several additional labels are provided in the event that a greater number of cartons are required due to the packing efficiency. In the event these extra labels are not needed, they are scanned by the additional wand scanner to maintain accurate status information as to all labels and therefore all transactions within the system.

With the system of the present invention, therefore, high speed article processing can be achieved and in the environment of the preferred embodiment a picking and sorting operation for warehouses is provided with a control system for efficiently monitoring the current operational status of a plurality of batch picks each including several orders, in turn, including several transactions. By assigning each transaction a unique identifying number which is physically associated with the article to be picked in the form of a machine readable label, and which is entered into the control system memory in the form of stored data, the current status of each transaction, order, and batch can be continuously monitored and controlled. The resultant system results in a more efficiently operated system with a higher throughput than previously achievable through conventional picking systems.

It will become apparent to those skilled in the art that various modifications to the preferred embodiment of the invention described and disclosed herein can be made without departing from the spirit or scope of the invention as defined by the appended claims.

```
COPYRIGHT    1979    RAPISTAN INC.    GRAND RAPIDS    MICHIGAN                00000010
                                                                              00000020
THIS SOFTWARE IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE           00000030
ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF              00000040
RAPISTAN'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEMS, EXCEPT AS          00000050
MAY BE OTHERWISE PROVIDED IN WRITING BY RAPISTAN INC.                         00000060
                                                                              00000070
THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE          00000080
AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY RAPISTAN INCORPORATED.         00000090
                                                                              00000100
RAPISTAN ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS          00000110
SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY RAPISTAN.                      00000120


*****************************************************************************00000010
*                                                                             00000020
* AUXILIARY FUNCTIONS DATA DICTIONARY                                         00000030
*                                                                             00000040
*****************************************************************************00000050
                                                                              00000060
I/O #     DESCRIPTION                                                         00000070
                                                                              00000080
01AX      CONTROL DATA SET FORMAT                                             00000090
02AX      START-UP CODE                                                       00000100
03AX      SYSTEM INITIALIZATION BATCH TABLE                                   00000110
04AX      SYSTEM INITIALIZATION BUFFER IN USE FLAG                            00000120
05AX      SYSTEM INITIALIZATION DISK BUFFERS                                  00000130
06AX      SYSTEM INITIALIZATION ORDER TABLE                                   00000140
07AX      SYSTEM INITIALIZATION BATCH ERROR FLAG                              00000150
08AX      SYSTEM INITIALIZATION DISK ERROR FLAG                               00000160
09AX      SYSTEM CONFIGURATION PROGRAM DIALOGUE                               00000170
```

```
*****************************************************************************00000180
*                                                                           00000190
* AUXILIARY FUNCTIONS DATA REFERENCES                                       00000200
*                                                                           00000210
*****************************************************************************00000220
                                                                            00000230
01AX     AXINT                                                              00000240
02AX     $INIT,$INITIAL                                                     00000250
03AX     $INITIAL                                                           00000260
04AX     $INITIAL                                                           00000270
05AX     $INITIAL                                                           00000280
06AX     $INITIAL                                                           00000290
07AX     $INITIAL                                                           00000300
08AX     $INITIAL                                                           00000310
09AX     AXCNFG                                                             00000320


*****************************************************************************00000330
*                                                                           00000340
* AUXILIARY DATA DICTIONARY ENTRIES                                         00000350
*                                                                           00000360
*****************************************************************************00000370
                                                                            00000380
01AX     CONTROL DATA SET FORMAT FOR AXINT.P                                00000390
                                                                            00000400
         PROGRAM AXINT IS USED TO INITIALIZE DATA SETS TO A PREDEFINED      00000410
         HEX VALUE. IT'S OPERATION IS CONTROLLED BY INITIALIZ-              00000420
         ATION COMMANDS IN A CONTROL DATA SET. THESE COMMANDS               00000430
         CAN BE CREATED USING THE TEXT EDITOR WITH TAB SETS AT              00000440
         10 AND 20. THE COMMAND FORMAT IS AS FOLLOWS:                       00000450
                                                                            00000460
         COL.                                                               00000470
         1.....8  10....15  20....22                                        00000480
          DSNAME    VOLSER    VALUE                                        00000490
                                                                            00000500
         DSNAME: DATA SET MEMBER NAME                                       00000510
                                                                            00000520
         VOLSER: VOLUME ID WHERE DATA SET IS LOCATED                        00000530
                                                                            00000540
         VALUE:  THE DECIMAL VALUE OF A HEX BYTE YOU WANT TO                00000550
                 INITIALIZE THE FILE TO. FOR INSTANCE                       00000560
                 64 = X'40'                                                 00000570
                  0 = X'00'                                                 00000580
                255 = X'FF'                                                 00000590
                                                                            00000600
         NOTE: ALL PARAMETERS MUST BE SPECIFIED.                            00000610
                                                                            00000620
         THE LAST COMMAND MUST BE AN 'END' STARTING IN COL. 1.              00000630
                                                                            00000640
02AX     START-UP CODE                                                      00000650
                                                                            00000660
         THE EDX OPERATING SYSTEM PASSES A PARAMETER TO THE                 00000670
         INITIALIZATION PROGRAM NAMED "$INITIAL." IF $INITIAL HAS           00000680
         A PARAMETER DEFIND IN ITS PROGRAM STATEMENT. (PARM=1)              00000690
         THIS PARAMETER CONTAINS A START-UP CODE INDICATING THE             00000700
         TYPE OF START-UP BEING PERFORMED. THE START-UP CODE                00000710
         VALUES FOLLOW:                                                     00000720
                                                                            00000730
                   '0' = MANUAL IPL                                         00000740
                   '1' = POWER FAIL/RESTART                                 00000750
                                                                            00000760
         THIS CODE IS SAVED IN $SYSCOM BY PROGRAM $INITIAL AND              00000770
         CHECKED BY THE SYSTEM START-UP PROGRAM.                            00000780
                                                                            00000790
03AX     SYSTEM INITIALIZATION BATCH TABLE                                  00000800
                                                                            00000810
         THIS TABLE IS DEFINED WITHIN MODULE $INITIAL.                      00000820
         IT'S PURPOSE IS TO HOLD THE BATCH ID, RELATIVE BATCH,              00000830
         AND NUMBER OF TRANSACTIONS WITHIN THE BATCH, FOR ALL               00000840
         BATCHES WITH A STATUS OF PENDING, ACTIVE OR COMPLETE.              00000850
         THIS TABLE IS BUILT AT SYSTEM START UP TIME AND WILL               00000860
         BE USED TO UPDATE THE ORDER RECORDS FOUND IN THE ORDER             00000870
         DISK FILE.                                                         00000880
                                                                            00000890
04AX     SYSTEM INITIALIZATION BUFFER IN USE FLAG.                          00000900
                                                                            00000910
```

```
          THIS FLAG IS DEFINED WITHIN MODULE $INITIAL.                    00000920
          THE FLAG INDICATES WHICH OF THE TWO DISK I/O BUFFERS            00000930
          WAS LAST USED. WHEN READING THE TRANSACTION FILE,               00000940
          MODULE $INITIAL WILL ALTERNATE THE USE OF THESE BUFFERS.        00000950
                                                                          00000960
05AX      SYSTEM INITIALIZATION DISK BUFFERS.                             00000970
                                                                          00000980
          THERE ARE TWO DISK BUFFERS DEFINED WITHIN MODULE $INITIAL.      00000990
          EACH BUFFER IS CAPABLE OF HOLDING 50 DISK SECTORS.              00001000
          THEIR USE IS DEFINED IN 04AX.                                   00001010
                                                                          00001020
06AX      SYSTEM INITIALIZATION ORDER TABLE.                              00001030
                                                                          00001040
          THIS TABLE IS DEFINED WITHIN MODULE $INITIAL.                   00001050
          IT CONTAINS COUNTS OF TRANSACTIONS BY THEIR STATUS              00001060
          FOR ALL ORDERS IN THE BATCH BEING PROCESSED. THIS               00001070
          TABLE IS ONLY USED AT SYSTEM START UP TIME TO UPDATE            00001080
          THE ORDER RECORDS ON DISK.                                      00001090
                                                                          00001100
07AX      SYSTEM INITIALIZATION BATCH ERROR FLAG.                         00001110
                                                                          00001120
          THIS FLAG IS DEFINED WITHIN MODULE $INITIAL.                    00001130
          WHEN SET, THIS FLAG INDICATES THAT AN UNUSED                    00001140
          RECORD WAS FOUND IN THE TRANSACTION FILE BEFORE                 00001150
          ALL TRANSACTIONS IN THE BATCH HAVE BEEN PROCESSED.              00001160
                                                                          00001170
08AX      SYSTEM INITIALIZATION DISK ERROR FLAG.                          00001180
                                                                          00001190
          THIS FLAG IS DEFINED WITHIN MODULE $INITIAL.                    00001200
          THIS FLAG IS SET WHENEVER A DISK I/O ERROR                      00001210
          HAS OCCURRED DURRING THE UPDATE OF THE ORDER                    00001220
          FILE AT SYSTEM START UP TIME.                                   00001230
                                                                          00001240
09AX      SYSTEM CONFIGURATION PROGRAM DIALOGUE                           00001250
                                                                          00001260
          WHEN USING THE SYSTEM CONFIGURATION UTILITY, PARAMATERS         00001270
          THAT WILL REMAIN THE SAME AS DISPLAYED DO NOT NEED TO BE        00001280
          ENTERED. JUST PRESSING THE "ENTER" OR "RETURN" KEY WILL         00001290
          INDICATE NO CHANGE. WHEN SPECIFYING MESSAGE ROUTING, ALL        00001300
          LOG DEVICES SHOULD BE ENTERED ON THE SAME LINE, SEPARATED       00001310
          BY SPACES OR CAMMAS. A ZERO (0) IN THE FIRST POSITION IN-       00001320
          DICATES THAT THE MESSAGE IS NOT TO BE PRINTED ON ANY LOG        00001330
          DEVICE.                                                         00001340
                                                                          00001350

**************************************************************************00000010
*                                                                        00000020
* SCANNER DATA DICTIONARY                                                00000030
*                                                                        00000040
**************************************************************************00000050
                                                                          00000060
I/O #              DESCRIPTION                                          00000070
                                                                          00000080
01SC     SCSTFLG   SCANNER STOP FLAG                                     00000090
02SC               INITIALIZATION                                       00000100
04SC               SCANNER DATA TRANSMISSION                            00000110
05SC     STUPBUFQ  STATUS UPDATE BUFFER QUEUE                            00000120
06SC               SCANNER ID                                           00000130
08SC     TSTATUPQ  TRANSACTION STATUS UPDATE QUEUE                       00000140
09SC               RECIRCULATION DESTINATION                            00000150
10SC     STOPEVT   SCANNER PROGSTOP EVENT                                00000160
11SC               ERROR ON READ                                        00000170
12SC               PASSING INFORMATION TO SCIND                         00000180
13SC               ERROR DESTINATION                                    00000190
14SC     STPNR     STOP ON NO READ FLAG                                 00000200
15SC               SLOW DOWN EVENT                                      00000210
16SC     TSTATUPR  RESERVE STATUS UPDATE QUEUE                          00000220
17SC     STUPBUFR  RESERVE STATUS UPDATE BUFFER QUEUE                    00000230
18SC     SCINDEV   PACKAGE PROCESSING TASK EVENT                         00000240
19SC     SCTSUEV   STATUS UPDATE TASK EVENT                              00000250
20SC     INDQUEBF  INDUCTION BUFFER QUEUE                                00000260
21SC     INDQUE    INDUCTION QUEUE                                       00000270
22SC               NOT USED                                             00000280
23SC               ROUTING OF ERROR CONDITIONS                          00000290
                                                                          00000300
```

```
*****************************************************************  00000310
*                                                                    00000320
* SCANNER DATA REFERENCES                                            00000330
*                                                                    00000340
*****************************************************************  00000350
                                                                     00000360
I/O #     MODULES REFERENCING                                        00000370
                                                                     00000380
01SC      SCINIT,SCSR1,SCSR2,SCSR3,SCSTO,SCSHP,SCXRP,SCSTU           00000390
02SC      SCINIT,SCSTO,SCSHP,SCXRP,SCTSU,SCIND                       00000400
03SC      NOT USED                                                   00000410
04SC      SCSTO,SCSHP,SCXRP,SCSR1,SCSR2,SCSR3                        00000420
05SC      SCSTO,SCSHP,SCXRP,SCTSU,SCIND                              00000430
06SC      SCSTO,SCSHP,SCXRP,SCTSU,SCIND,SCSR1,SCSR2,SCSR3,AXCNFG     00000440
07SC      NOT USED                                                   00000450
08SC      SCSTO,SCSHP,SCXRP,SCTSU,SCIND                              00000460
09SC      SCIND,AXCNFG                                               00000470
10SC      SCINIT                                                     00000480
11SC      SCSR1,SCSR2,SCSR3,SCIND,OIHLT,OIREC                        00000490
12SC      SCIND,SCSR1,SCSR2,SCSR3                                    00000500
13SC      SCIND,AXCNFG                                               00000510
14SC      SCSR1,SCSR2,SCSR3,OIHLT,OIREC                              00000520
15SC      SCIND,SCTSU,PSOC                                           00000530
16SC      SCIND,SCTSU                                                00000540
17SC      SCIND,SCTSU                                                00000550
18SC      SCIND,SCSR1,SCSR2,SCSR3,SCINIT                             00000560
19SC      SCTSU,SCIND,SCINIT,SCSTO,SCSHP,SCXRP                       00000570
20SC      SCIND,SCSR1,SCSR2,SCSR3                                    00000580
21SC      SCIND,SCSR1,SCSR2,SCSR3                                    00000590
22SC      NOT USED                                                   00000600
23SC      SCIND                                                      00000610
                                                                     00000620


*****************************************************************  00000630
*                                                                    00000640
* SCANNER DATA DICTIONARY ENTRIES                                    00000650
*                                                                    00000660
*****************************************************************  00000670
                                                                     00000680
01SC      SCANNER STOP FLAG                                          00000690
            (SCSTFLG)                                                00000700
                                                                     00000710
          THIS FLAG IS INITIALIZED TO 0 BY THE SCANNER INITIALIZATION 00000720
          PROGRAM. IT IS CHECKED BY ALL OF THE SCANNER TASKS. WHEN A 00000730
          FATAL ERROR IS ENCOUNTERED THE FAILING TASK WILL SET THIS  00000740
          FLAG TO -1. THIS WILL BRING DOWN THE SCANNER SUB-SYSTEM.   00000750
                                                                     00000760
02SC      THE MAIN SCANNER TASK (SCINIT) WHEN LOADED WILL PICK UP ALL 00000770
          SYSCOM ADDRESSES NEEDED FOR THE SCANNER SUB-SYSTEM. THE OT- 00000780
          HER SCANNER TASKS WHEN ATTACHED WILL RETRIEVE THESE ADDRES- 00000790
          SES AND PLACE THEM IN THE APPROPRIATE INSTRUCTIONS THROUGH  00000800
          THE USE OF THE 'P' PARAMETERS.                             00000810
                                                                     00000820
04SC      SCANNER DATA TRANSMISSION                                  00000830
          (FORM OF DATA BROUGHT IN)                                  00000840
                                                                     00000850
          THE DATA READ FROM THE SCANNER WILL BE SIX EBCDIC DIGITS   00000860
          FOLLOWED BY A LINE FEED AND A CARRIAGE RETURN.             00000870
                                                                     00000880
05SC      STATUS UPDATE BUFFER QUEUE                                 00000890
                                                                     00000900
          STUPBUFQ  DEFINEQ   COUNT=50,SIZE=16                       00000910
                                                                     00000920
          STUPBUFQ IS A QUEUE THAT CONTAINS BUFFERS TO BE USED TO    00000930
          STORE SCANNER ID AND TRANSACTION NUMBER. THIS INFORMATION  00000940
          IS PUT IN BY THE SCANNER HANDLING ROUTINES (SCSTO, SCSHP,  00000950
          SCXRP, SCIND). THIS INFORMATION IS USED BY SCTSU TO CHANGE 00000960
          THE STATUS OF THE REFERENCED TRANSACTIONS.                 00000970
                                                                     00000980
          TO GET A BUFFER USE:                                       00000990
                                                                     00001000
                    FIRSTQ    STUPBUFQ,LOC,EMPTY=ROUTINE      ,WHERE: 00001010
                                                                     00001020
          LOC     = LABEL ON A WORD TO RECEIVE THE BUFFER ADDRESS.   00001030
          ROUTINE = NAME OF INSTRUCTION TO BRANCH TO IF STUPBUFQ     00001040
                    IS EMPTY.                                        00001050
                                                                     00001060
```

```
TO RETURN A BUFFER TO THE QUEUE USE:                                    00001070
                                                                        00001080
          NEXTQ     STUPBUFQ,LOC,FULL=ROUTINE      ,WHERE:            00001090
                                                                        00001100
LOC     = LABEL ON THE WORD THAT CONTAINS THE ADDRESS OF                00001110
          THE BUFFER.                                                   00001120
ROUTINE = NAME OF THE INSTRUCTION TO BRANCH TO IF STUPBUFQ              00001130
          IS FULL.                                                      00001140
                                                                        00001150
06SC    SCANNER ID'S                                                    00001160
                                                                        00001170
SCANNER ID'S USED IN THE SCANNER SUB-SYSTEM ARE:                        00001180
                                                                        00001190
ID        SCANNER                                                       00001200
                                                                        00001210
 1        INDUCTION #1                                                  00001220
 2        INDUCTION #2                                                  00001230
 3        INDUCTION #3                                                  00001240
 4        EXCESS REPACK                                                 00001250
 5        STOCK OUT                                                     00001260
 6        STAGED FOR SHIPMENT                                           00001270
                                                                        00001280
08SC    TRANSACTION STATUS UPDATE QUEUE                                 00001290
                                                                        00001300
THIS TRANSACTION STATUS UPDATE QUEUE IS USED BY SCTSU TO                00001310
GET THE TRANSACTION NUMBER AND NEW STATUS OF A TRANSACTION              00001320
TO BE UPDATED. IT CONTAINS ADDRESSES OF BUFFERS THAT CONTAIN            00001330
THE INFORMATION NEEDED.                                                 00001340
                                                                        00001350
TSTATUPQ  DEFINEQ   COUNT=50                                            00001360
                                                                        00001370
TO PLACE AN ENTRY ON THE QUEUE USE:                                     00001380
                                                                        00001390
          NEXTQ     TSTATUPQ,LOC,FULL=ROUTINE      ,WHERE:            00001400
                                                                        00001410
LOC     = LABEL ON WORD CONTAINING ADDRESS OF BUFFER                    00001420
ROUTINE = LABEL OF INSTRUCTION TO BRANCH TO IF QUEUE IS FULL.           00001430
                                                                        00001440
TO GET AN ENTRY OFF THE QUEUE USE:                                      00001450
                                                                        00001460
          FIRSTQ    TSTATUPQ,LOC,EMPTY=ROUTINE     ,WHERE:            00001470
                                                                        00001480
LOC=LABEL OF A WORD TO RECEIVE ADDRESS OF BUFFER                        00001490
ROUTINE=LABEL OF INSTRUCTION TO BRANCH TO IF QUEUE IS EMPTY.            00001500
                                                                        00001510
09SC    RECIRCULATION DESTINATION                                       00001520
                                                                        00001530
THE RECIRCULATION DESTINATION WILL BE DESTINATION ZERO (0).             00001540
                                                                        00001550
10SC    SCANNER PROGSTOP EVENT                                          00001560
                                                                        00001570
THIS EVENT IS WAITED ON BY THE SCANNER INITIALIZATION ROUTINE           00001580
AFTER THE SCANNER STOP FLAG HAS BEEN SET.  THIS EVENT WILL              00001590
NEVER BE POSTED, SO A PROGRAM STOP WILL NEVER BE EXECUTED.              00001600
                                                                        00001610
11SC    INDUCTION SCANNER READ ERROR FLAG                               00001620
                                                                        00001630
THIS FLAG WILL BE INTERNAL TO THE SCANNER SUB-SYSTEM. WHEN              00001640
A READ ERROR OCCURS, THE STOP ON 'NO READ' FLAG (14SC) IN               00001650
SYSCOM IS PASSED AS THE ERROR FLAG. IF NO ERROR IS DETECTED             00001660
THE READ ERROR FLAG IS SET TO -1.                                       00001670
                                                                        00001680
PASSING INFORMATION TO SCIND                                            00001690
                                                                        00001700
PASSING INFORMATION TO SCIND IS DONE THROUGH THE USE OF A               00001710
QUEUE. FIRST A BUFFER IS OBTAINED FROM THE INDUCTION BUFFER             00001720
QUEUE (20SC).  THE SCAN INFORMATION IS THEN PUT INTO THIS               00001730
8-BYTE BUFFER.  THE ADDRESS OF THIS BUFFER IS THEN PUT ON               00001740
THE INDUCTION QUEUE (21SC).                                             00001750
                                                                        00001760
THE FORMAT OF THE DATA IN THE BUFFER IS:                                00001770
                                                                        00001780
     WORD#          DATA                                               00001790
                                                                        00001800
       1            SCANNER ID                                         00001810
```

```
            2-3          TRANSACTION NUMBER                              00001820
              4          ERROR FLAG                                      00001830
                                                                         00001840
13SC    ERROR DESTINATION                                               00001850
                                                                         00001860
        ONLY SPECIAL DEFAULT TRANSACTION #000000, GOES TO THIS          00001870
        SPECIAL DESTINATION. THIS DESTINATION MAY VARY FROM SYSTEM      00001880
        TO SYSTEM.                                                      00001890
                                                                         00001900
14SC    STOP ON 'NO READ' FLAG: (STPNR)                                 00001910
                                                                         00001920
        THIS FLAG WILL RESIDE IN SYSCOM AND BE INITIALIZED BY THE       00001930
        OPERATOR INTERFACE SUB-SYSTEM. IF THIS FLAG IS A "0",           00001940
        ALL 'NO READS' AT INDUCTION ARE SENT TO RECIRCULATION.          00001950
        IF IT IS A "1", THEN NO MESSAGE IS SENT TO THE PSC WHEN         00001960
        A 'NO READ' CONDITION OCCURS. THIS STOPS THE PACKAGE            00001970
        AT THE INDUCTION POINT.                                         00001980
                                                                         00001990
        TO EXAMINE THIS FLAG USE THE FOLLOWING PROCESS:                 00002000
                                                                         00002010
        MOVE      #1,$SYSCOM        * GET ADDRESS OF SYSCOM             00002020
        MOVE      #2,(+SCNR,#1)     * GET ADDRESS OF STPNR              00002030
        MOVE      FLAG,(0,#2)       * GET STPNR                         00002040
                                                                         00002050
        "FLAG" NOW CONTAINS THE STOP ON 'NO READ' FLAG.                 00002060
                                                                         00002070
15SC    SLOW DOWN FLAG                                                  00002080
                                                                         00002090
        THIS EVENT WILL BE DEFINED IN SYSCOM.  IT IS A LINK             00002100
        TO THE PSC COMMUNICATIONS SUB SYSTEM.  THIS EVENT               00002110
        WHEN RESET WILL SIGNAL A NEED TO SLOW DOWN THE RATE             00002120
        OF TRANSACTIONS BEING INDUCTED.  WHEN POSTED, EVERY-            00002130
        THING RUNS NORMALLY. WHEN RESET, NO INDUCTION MESSAGE           00002140
        IS SENT TO THE PSC.                                             00002150
                                                                         00002160
16SC    RESERVE STATUS UPDATE QUEUE                                     00002170
                                                                         00002180
        THIS QUEUE IS USED FOR STATUS UPDATES WHEN THE SCANNER          00002190
        SUB SYSTEM GOES INTO SLOW DOWN MODE.  IT IS USED THE            00002200
        SAME WAY AS THE PRIMARY UPDATE QUEUE. SEE (08SC).               00002210
                                                                         00002220
17SC    RESERVE STATUS UPDATE BUFFER QUEUE                              00002230
                                                                         00002240
        THIS QUEUE IS USED TO SUPPLY BUFFERS WHEN THE SCANNER           00002250
        SUB SYSTEM GOES INTO SLOW DOWN MODE.  IT IS USED THE            00002260
        SAME WAY AS THE PRIMARY STATUS UPDATE BUFFER QUEUE. SEE         00002270
        05SC.                                                           00002280
                                                                         00002290
18SC    PACKAGE PROCESSING TASK EVENT (SCINDEV)                         00002300
                                                                         00002310
        THIS EVENT IS POSTED WHEN SOMETHING IS PUT IN THE INDUCTION     00002320
        QUEUE.  IT IS RESET WHEN SCIND HAS EMPTIED THE QUEUE.  IT IS    00002330
        ALSO POSTED BY SCINIT EVERY 5 SECONDS.                          00002340
                                                                         00002350
19SC    STATUS UPDATE TASK EVENT (SCTSUEV)                              00002360
                                                                         00002370
        THIS EVENT IS POSTED WHEN SOMETHING IS PUT IN THE STATUS UPDATE 00002380
        QUEUE.  IT IS RESET WHEN SCTSU HAS EMPTIED THE QUEUE.  IT IS    00002390
        ALSO POSTED BY SCINIT EVERY 5 SECONDS.                          00002400
                                                                         00002410
20SC    INDUCTION BUFFER QUEUE (INDQUEBF)                               00002420
                                                                         00002430
        THIS QUEUE IS USED TO SUPPLY BUFFERS FOR SENDING INFORMATION    00002440
        TO SCIND, (SEE 12SC).  THERE ARE 50 BUFFERS, EACH A SIZE OF     00002450
        8 BYTES.                                                        00002460
                                                                         00002470
21SC    INDUCTION QUEUE (INDQUE)                                        00002480
                                                                         00002490
        THIS QUEUE IS USED TO SEND INFORMATION TO SCIND.  THIS QUEUE    00002500
        CONTAINS 50 1-WORD ELEMENTS.                                    00002510
                                                                         00002520
22SC    NOT USED                                                        00002530
                                                                         00002540
23SC    ROUTING OF ERROR CONDITIONS                                     00002550
                                                                         00002560
```

```
    TRANSACTION SCANNED       :  RECIRC MODE  :  HOLD MODE     00002570
------------------------------+---------------+---------------00002580
1.  NOT ON FILE               :  ERROR CHUTE  :  ERROR CHUTE   00002590
2.  ON FILE, BATCH PENDING    : *RECIRC       : *RECIRC        00002600
3.  ON FILE, BATCH COMPLETE   : *ERROR CHUTE  : *ERROR CHUTE   00002610
4.  TRANS# FOUND, BUT BATCH-  :               :                00002620
    -IS ADDING OR DELETING    :  ERROR CHUTE  :  ERROR CHUTE   00002630
5.  NO READ (SCANNER ERROR)   :  RECIRC       :  HOLD (14SC)   00002640
6.  TRANS# 000000 (13SC)      :  ERROR CHUTE  :  ERROR CHUTE   00002650
                                                               00002660
                    * NOTE - CHANGE STATUS TO 'IN SORTATION'   00002670
                                                               00002680


*****************************************************************************00000010
*                                                                           00000020
* PSC DATA DICTIONARY INDEX                                                 00000030
*                                                                           00000040
*****************************************************************************00000050
                                                                            00000060
I/O #    DESCRIPTION                                                        00000070
                                                                            00000080
01PS     PSC OUTPUT BUFFER FORMAT                                           00000090
02PS     PSC OUTPUT QUEUE AND EVENT                                         00000100
03PS     PSC CALL CODES                                                     00000110
04PS     INDUCT PACKAGE                                                     00000120
05PS     OPEN LINK TO PSC                                                   00000130
06PS     SEND IDLE MESSAGE TO PSC                                           00000140
07PS     COMMUNICATIONS STATUS FLAG                                         00000150
08PS     IDLE MESSAGE COUNT                                                 00000160
09PS     RESPONSE COUNT TABLE                                               00000170
10PS     RESPONSE DUMP FLAG                                                 00000180
11PS     PSC COMMUNICATIONS PANIC CALL                                      00000190
12PS     COMMUNICATIONS SEQUENCE BYTES                                      00000200
13PS     NUMBER OF TIME-OUTS                                                00000210
14PS     INVALID SEQUENCE COUNTER                                           00000220
15PS     INVALID CHECKSUM COUNTER                                           00000230
16PS     PSC MESSAGE FORMATTING                                             00000240
17PS     TRANSMISSION COMPLETION EVENT                                      00000250
18PS     PSC IDLE MESSAGE FORMAT                                            00000260
19PS     PSC OPEN LINK MESSAGE FORMAT                                       00000270
20PS     PSC INDUCT PACKAGE MESSAGE FORMAT                                  00000280
21PS     TRANSMISSION COMPLETION CODE WORD                                  00000290
22PS     PSC RESPONSE MESSAGE FORMAT                                        00000300
23PS     PSC RESPONSE BUFFERS                                               00000310
24PS     PSC RESPONSE QUEUE                                                 00000320
25PS     PSC RESPONSES                                                      00000330
26PS     ALARM PROCESSING ROUTINE                                           00000340
27PS     PSC MESSAGE CHECKSUM CALCULATION                                   00000350
28PS     PSC RESPONSE OVERFLOW EVENT                                        00000360
29PS     PSC INPUT/OUTPUT EVENT                                             00000370
30PS     PSC RESPONSE PROCESSING EVENT                                      00000380


*****************************************************************************00000390
*                                                                           00000400
* PSC DATA REFERENCES                                                       00000410
*                                                                           00000420
*****************************************************************************00000430
                                                                            00000440
I/O #    MODULES REFERENCING                                                00000450
                                                                            00000460
01PS     SCIND,PSPOL,OIPSC                                                  00000470
02PS     SCIND,PSPOL,OIPSC                                                  00000480
03PS     SCIND                                                              00000490
04PS     SCIND                                                              00000500
05PS     OIPSC                                                              00000510
06PS     PSPOL                                                              00000520
07PS     PSOC,PSPAN,OIPSC,SCIND                                             00000530
08PS     PSOC,PSPOL                                                         00000540
09PS     PSOC,PSAO                                                          00000550
10PS     PSOC,PSAO,PSIO                                                     00000560
11PS     PSOC,PSIO,PSRP,PSPOL,PSPAN,PSAO,PSCKSM                             00000570
12PS     PSOC,PSMF                                                          00000580
```

```
13PS      PSOC,PSID                                                    00000590
14PS      PSOC,PSID                                                    00000600
15PS      PSOC,PSID                                                    00000610
16PS      PSOC,PSMF                                                    00000620
17PS      PSOC,PSTD                                                    00000630
18PS      PSMF                                                         00000640
19PS      PSMF                                                         00000650
20PS      PSMF                                                         00000660
21PS      PSIO,PSTD                                                    00000670
22PS      PSIO,PSRP                                                    00000680
23PS      PSIO,PSRP,PSAD                                               00000690
24PS      PSIO,PSRP                                                    00000700
25PS      PSRP,PSAD                                                    00000710
26PS      PSRP,PSAD                                                    00000720
27PS      PSMF,PSCKSM                                                  00000730
28PS      PSIO,PSRP,PSOC                                               00000740
29PS      PSOC,PSID                                                    00000750
30PS      PSDI,PSRP                                                    00000760


*************************************************************************00000770
*                                                                      *00000780
* PSC DATA DICTIONARY ENTRIES                                          *00000790
*                                                                      *00000800
*************************************************************************00000810
                                                                       00000820
01PS      PSC OUTPUT BUFFER FORMAT                                     00000830
                                                                       00000840
          *  CALL CODE  *                                              00000850
          * PARAMETER 1 *                                              00000860
          *      .      *                                              00000870
          *      .      *    SIZE = 5 WORDS                            00000880
          *      .      *                                              00000890
          * PARAMETER N *                                              00000900
                                                                       00000910
          FOR VALID CALL CODES SEE:  03PS                              00000920
                                                                       00000930
          NOTE: ALL BUFFERS ARE TO BE OBTAINED FROM A BUFFER QUEUE     00000940
                DEFINED IN $SYSCOM.                                    00000950
                                                                       00000960
02PS      PSC OUTPUT QUEUE                                             00000970
                                                                       00000980
          THE PSC OUTPUT QUEUE IS DEFINED IN $SYSCOM AND IS USED TO PASS 00000990
          THE ADDRESS OF THE CALL BUFFER TO THE PSC COMMUNICATIONS      00001000
          SUB-SYSTEM. AFTER QUEUEING THE CALL BUFFER, THE PSC          00001010
          RUN EVENT ALSO DEFINED IF $SYSCOM MUST BE POSTED.            00001020
                                                                       00001030
03PS      PSC CALL CODES                                               00001040
                                                                       00001050
          POPN      OPEN LINK TO PSC                                   00001060
          PIDL      SEND IDLE MESSAGE TO PSC                           00001070
          PIND      SEND INDUCT PACKAGE MESSAGE TO PSC                 00001080
                                                                       00001090
04PS      INDUCT PACKAGE                                               00001100
                                                                       00001110
          CODE:  PIND                                                  00001120
                                                                       00001130
                    INPUT BUFFER LAYOUT:                               00001140
                                                                       00001150
                    WORD #     CONTENTS                                00001160
                                                                       00001170
                      1        CALL CODE                               00001180
                     2-3       TRANSACTION NUMBER                      00001190
                      4        INDUCT SCANNER I.D.                     00001200
                      5        SORT DESTINATION                        00001210
                                                                       00001220
05PS      OPEN LINK TO PSC                                             00001230
                                                                       00001240
          CODE: POPN                                                   00001250
                                                                       00001260
                    INPUT BUFFER LAYOUT:                               00001270
                                                                       00001280
                    WORD #     CONTENTS                                00001290
                                                                       00001300
                      1        CALL CODE                               00001310
```

06PS SEND IDLE MESSAGE TO PSC

CODE: PIDL

INPUT BUFFER LAYOUT:

WORD #	CONTENTS
1	CALL CODE

07PS COMMUNICATIONS STATUS FLAG

THE COMMUNISATIONS STATUS FLAG IS A GLOBAL VARIABLE DEFINED IN $SYSCOM. IT CONTAINS THE CURRENT STATUS OF THE COMMUNICATIONS LINK BETWEEN DAS AND PSC.
THE STATUS VALUES ARE:

ZERO = "OPEN"
OTHER = "CLOSED"

08PS IDLE MESSAGE COUNT

THE IDLE MESSAGE COUNT IS A LOCAL VARIABLE DEFINED WITHIN THE PSC COMMUNICATIONS SUB-SYSTEM. IT CONTAINS THE NUMBER OF COMMUNICATION INACTIVITY CYCLES TO BE PERFORMED BEFORE THE PSC REQUIRES AN IDLE MESSAGE TO BE SENT.

09PS RESPONSE COUNT TABLE

THE PSC RESPONSE COUNT TABLE CONTAINS ALL POSSIBLE PSC RESPONSE CODES (EXCEPT TRANS. COMPLETE) AND A COUNT THAT INDICATES THE NUMBER OF TIMES DAS HAS RECEIVED THIS RESPONSE FROM THE PSC. IN ADDITION TO THESE COUNTS, ONE ADDITIONAL COUNTER IS DEFIND FOR ALL UNKNOWN RESPONSES RECEIED FROM THE PSC.

10PS RESPONSE DUMP FLAG

THE PSC RESPONSE DUMP FLAG IS A LOCAL VARIABLE DEFINED WITHIN THE PSC COMMUNICATIONS SUB-SYSTEM. WHEN SET, ALL ABNORMAL RESPONSES RECEIVED BY DAS FROM PSC WILL BE OUTPUT ON THE LOG DEVICES.

11PS PSC COMMUNICATIONS PANIC CALL

TO PROCESS A PANIC WITHIN THE PSC COMMUNICATIONS SUB-SYSTEM THE USER SHOULD CALL THE SUBROUTINE "PSPAN" WITH THE PANIC CODE AS A PARAMETER. NO RETURN WILL BE MADE FROM THIS CALL.

12PS COMMUNICATION SEQUENCE BYTES

THERE ARE TWO COMMUNICATION SEQUENCE BYTES, ONE FOR INPUT FROM THE PSC AND THE OTHER FOR NEXT OUTPUT TO THE PSC. AFTER EACH SUCCESSFUL INTERCHANGE BETWEEN DAS AND PSC, THESE SEQUENCE BYTES WILL BE UPDATED TO REFLECT THE LAST SEQUENCE VALUE RECEIVED AND LAST SEQUENCE VALUE TRANSMITTED. THE INITIAL VALUES OF THESE SEQUENCE BYTES FOLLOW:

INPUT = EBCDIC "5"
OUTPUT = EBCDIC "5"

THE ALTERNATE SEQUENCE BYTES VALUES = EBCDIC "J".

```
13PS    NUMBER OF TIME-OUTS                                          00001960
                                                                     00001970
        THIS COUNTER IS A LOCAL VARIABLE DEFINED WITHIN THE PSC      00001980
        COMMUNICATIONS SUB-SYSTEM. IT IS INCREMENTED EACH TIME A     00001990
        PSC RESPONSE HAS TIMED OUT.                                  00002000
                                                                     00002010
14PS    INVALID SEQUENCE COUNTER                                     00002020
                                                                     00002030
        THIS COUNTER IS A LOCAL VARIABLE DEFINED WITHIN THE PSC      00002040
        COMMUNICATIONS SUB-SYSTEM. IT IS INCREMENTED EACH TIME       00002050
        THE PSC RESPONDS WITH AN INVALID SEQUENCE BYTE.              00002060
                                                                     00002070
15PS    INVALID CHECKSUM COUNTER                                     00002080
                                                                     00002090
        THIS COUNTER IS A LOCAL VARIABLE DEFINED WITHIN THE PSC      00002100
        COMMUNICATIONS SUB-SYSTEM. IT IS INCREMENTED EACH TIME       00002110
        THE PSC RESPONDS WITH AN INVAID CHECKSUM.                    00002120
                                                                     00002130
16PS    PSC MESSAGE FORMATTING                                       00002140
                                                                     00002150
        TO FORMAT A PSC OUTPUT MESSAGE, THE SUBROUTINE "PSMF" IS     00002160
        CALLED. PARAMETERS PASSED TO THIS SUBROUTINE ARE:            00002170
        DATA INPUT FROM PSC OUTPUT QUEUE.                            00002180
        PARAMETERS RETURNED FROM THIS SUBROUTINE ARE:                00002190
        COMPLETION CODE (SUCCESSFUL, NOT SUCCESSFUL).                00002200
                                                                     00002210
17PS    TRANSMISSION COMPLETION EVENT                                00002220
                                                                     00002230
        THIS EVENT IS DEFINED WITHIN THE PSC COMMUNICATIONS SUB-     00002240
        SYSTEM AND IS USED TO PASS ONE OF THE FOLLOWING CODES        00002250
        BACK TO THE PSC OUTPUT CONTROL TASK "PSOC."                  00002260
                                                                     00002270
                  RETRY TRANSMISSION                                 00002280
                  DATA RECEIVED                                      00002290
                  IDLE RECEIVED                                      00002300
                                                                     00002310
18PS    PSC IDLE MESSAGE FORMAT                                      00002320
                                                                     00002330
        BYTE #    CONTENTS                                           00002340
                                                                     00002350
          1       CURRENT OUTPUT SEQUENCE VALUE  (EBCDIC)            00002360
          2       IDLE CONTROL CODE "I"          (EBCDIC)            00002370
        3-13      SPACES                         (EBCDIC)            00002380
         14       CHECKSUM                       (EBCDIC)            00002390
                                                                     00002400
        NOTE: CHECKSUM = ADDITION OF BYTES 1-13 THEN FORCED TO       00002410
              BE A PRINTABLE CHARACTER.                              00002420
                                                                     00002430
19PS    PSC OPEN LINK MESSAGE FORMAT                                 00002440
                                                                     00002450
        BYTE #    CONTENTS                                           00002460
                                                                     00002470
          1       CURRENT OUTPUT SEQUENCE VALUE  (EBCDIC)            00002480
          2       OPEN LINK MESSAGE CODE "L"     (EBCDIC)            00002490
        3-13      SPACES                         (EBCDIC)            00002500
         14       CHECKSUM                                           00002510
                                                                     00002520
        NOTE: CHECKSUM = ADDITION OF BYTES 1-13 THEN FORCED TO       00002530
              BE A PRINTABLE CHARACTER.                              00002540
                                                                     00002550
20PS    PSC INDUCT PACKAGE MESSAGE FORMAT                            00002560
                                                                     00002570
        BYTE #    CONTENTS                                           00002580
                                                                     00002590
          1       CURRENT OUTPUT SEQUENCE VALUE  (EBCDIC)            00002600
          2       INDUCT PACKAGE CODE "D"        (EBCDIC)            00002610
        3-8       TRANSACTION NUMBER             (EBCDIC)            00002620
        9-10      INDUCT NUMBER                  (EBCDIC)            00002630
       11-13      SORT DESTINATION               (EBCDIC)            00002640
         14       CHECKSUM                       (EBCDIC)            00002650
                                                                     00002660
        NOTE: CHECKSUM = ADDITION OF BYTES 1-13 THEN FORCED TO       00002670
              BE A PRINTABLE CHARACTER.                              00002680
                                                                     00002690
```

```
21PS    TRANSMISSION COMPLETION CODE EVENT                              00002700
                                                                        00002710
        THE COMPLETION CODE ECB IS A LOCAL EVENT DEFINED WITHIN         00002720
        THE PSC COMMUNICATINS SUB-SYSTEM. IT IS USED TO HOLD THE        00002730
        TRANSMISSION COMPLETION EVENT CODE THAT IS POSTED BACK TO       00002740
        THE PSC OUTPUT CONTROL TASK "PSOC."                             00002750
                                                                        00002760
22PS    PSC RESPONSE MESSAGE FORMAT                                     00002770
                                                                        00002780
        BYTE #    CONTENTS                                              00002790
                                                                        00002800
          1       SEQUENCE BYTE VALUE       (EBCDIC)                    00002810
          2       RESPONSE MESSAGE TYPE     (EBCDIC)                    00002820
        3-13      RESPONSE PARAMETERS       (EBCDIC)                    00002830
         14       CHECKSUM                  (EBCDIC)                    00002840
         15       CARRIAGE RETURN           (EBCDIC)                    00002850
         16       LINE FEED                 (EBCDIC)                    00002860
                                                                        00002870
        NOTE: CHECKSUM = ADDITION OF BYTES 1-13 THEN FORCED TO          00002880
              BE A PRINTABLE CHARACTER.                                 00002890
                                                                        00002900
              FOR VALID RESPONSE MESSAGE TYPES SEE: 25PS                00002910
                                                                        00002920
23PS    PSC RESPONSE BUFFERS                                            00002930
                                                                        00002940
        BYTE #    CONTENTS                                              00002950
                                                                        00002960
          1       SEQUENCE VALUE RECEIVED     (EBCDIC)                  00002970
          2       RESPONSE MESSAGE TYPE       (EBCDIC)                  00002980
        3-13      RESPONSE PARAMETERS         (EBCDIC)                  00002990
         14       CHECKSUM                    (EBCDIC)                  00003000
                                                                        00003010
        NOTE: ALL BUFFERS ARE TO BE OBTAINED FROM A BUFFER QUEUE        00003020
              DEFINED WITHIN THE PSC COMMUNICATIONS SUB-SYSTEM.         00003030
                                                                        00003040
              FOR VALID RESPONSE MESSAGE TYPES SEE: 25PS                00003050
                                                                        00003060
24PS    PSC RESPONSE QUEUE                                              00003070
                                                                        00003080
        THIS QUEUE IS DEFINED WITHIN THE PSC COMMUNICATIONS SUB-        00003090
        SYSTEM AND IS USED TO PASS THE PSC RESPONSE MESSAGE TO          00003100
        THE PSC RESPONSE PROCESSOR ROUTINE.                             00003110
                                                                        00003120
25PS    PSC RESPONSES                                                   00003130
                                                                        00003140
        FOR PSC RESPONSES SEE: SOFTWARE MANUAL FOR THE PROGRAMMABLE     00003150
        SORT CONTROLLER, SECTION 9, DOCUMENT NUMBER                     00003160
        76-01730-600-0094-??.                                           00003170
                                                                        00003180
26PS    ALARM PROCESSING ROUTINE                                        00003190
                                                                        00003200
        THIS ROUTINE IS USED TO PRINT MESSAGES ON THE LOG DEVICES       00003210
        FOR PSC RESPONSES OTHER THAN A TRANSFER COMPLETE. PARAMETERS    00003220
        PASSED TO THIS ROUTINE ARE: DATA RECEIVED FROM THE PSC.         00003230
                                                                        00003240
27PS    PSC MESSAGE CHECKSUM CALCULATION                                00003250
                                                                        00003260
        TO CALCULATE A MESSAGE CHECKSUM, CALL ROUTINE 'PSCKSM'          00003270
        WITH THE ADDRESS OF THE MESSAGE AND THE ADDRESS WHERE           00003280
        THE CHECKSUM IS TO BE PLACED.                                   00003290
                                                                        00003300
28PS    PSC RESPONSE OVERFLOW EVENT                                     00003310
                                                                        00003320
        THIS EVENT INDICATES WHEN THERE HAS BEEN A PSC RESPONSE         00003330
        OVERFLOW CONDITION DO TO THE RESPONSE QUEUE BEING FULL.         00003340
        WHEN THE EVENT IS RESET, ALL NEW TRANSMISSIONS TO THE PSC       00003350
        ARE DELAYED UNTIL THE PSC RESPONSE QUEUE IS CLEANED OUT         00003360
        AND THE EVENT ID POSTED. THIS EVENT IS DEFINED WITHIN           00003370
        THE PSC COMMUNICATIONS SUB-SYSTEM.                              00003380
                                                                        00003390
29PS    PSC INPUT/OUTPUT EVENT                                          00003400
                                                                        00003410
```

```
          THIS EVENT IF DEFINED WITHIN THE PSC COMMUNICATIONS SUB-SYSTEM.00003420
          WHEN POSTED, IT INITIATES THE PSC INPUT/OUTPUT SEQUENCE.       00003430
          THIS EVENT IS RESET AT THE END OF A PSC INPUT/OUTPUT           00003440
          SEQUENCE.                                                      00003450
                                                                         00003460
30PS      PSC RESPONSE PROCESSING EVENT                                  00003470
                                                                         00003480
          THIS EVENT ID DEFINED WITHIN THE PDC COMMUNICATIONS SUB-SYSTEM.00003490
          WHEN POSTED, IT INITIATES THE PSC RESPONSE PROCESSING SEQUENCE.00003500
          THIS EVENT IS RESET AFTER ALL CURRENT PSC RESPONSES HAVE       00003510
          BEEN PROCESSED.                                                00003520



*************************************************************************00000010
*                                                                       00000020
* ALARM MESSAGE DATA DICTIONARY INDEX                                   00000030
*                                                                       00000040
*************************************************************************00000050
                                                                         00000060
I/O #                    DESCRIPTION                                   00000070
                                                                         00000080
01AL      ALMQUE     ALARM MESSAGE QUEUE                               00000090
02AL      BUFFQUE    BUFFER POOL QUEUE                                 00000100
03AL                 ALARM MESSAGES                                    00000110
04AL      TCW        TERMINAL CONROL WORD                              00000120
05AL      OTADTBL    OUTPUT TASK START EVENT TABLE                     00000130
06AL      OTEETBL    OUTPUT TASKS END-EVENT TABLE                      00000140
07AL                 PANIC                                             00000150
08AL                 FATAL ERROR                                       00000160
09AL      TIMRTSK    TIME OUT TASK                                     00000170
10AL      DATES      DATE AND TIME SUBROUTINE                          00000180
11AL      CRFLG      MESSAGE COMPRESS FLAG                             00000190
12AL                 NON-COMPRESSABLE MESSAGES                         00000200
13AL      MOVTXT     MESSAGE COMPRESSING SUBROUTINE                    00000210
                                                                         00000220



*************************************************************************00000230
*                                                                       00000240
* ALARM MESSAGES DATA REFERENCES                                        00000250
*                                                                       00000260
*************************************************************************00000270
                                                                         00000280
I/O #                    MODULES REFERENCING                           00000290
                                                                         00000300
01AL      ALMPG,SCTSU,SCIND,SCSTO,SCSHP,SCXRP,PSOC,PSRP,PSAO,OICCPA      00000310
          $INIT,PSIO,OIHLT,OIREC,OICCSA                                  00000320
02AL      ALMPG,SCTSU,SCIND,SCSTO,SCSHP,SCXRP,PSOC,PSRP,PSAO,OICCSA      00000330
          $INIT,PSIO,OIHLT,OIREC                                         00000340
03AL.01   SCIND                                                         00000350
     .02  SCIND,SCTSU                                                   00000360
     .03  SCIND,SCSHP,SCSTO,SCXRP                                       00000370
     .04  PSAO                                                          00000380
     .05  PSOC                                                          00000390
     .06  PSAO                                                          00000400
     .07  OISTB                                                         00000410
     .08  OIENB                                                         00000420
     .09  PSOC                                                          00000430
     .10  PSAO                                                          00000440
     .11  PSAO                                                          00000450
     .12  $INIT                                                        00000460
     .13  $INIT                                                        00000470
     .14  PSIO                                                          00000480
     .15  PSIO                                                          00000490
     .16  PSIO                                                          00000500
     .17  PSRP                                                          00000510
     .18  PSAO                                                          00000520
     .19  OIHLT                                                         00000530
     .20  OIREC                                                         00000540
     .21  PSAO                                                          00000550
```

```
04AL    ALMPG,AXCNFG                                                    00000560
05AL    ALMPG,AXCNFG                                                    00000570
06AL    ALMPG,ALTRM1,ALTRM2,ALTRM3                                      00000580
07AL    ALMPG,SCIND,SCTSU,SCSTO,SCSHP,SCXRP,PSPAN,FMPAN,$INIT,DICCPA,   00000590
        DIREC,OIHLT                                                     00000600
08AL    ALMPG,SCIND,SCTSU,SCSTO,SCSHP,SCXRP,PSPAN,FMPAN,$INIT,DICCPA,   00000610
        SCSR1,SCSR2,SCSR3                                               00000620
09AL    ALMPG                                                           00000630
10AL    ALMPG                                                           00000640
11AL    ALMPG                                                           00000650
12AL    ALMPG                                                           00000660
13AL    ALMPG                                                           00000670
                                                                        00000680


*****************************************************************************00000690
*                                                                       00000700
* ALARM MESSAGE DATA DICTIONARY ENRIES                                  00000710
*                                                                       00000720
*****************************************************************************00000730
                                                                        00000740
01AL    ALARM MESSAGE QUEUE                                             00000750
                                                                        00000760
        ALMQUE                                                          00000770
                                                                        00000780
                THE ALARM MESSAGE QUEUE RESIDES IN SYSCOM. EACH ENTRY   00000790
                IN THIS QUEUE IS 1-WORD LONG AND IS JUST A POINTER      00000800
                TO A DATA BUFFER. THIS 8-WORD BUFFER IS WHERE THE       00000810
                DATA FOR THE MESSAGE IS STORED. THIS BUFFER IS ALSO     00000820
                IN SYSCOM. SEE D.D. 12.                                 00000830
                                                                        00000840
                THE MAXIMUM NUMBER OF ENTRIES THAT CAN BE PUT IN        00000850
                ALMQUE IS 110.                                          00000860
                                                                        00000870
        TO PUT A MESSAGE ON THE ALARM QUEUE USE:                        00000880
                                                                        00000890
        LABEL   NEXTQ   *,LOC,FULL=ROUTINE,P1=ALMQUE    WHERE:         00000900
                                                                        00000910
        LOC=LABEL ON A WORD CONTAINING THE ADDRESS OF THE BUFFER        00000920
            THAT CONTAINS THE MESSAGE DATA.                             00000930
        ROUTINE=LABEL OF STATEMENT TO EXECUTE IF THE QUEUE IS FULL.     00000940
        ALMQUE=THE ADDRESS OF THE ALARM BUFFER                          00000950
                                                                        00000960
        TO GET THE ADDRESS OF ALMQUE, USE THE FOLLOWING PROCESS:        00000970
                                                                        00000980
                MOVE    #1,$SYSCOM    * GET ADDRESS OF SYSCOM           00000990
                MOVE    ALMQUE,(*ALMS,#1) * GET ADDRESS OF ALMQUE       00001000
                                                                        00001010
02AL    BUFFQUE                                                         00001020
                                                                        00001030
        BUFFQUE IS A QUEUE THAT CONTAINS ALARM MESSAGE OUTPUT           00001040
        BUFFERS. THESE BUFFERS ARE 8-WORDS LONG. TO OBTAIN A            00001050
        BUFFER USE:                                                     00001060
                                                                        00001070
        LABEL   FIRSTQ   *,LOC,EMPTY=ROUTINE,P1=BUFFQUE                 00001080
                                                                        00001090
        WHERE:                                                          00001100
                                                                        00001110
        LOC=NAME OF A VARIABLE TO RECEIVE THE ADDRESS OF THE BUFFER     00001120
        ROUTINE=NAME OF STATEMENT TO BRANCH TO IF BUFFQUE IS EMPTY.     00001130
        BUFFQUE=THE ADDRESS OF THE BUFFER QUEUE                         00001140
                                                                        00001150
        WHEN FILLING THIS BUFFER WITH ALARM MESSAGE DATA USE THE        00001160
        FOLLOWING FORMAT.                                               00001170
                                                                        00001180
        WORD #    CONTENTS                                              00001190
                                                                        00001200
          1       MSG #              SEE DATA DICTIONARY 13             00001210
         2-8      MESSAGE PARAMETERS                                    00001220
                                                                        00001230
        TO RESTORE A BUFFER USE:                                        00001240
                                                                        00001250
                NEXTQ   *,LOC,FULL=ROUTINE,P1=BUFFQUE                   00001260
                                                                        00001270
```

```
WHERE:                                                                  00001280
                                                                        00001290
LOC=LABEL ON THE ADDRESS OF THE BUFFER                                  00001300
ROUTINE=INSTRUCTION TO BRANCH TO IF BUFFQUE IS FULL.                    00001310
BUFFQUE=THE ADDRESS OF THE BUFFER QUEUE                                 00001320
                                                                        00001330
TO OBTAIN THE ADDRESS OF BUFFQUE USE:                                   00001340
                                                                        00001350
          MOVE      #1,$SYSCOM    * GET ADDRESS OF SYSCOM               00001360
          MOVE      BUFFQUE,(*ALBF,#1) * GET ADDRESS OF BUFFQUE         00001370
                                                                        00001380
03AL      ALARM MESSAGES                                                00001390
                                                                        00001400
          MSG #     MESSAGE                                           00001410
                                                                        00001420
            1       TRANSACTION (TRANS #) AT (SCANNER NAME)           00001430
                    SCANNER OUT OF BATCH.                             00001440
                                                                        00001450
            2       TRANSACTION (TRANS #) AT (SCANNER NAME)           00001460
                    SCANNER NOT ON FILE.                              00001470
                                                                        00001480
            3       SCANNER ERROR AT (SCANNER NAME) SCANNER.          00001490
                                                                        00001500
            4       DATA AS RECEIVED FROM PSC = (DATA)                00001510
                                                                        00001520
            5       PSC COMMUNICATIONS LINE OPEN                      00001530
                                                                        00001540
            6       TRANSACTION (TRANS #) MIS-SORTED TO SORT LINE     00001550
                    (LINE #)                                          00001560
                                                                        00001570
            7       START OF BATCH (BATCH ID)                         00001580
                                                                        00001590
            8       END OF BATCH (BATCH ID)                           00001600
                                                                        00001610
            9       PSC COMMUNICATIONS FAILURE                        00001620
                                                                        00001630
           10       SORTATION LINE (DESTINATION) FULL.                00001640
                                                                        00001650
           11       UNSUCCESSFUL TRANSFER AT SORT LINE (DESTINATION)  00001660
                    FOR TRANSACTION (TRANS #)                         00001670
                                                                        00001680
           12       SYSTEM STARTED                                    00001690
                                                                        00001700
           13       POWER FAIL/RESTART                                00001710
                                                                        00001720
           14       INVALID CHECKSUM FROM PSC, DATA = (DATA)          00001730
                                                                        00001740
           15       INVALID SEQUENCE FROM PSC, DATA = (DATA)          00001750
                                                                        00001760
           16       PSC TIME-OUT                                      00001770
                                                                        00001780
           17       TRANSACTION (TRANS #) ON SORT LINE (LINE #)       00001790
                    NOT ON FILE                                       00001800
                                                                        00001810
           18       DATA RECEIVED FROM PSC OUT-OF-RANGE, DATA = (DATA) 00001820
                                                                        00001830
           19       HOLD ON SCANNER ERROR                             00001840
                                                                        00001850
           20       RECIRCULATE ON SCANNER ERROR                      00001860
                                                                        00001870
           21       TRANSACTION (TRANS #) HAS BEEN LOST FROM TRACKING 00001880
                                                                        00001890
          ALARM MESSAGE INPUT BUFFER                                    00001900
                                                                        00001910
          *MESSAGE #1          TRANSACTION OUT OF BATCH                 00001920
                                                                        00001930
                    INPUT BUFFER LAYOUT                               00001940
                                                                        00001950
                    WORD #      CONTENTS                              00001960
                                                                        00001970
                      1         MESSAGE # (1)                         00001980
                     2-3        TRANSACTION #                         00001990
                     4-5        SCANNER ID                            00002000
                     6-8        UNUSED                                00002010
                                                                        00002020
```

```
*MESSAGE #2          TRANSACTION NOT ON FILE                    00002030
                                                                00002040
          INPUT BUFFER LAYOUT                                   00002050
                                                                00002060
          WORD #     CONTENTS                                   00002070
                                                                00002080
            1        MESSAGE # (2)                              00002090
           2-3       TRANSACTION #                              00002100
           4-5       SCANNER ID                                 00002110
           6-8       UNUSED                                     00002120
                                                                00002130
*MESSAGE #3          NO READ AT (SCANNER ID) SCANNER            00002140
                                                                00002150
          INPUT BUFFER LAYOUT                                   00002160
                                                                00002170
          WORD #     CONTENTS                                   00002180
                                                                00002190
            1        MESSAGE #(3)                               00002200
           2-3       UNUSED                                     00002210
           4-5       SCANNER ID                                 00002220
           6-8       UNUSED                                     00002230
                                                                00002240
*MESSAGE #4          PSC DATA MESSAGE                           00002250
                                                                00002260
          INPUT BUFFER LAYOUT                                   00002270
                                                                00002280
          WORD #     CONTENTS                                   00002290
                                                                00002300
            1        MESSAGE #(4)                               00002310
           2-8       DATA RECEIVED FROM PSC  (EBCDIC)           00002320
                                                                00002330
*MESSAGE #5          PSC COMMUNICATIONS MESSAGE                 00002340
                                                                00002350
          INPUT BUFFER LAYOUT                                   00002360
                                                                00002370
          WORD #     CONTENTS                                   00002380
                                                                00002390
            1        MESSAGE # (5)                              00002400
           2-8       UNUSED                                     00002410
                                                                00002420
*MESSAGE #6          MIS-SORTED CARTON MESSAGE                  00002430
                                                                00002440
          INPUT BUFFER LAYOUT                                   00002450
                                                                00002460
          WORD #     CONTENTS                                   00002470
                                                                00002480
            1        MESSAGE # (6)                              00002490
           2-4       TRANSACTION #              (EBCDIC)        00002500
           5-6       MIS-SORTED SHIPPING LINE #  (EBCDIC)       00002510
           7-8       UNUSED                                     00002520
                                                                00002530
*MESSAGE #7          START OF BATCH                             00002540
                                                                00002550
          INPUT BUFFER LAYOUT                                   00002560
                                                                00002570
          WORD #     CONTENTS                                   00002580
                                                                00002590
            1        MESSAGE # (7)                              00002600
           2-3       BATCH ID (RIGHT JUSTIFIED)                 00002610
           4-8       UNUSED                                     00002620
                                                                00002630
*MESSAGE #8          END OF BATCH                               00002640
                                                                00002650
          INPUT BUFFER LAYOUT                                   00002660
                                                                00002670
          WORD #     CONTENTS                                   00002680
                                                                00002690
            1        MESSAGE # (8)                              00002700
           2-3       BATCH ID (RIGHT JUSTIFIED)                 00002710
           4-8       UNUSED                                     00002720
                                                                00002730
MESSAGE #9           PSC COMMUNICATION FAILURE                  00002740
                                                                00002750
          INPUT BUFFER LAYOUT                                   00002760
                                                                00002770
```

```
          WORD #     CONTENTS                                          00002780
                                                                       00002790
            1        MESSAGE # (9)                                     00002800
           2-8       UNUSED                                            00002810
                                                                       00002820
*MESSAGE #10         SORTATION LINE FULL                               00002830
                                                                       00002840
          INPUT BUFFER LAYOUT                                          00002850
                                                                       00002860
          WORD #     CONTENTS                                          00002870
                                                                       00002880
            1        MESSAGE # (10)                                    00002890
           2-3       DESTINATION  (EBCDIC)                             00002900
           4-8       UNUSED                                            00002910
                                                                       00002920
*MESSAGE #11         TRANSFER FAILURE                                  00002930
                                                                       00002940
          INPUT BUFFER LAYOUT                                          00002950
                                                                       00002960
          WORD #     CONTENTS                                          00002970
                                                                       00002980
            1        MESSAGE # (11)                                    00002990
           2-4       TRANSACTION #  (EBCDIC)                           00003000
           5-6       DESTINATION    (EBCDIC)                           00003010
           7-8       UNUSED                                            00003020
                                                                       00003030
*MESSAGE #12         SYSTEM STARTED                                    00003040
                                                                       00003050
          INPUT BUFFER LAYOUT                                          00003060
                                                                       00003070

          WORD #     CONTENTS                                          00003080
                                                                       00003090
            1        MESSAGE # (12)                                    00003100
           2-8       UNUSED                                            00003110
                                                                       00003120
*MESSAGE #13         POWER FAIL/RESTART                                00003130
                                                                       00003140
          INPUT BUFFER LAYOUT                                          00003150
                                                                       00003160
          WORD #     CONTENTS                                          00003170
                                                                       00003180
            1        MESSAGE # (13)                                    00003190
           2-8       UNUSED                                            00003200
                                                                       00003210
*MESSAGE #14         INVALID CHECKSUM FROM PSC                         00003220
                                                                       00003230
          INPUT BUFFER LAYOUT                                          00003240
                                                                       00003250
          WORD #   CONTENTS                                           00003260
                                                                       00003270
            1      MESSAGE #(14)                                       00003280
           2-8     DATA RECEIVED FROM PSC  (EBCDIC)                    00003290
                                                                       00003300
*MESSAGE #15         INVALID SEQUENCE FROM PSC                         00003310
                                                                       00003320
          INPUT BUFFER LAYOUT                                          00003330
                                                                       00003340
          WORD #   CONTENTS                                           00003350
                                                                       00003360
            1      MESSAGE #(15)                                       00003370
           2-8     DATA RECEIVED FROM PSC  (EBCDIC)                    00003380
                                                                       00003390
*MESSAGE #16         PSC TIME-OUT                                      00003400
                                                                       00003410
          INPUT BUFFER LAYOUT                                          00003420
                                                                       00003430
          WORD #   CONTENTS                                           00003440
                                                                       00003450
            1      MESSAGE #(16)                                       00003460
           2-8     UNUSED                                              00003470
                                                                       00003480
*MESSAGE #17         TRANSACTION ON SORT LINE NOT ON FILE              00003490
                                                                       00003500
          INPUT BUFFER LAYOUT                                          00003510
```

```
                WORD #     CONTENTS

                  1        MESSAGE # (17)
                 2-4       TRANSACTION #  (EBCDIC)
                 5-6       DESTINATION    (EBCDIC)
                 7-8       UNUSED

*MESSAGE #18               DATA RECEIVED FROM PSC OUT-OF-RANGE

                INPUT BUFFER LAYOUT

                WORD #     CONTENTS

                  1        MESSAGE # (18)
                 2-8       DATA RECEIVED FROM PSC (EBCIDC)

*MESSAGE #19               HOLD ON SCANNER ERROR

                INPUT BUFFER LAYOUT

                WORD #     CONTENTS

                  1        MESSAGE # (19)
                 2-8       UNUSED

*MESSAGE #20               RECIRCULATE ON SCANNER ERROR

                INPUT BUFFER LAYOUT

                WORD #     CONTENTS

                  1        MESSAGE # (20)
                 2-8       UNUSED

*MESSAGE #21               TRANSACTION HAS BEEN LOST FROM TRACKING

                INPUT BUFFER LAYOUT

                WORD #     CONTENTS

                  1        MESSAGE # (21)
                 2-4       TRANSACTION #        (EBCDIC)
                 5-8       UNUSED

04AL    TERMINAL CONTROL WORD          (TCW)

THE TERMINAL CONTROL WORDS ARE LOCATED IN $SYSCOM (SEE 39FM).
ASSOCIATED WITH EACH MESSAGE THAT ALMPG WILL OUTPUT IS A TCW
WHICH SPECIFIES WHICH TERMINAL(S) ANY SPECIFIC MESSAGE WILL BE
SENT TO.

EACH BIT IN THE TCW CORRESPONDS TO AN OUTPUT DEVICE.
IF THE BIT IS SET 'ON', TO A ONE, THEN THE ASSOCIATED
MESSAGE IS SENT TO THAT DEVICE.

                    TCW

0/1/2/3/4/5/6/7/8/9/10/11/12/13/14/15  BITS

BIT                 OUTPUT DEVICE

 0                  LOG DEVICE #1
 1                  LOG DEVICE #2
 2                  LOG DEVICE #3
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
```

```
14                                                                  00004280
15                                                                  00004290
                                                                    00004300
TERMINAL CONTROL WORD TABLE    (MESSAGE ROUTING TABLE)              00004310
                                                                    00004320
THE TERMINAL CONTROL WORDS FOR ALL OF THE ALARM MESSAGES            00004330
ARE IN THE TCW TABLE IN $SYSCOM. TO REFERENCE A TCW FOR A           00004340
MESSAGE GET THE ADDRESS OF THE TABLE (SEE 39FM). SUBTRACT           00004350
ONE FROM THE MESSAGE NUMBER, DOUBLE IT, AND ADD IT TO               00004360
THIS ADDRESS.                                                       00004370
                                                                    00004380
          TCW1      TCW FOR MESSAGE 1                               00004390
          TCW1      TCW FOR MESSAGE 2                               00004400
           .              .                                         00004410
           .              .                                         00004420
           .              .                                         00004430
          TCW N     TCW FOR MESSAGE N                               00004440
                                                                    00004450
05AL      OUTPUT TASK START EVENT TABLE     (OTADTBL)               00004460
                                                                    00004470
THIS TABLE RESIDES IN THE ALMPG PROGRAM. IT IS A TABLE              00004480
OF ECB ADDRESSES FOR THE OUTPUT TASKS. THESE TASKS ARE              00004490
USED FOR PRINTING ALARM MESSAGES.                                   00004500
                                                                    00004510
OUTPUT TASK START EVENT TABLE:                                      00004520
                                                                    00004530
OTADTBL   EQU       *                                               00004540
          DATA      A(ALTRM1S)                                      00004550
          DATA      A(ALTRM2S)                                      00004560
            .           .                                           00004570
            .           .                                           00004580
            .           .                                           00004590
          DATA      A(ALTRMNS)                                      00004600
                                                                    00004610
THERE WILL BE ONE OF THESE EVENTS FOR EACH DEVICE                   00004620
THAT AN ALARM MESSAGE IS TO BE SENT TO.                             00004630
                                                                    00004640
05AL      OUTPUT TASKS END-EVENT TABLE      (OTEETBL)               00004650
                                                                    00004660
THIS TABLE IS A TABLE OF ADDRESSES. THESE ADDRESSES ARE             00004670
THE ADDRESSES OF THE ECB'S FOR THE END EVENTS OF THE                00004680
OUTPUT TASKS.                                                       00004690
                                                                    00004700
OUTPUT TASK END EVENT TABLE:                                        00004710
                                                                    00004720
OTEETBL   EQU       *                                               00004730
          DATA      A(ALTRM1E)                                      00004740
          DATA      A(ALTRM2E)                                      00004750
            .           .                                           00004760
            .           .                                           00004770
            .           .                                           00004780
          DATA      A(ALTRMNE)                                      00004790
                                                                    00004800
07AL      PANIC                                                     00004810
                                                                    00004820
IF A FATAL SYSTEM ERROR IS ENCOUNTERED, A PROGRAM CAN PANIC         00004830
BY PUTTING A NUMBER ON THE ALARM MESSAGE QUEUE (ALMQUE)             00004840
THAT IS LESS THAN THE ADDRESS OF $SYSCOM. THIS WILL CAUSE           00004850
A FATAL ERROR MESSAGE TO BE PRINTED ON ALL LOG PRINTERS.            00004860
THE NUMBER PUT ON THE QUEUE WILL ALSO BE PRINTED.                   00004870
                                                                    00004880
WHEN USING THIS TECHNIQUE THE PROGRAM PANICING MUST MAKE            00004890
MAKE SURE THAT IT BRINGS ITSELF DOWN.                               00004900
                                                                    00004910
FOR A LIST OF PANIC CODES SEE SECTION 08AL.                         00004920
                                                                    00004930
08AL      FATAL ERROR                                               00004940
                                                                    00004950
IF A PROGRAM HAS A FATAL ERROR IT WILL PUT A UNIQUE ERROR           00004960
CODE ON THE ALARM MESSAGE QUEUE (ALMQUE). WHEN THE ALARM            00004970
MESSAGE PROGRAM DEQUEUES THE ERROR CODE, IT WILL FORMAT             00004980
THE ERROR CODE INTO A MESSAGE OF THE FORM:                          00004990
                                                                    00005000
```

```
          FATAL ERROR DD          WHERE:                                00005010
                                                                        00005020
          DD = THE PANIC CODE                                           00005030
                                                                        00005040
THIS MESSAGE WILL BE PRINTED ON ALL LOG DEVICES.                        00005050
                                                                        00005060
FILE MANAGER FATAL ERRORS                                               00005070
                                                                        00005080
ERROR #   MODULE    ERROR                                               00005090
                                                                        00005100
  01        FMRWB     DISK ERROR WHILE ADDING BATCH RECORD              00005110
  02        FMRWB     DISK ERROR WHILE ADDING TRANS. RECORD             00005120
  03        FMRWB     DISK ERROR WHILE DELETING BATCH                   00005130
  04        FMRWB     DISK ERROR WHILE CHANGING BATCH STATUS            00005140
  05        FMRWB     DISK ERROR WHILE CHANGING TRANS. STATUS           00005150
                      (TERMINALS)                                       00005160
  06        FMRWB     DISK ERROR WHILE CHANGING TRANS. STATUS           00005170
                      (SCANNERS)                                        00005180
  07        FMRWB     DISK ERROR WHILE READING BATCH REC. VIA           00005190
                      BATCH I.D.                                        00005200
  08        FMRWB     DISK ERROR WHILE READING TRANS. REC. VIA          00005210
                      TRANS. NO.                                        00005220
  09        FMRWB     DISK ERROR WHILE READING TRANS. DIR. REC.         00005230
  10        FMRWB     DISK ERROR WHILE READING TRANS. FILE BLOCK        00005240
  11        FMRWB     DISK ERROR WHILE READING ORDER REC. VIA           00005250
                      ORDER I.D.                                        00005260
  12        FMRWB     DISK ERROR WHILE READING ORDER REC. VIA           00005270
                      REL. ORDER NO.                                    00005280
  13        FMRWB     DISK ERROR WHILE CODE START FILE INIT.            00005290
  14        FMRWB     DISK ERROR WHILE DELETING TRANS. FILE REC.        00005300
  15        FMATR     INVALID COMPLETION CODE RETURNED                  00005310
  16        FMDEL     BATCH DELETION QUEUE FULL                         00005320
  17        FMRWB     INVALID INPUT PARAMETERS                          00005330
                                                                        00005340
                                                                        00005350
OPERATOR INTERFACE FATAL ERRORS                                         00005360
                                                                        00005370
ERROR #   MODULE    ERROR                                               00005380
                                                                        00005390
 101        OIWSR     IPM BUFFER QUEUE EMPTY (OICCGI)                   00005400
 102        OIWSR     IPM BUFFER QUEUE FULL (OICCFI)                    00005410
 103        OIWSR     SRB BUFFER QUEUE EMPTY (OICCGR)                   00005420
 104        OIWSR     SRB BUFFER QUEUE FULL (OICCFR)                    00005430
 105        OIWSR     INVALID CALL CODE (OICCGB)                        00005440
 106        OIWSR     INVALID COMPLETION CODE (OICCGB)                  00005450
 107        OIBSR     IPM BUFFER QUEUE EMPTY (OICCGI)                   00005460
 108        OIBSR     IPM BUFFER QUEUE FULL (OICCFI)                    00005470
 109        OIBSR     SRB BUFFER QUEUE EMPTY (OICCGR)                   00005480
 110        OIBSR     SRB BUFFER QUEUE FULL (OICCFR)                    00005490
 111        OIBSR     INVALID CALL CODE (OICCGB)                        00005500
 112        OIBSR     INVALID COMPLETION CODE (OICCGB)                  00005510
 113        OIBSR     INVALID CALL CODE (OICCFO)                        00005520
 114        OIBSR     INVALID RELATIVE ORDER NUMBER (OICCFO)            00005530
 115        OIBSR     INVALID COMPLETION CODE (OICCFO)                  00005540
 116        OIDSR     IPM BUFFER QUEUE EMPTY (OICCGI)                   00005550
 117        OIDSR     IPM BUFFER QUEUE FULL (OICCFI)                    00005560
 118        OIDSR     SRB BUFFER QUEUE EMPTY (OICCGR)                   00005570
 119        OIDSR     SRB BUFFER QUEUE FULL (OICCFR)                    00005580
 120        OIDSR     INVALID CALL CODE (TRANS. DIR)                    00005590
 121        OIDSR     SBB BUFFER QUEUE FULL (OICCFS)                    00005600
 122        OIDSR     INVALID CALL CODE (OICCGB)                        00005610
 123        OIDSR     INVALID COMPLETION CODE (OICCGB)                  00005620
 124        OIDSR     INVALID CALL CODE (OICCFO)                        00005630
 125        OIDSR     INVALID RELATIVE ORDER NUMBER (OICCFO)            00005640
 126        OIDSR     INVALID COMPLETION CODE (OICCFO)                  00005650
 127        OIDSR     INVALID CALL CODE (OICCGO)                        00005660
 128        OIDSR     INVALID COMPLETION CODE (OICCGO)                  00005670
 129        OIDSR     INVALID CALL CODE (OICCST)                        00005680
 130        OIDSR     INVALID RELATIVE TRANSACTION NUMBER (OICCST       00005690
 131        OIDSR     INVALID COMPLETION CODE (OICCST)                  00005700
 132        OITSR     IPM BUFFER QUEUE EMPTY (OICCGI)                   00005710
 133        OITSR     IPM BUFFER QUEUE FULL (OICCFI)                    00005720
 134        OITSR     SRB BUFFER QUEUE EMPTY (OICCGR)                   00005730
```

```
135   OITSR   SRB BUFFER QUEJE FULL (OICCFR)                          00005740
136   OITSR   INVALID CALL CODE(TRANS DIR)                           00005750
137   OITSR   INVALID COMPLETION CODE(TRANS DIR)                     00005760
138   OITSR   INVALID CALL CODE (OICCGB)                             00005770
139   OITSR   INVALID COMPLETION CODE (OICCGB)                       00005780
140   OITSR   INVALID CALL CODE (OICCGT)                             00005790
141   OITSR   INVALID COMPLETION CODE (OICCGT)                       00005800
142   OISTB   IPM BUFFER QUEJE EMPTY (OICCGI)                        00005810
143   OISTB   IPM BUFFER QUEUE FULL (OICCFI)                         00005820
144   OISTB   SRB BUFFER QUEJE EMPTY (OICCGR)                        00005830
145   OISTB   SRB BUFFER QUEJE FJLL (OICCFR)                         00005840
146   OISTB   INVALID CALL CODE (OICCGB)                             00005850
147   OISTB   INVALID COMPLETION CODE (OICCGB)                       00005860
148   OISTB   INVALID CALL CODE (OICCCB)                             00005870
149   OISTB   INVALID STATUS CODE (OICCCB)                           00005880
150   OISTB   INVALID COMPLETION CODE (OICCCB)                       00005890
151   OISTB   ALARM QJEUE FULL (OICCSA)                              00005900
152   OISTB   ALARM BJFFER EMPTY (OICCSA)                            00005910
153   OIENB   IPM BUFFER QUEUE EMPTY (OICCGI)                        00005920
154   OIENB   IPM BUFFER QUEJE FULL (OICCFI)                         00005930
155   OIENB   SRB BUFFER QUEJE EMPTY (OICCGR)                        00005940
156   OIENB   SRB BUFFER QUEJE FJLL (OICCFR)                         00005950
157   OIENB   INVALID CALL CODE (OICCGB)                             00005960
158   OIENB   INVALID COMPLETION CODE (OICCGB)                       00005970
159   OIENB   INVALID CALL CODE (OICCCB)                             00005980
160   OIENB   INVALID STATUS CODE (OICCCB)                           00005990
161   OIENB   INVALID COMPLETION CODE (OICCCB)                       00006000
162   OIENB   ALARM QJEUE FULL (OICCSA)                              00006010
163   OIENB   ALARM BJFFER QJEUE EMPTY (OICCSA)                      00006020
164   OIASL   DISK WRITE ERROR SORT TABLE (OICCWS)                   00006030
165   OIRSL   DISK WRITE ERROR SORT TABLE (OICCWS)                   00006040
166   OIRDB   IPM BUFFER QUEJE EMPTY (OICCGI)                        00006050
167   OIRDB   IPM BUFFER QUEJE FJLL (OICCFI)                         00006060
168   OIRDB   SRB BUFFER QUEJE EMPTY (OICCGR)                        00006070
169   OIRDB   SRB BUFFER QUEJE FULL (OICCFR)                         00006080
170   OIRDB   BATCH ALREADY BEING ADDED (OICCAB)                     00006090
171   OIRDB   INVALID CALL CODE (OICCAB)                             00006100
172   OIRDB   INVALID COMPLETION CODE (OICCAB)                       00006110
173   OIRDB   INVALID CALL CODE (OICCAT)                             00006120
174   OIRDB   INVALID COMPLETION CODE (OICCAT)                       00006130
175   OIWRB   SRB BUFFER QUEJE EMPTY (OICCGR)                        00006140
176   OIRDB   INVALID CALL CODE (OICCRB)                             00006150
177   OIRDB   INVALID COMPLETION CODE (OICCRB)                       00006160
178   OIRDB   INVALID RELATIVE BATCH NUMBER (OICCDB)                 00006170
179   OIRDB   INVALIDE CALL CODE (OICCDB)                            00006180
180   OIRDB   INVALID COMPLETION CODE (OICCDB)                       00006190
181   OIPSC   PSC QUEJE FULL                                         00006200
182   OIPSC   PSC BUFFER QUEJE EMPTY                                 00006210
183   OIATS   INVALID CALL CODE (OICCGB)                             00006220
184   OIATS   INVALID COMPLETION CODE (OICCGB)                       00006230
185   OIATS   IPM BUFFER QUEJE EMPTY (OICCGI)                        00006240
186   OIATS   IPM BUFFER QUEJE FJLL (OICCFI)                         00006250
187   OIATS   SRB BUFFER QUEJE EMPTY (OICCGR)                        00006260
188   OIATS   SRB BUFFER QUEJE FULL (OICCFR)                         00006270
189   OIATS   INVALID CALL CODE (OICCGT)                             00006280
190   OIATS   INVALID COMPLETION CODE (OICCGT)                       00006290
191   OIATS   INVALID CALL CODE (TRANS STATUS)                       00006300
192   OIATS   INVALID STATUS CODE (TRANS STATUS)                     00006310
193   OIATS   INVALID COMPLETION CODE (TRANS STATUS)                 00006320
194   OIWRB   IPM BUFFER QUEJE EMPTY (OICCGI)                        00006330
195   OIWRB   IPM BUFFER QUEJE FULL (OICCFI)                         00006340
196   OIMSR   IPM BUFFER QUEJE EMPTY (OICCGI)                        00006350
197   OIWRB   SBB BUFFER QUEJE FJLL (OICCFS)                         00006360
198   OIWRB   SRB BUFFER QUEJE FJLL (OICCFR)                         00006370
199   OIWRB   INVALID CALL CODE (OICCWB)                             00006380
200   OIWRB   INVALID COMPLETION CODE (OICCWB)                       00006390
601   OIWRB   INVALID CALL CODE (OICCFO)                             00006400
602   OIWRB   INVALID RELATIVE ORDER NUMBER (OICCFO)                 00006410
603   OIWRB   INVALID COMPLETION CODE (OICCFO)                       00006420
604   OIWRB   INVALID CALL CODE (OICCST)                             00006430
605   OIWRB   INVALID RELATIVE TRANSACTION NUMBER(OICCST)            00006440
606   OIWRB   INVALID COMPLETION CODE (OICCST)                       00006450
607   OIWRB   INVALID CALL CODE (OICCGB)                             00006460
608   OIWRB   INVALID COMPLETION CODE (OICCGB)                       00006470
```

```
  609     OIDLB     IPM BUFFER QUEJE EMPTY (OICCGI)                    00006480
  610     OIDLB     IPM BUFFER QUEJE FULL (OICCFI)                     00006490
  611     OIDLB     SRB BUFFER QUEUE EMPTY (OICCGR)                    00006500
  612     OIDLB     SRB BUFFER QUEUE FULL (OICCFR)                     00006510
  613     OIDLB     INVALID CALL CODE (OICCGB)                         00006520
  614     OIDLB     INVALID COMPLETION CODE (OICCGB)                   00006530
  615     OIDLB     INVALID RELATIVE BATCH NUMBER (OICCDB)             00006540
  616     OIDLB     INVALID CALL CODE (OICCDB)                         00006550
  617     OIDLB     INVALID COMPLETION CODE (OICCDB)                   00006560
  618     OICTL     INVALID LOAD CODE (LOADIT)                         00006570
  619     OIHLT     ALARM BUFFER QUEUE EMPTY                           00006580
  620     OIHLT     ALARM MESSAGE QUEJE FULL                           00006590
  621     OIREC     ALARM BUFFER QUEUE EMPTY                           00006600
  622     OIREC     ALARM MESSAGE QUEUE FULL                           00006610
  623     FMATR     IPM BUFFER QUEJE EMPTY                             00006620
  624     FMATR     ERROR RETURNED FROM F.M. WHILE CHANGING            00006630
                    BATCH STATUS TO PENDING.                           00006640
  625     FMATR     IPM BUFFER QUEJE FULL                              00006650
  626     FMATR     ERROR RETURNED FROM F.M. WHILE DELETING            00006660
                    A BATCH.                                           00006670
  627     OITSR     INVALID RELATIVE ORDER                             00006680
  628     OITSR     INVALID FM CALL CODE                               00006690
  629     OIMSR     IPM BUFFER QUEJE FULL (OICCFI)                     00006700
  630     OIMSR     SRB BUFFER QUEJE EMPTY (OICCGR)                    00006710
  631     OIMSR     SRB BUFFER QUEJE FULL (OICCFR)                     00006720
  632     OIMSR     SBB BUFFER QUEJE FULL (OICCFS)                     00006730
  633     OIMSR     INVALID CALL CODE (OICCGB)                         00006740
  634     OIMSR     INVALID COMPLETION CODE (OICCGB)                   00006750
  635     OIMSR     INVALID CALL CODE (OICCFO)                         00006760
  636     OIMSR     INVALID RELATIVE ORDER NUMBER (OICCFO)             00006770
  637     OIMSR     INVALID COMPLETION CODE (OICCFO)                   00006780
  638     OIMSR     INVALID CALL CODE (OICCST)                         00006790
  639     OIMSR     INVALID RELATIVE TRANSACTION NUMBER (OICCST00006800
  640     OIMSR     INVALID COMPLETION CODE (OICCST)                   00006810
  641     OIMSR     INVALID CALL CODE (OICCGO)                         00006820
  642     OIMSR     INVALID COMPLETION CODE (OICCGO)                   00006830
  643     OIMSR     INVALID CALL CODE (TRANS. DIR)                     00006840
                                                                       00006850
ALARM MESSAGE FATAL ERRORS                                             00006860
                                                                       00006870
ERROR #   MODULE    ERROR                                            00006880
                                                                       00006890
  201     ALMPG     BAD MESSAGE NUMBER                                 00006900
  202     ALMPG     BUFFER QUEUE FULL                                  00006910
                                                                       00006920
SCANNER INPUT FATAL ERRORS                                             00006930
                                                                       00006940
ERROR #   MODULE    ERROR                                            00006950
                                                                       00006960
  301     SCIND     DIRECTORY READ ERROR                               00006970
  302     SCIND     BUFFER QUEJE EMPTY (ALARM)                         00006980
  303     SCIND     PSC OUTPUT QUEUE FULL                              00006990
  304     SCIND     PSC BUFFER QUEJE EMPTY                             00007000
  305     SCIND     STATUS UPDATE QUEUE FULL                           00007010
  306     SCIND     STATUS JPDATE BUFFER QUEUE EMPTY                   00007020
  307     SCTSU     ERROR RETURNED FROM FILE MANAGER                   00007030
  308     SCTSU     STATUS JPDATE BUFFER QJEUE FULL                    00007040
  309     SCTSU     ALARM QJEUE FULL                                   00007050
  310     SCTSU     ALARM BUFFER QJEUE EMPTY                           00007060
  311     SCTSU     FILE MANAGER BJFFER QUEJE FJLL                     00007070
  312     SCTSU     FILE MANAGER BJFFER QUEUE EMPTY                    00007080
  313     SCSTO     STATUS JPDATE QUEUE FULL                           00007090
          SCSHP                                                        00007100
          SCXRP                                                        00007110
  314     SCSTO     STATUS UPDATE BUFFER QUEUE EMPTY                   00007120
          SCSHP                                                        00007130
          SCXRP                                                        00007140
  315     SCSTO     ALARM QJEUE FULL                                   00007150
          SCSHP                                                        00007160
          SCXRP                                                        00007170
  316     SCSTO     ALARM BUFFER QJEJE EMPTY                           00007180
          SCSHP                                                        00007190
          SCXRP                                                        00007200
```

```
  317      SCIND     ALARM QUEUE FULL                                  00007210
  318      SCSR1     INDUCTION QUEUE FULL                              00007220
  319      SCSR1     INDUCTION BUFFER QUEUE EMPTY                      00007230
  320      SCSR2     INDUCTION QUEUE FULL                              00007240
  321      SCSR2     INDUCTION BUFFER QUEUE EMPTY                      00007250
  322      SCSR3     INDUCTION QUEUE FULL                              00007260
  323      SCSR3     INDUCTION BUFFER QUEUE EMPTY                      00007270
  324      SCIND     INDUCTION BUFFER QUEUE FULL                       00007280
                                                                       00007290
PSC COMMUNICATIONS FATAL ERRORS                                        00007300
                                                                       00007310
ERROR #    MODULE    ERROR                                           00007320
                                                                       00007330
  401      PSOC      ALARM MESSAGE QUEUE FULL                        00007340
  402      PSOC      ALARM MSG. BUFFER QUEUE EMPTY                   00007350
  403      PSIO      PSC RESPONSE QUEUE FULL                         00007360
  404      PSIO      PSC RESPONSE BUFFER QUEUE EMPTY                 00007370
  405      PSRP      F.M. BUFFER QUEUE FULL                          00007380
  406      PSRP      ALARM MESSAGE QUEUE FULL                        00007390
  407      PSRP      ALARM MESSAGE BUFFER QUEUE EMPTY                00007400
  408      PSRP      UNEXPECTED F.M. COMPLETION CODE RETURNED        00007410
  409      PSRP      F.M. BUFFER QUEUE EMPTY                         00007420
  410      PSRP      PSC RESPONSE BUFFER QUEUE FULL                  00007430
  411      PSPOL     PSC OUTPUT QUEUE FULL                           00007440
  412      PSPOL     PSC OUTPUT BUFFER QUEUE EMPTY                   00007450
  413      PSOC      PSC OUTPUT BUFFER QUEUE FULL                    00007460
  414      PSOC      PSC MESSAGE FORMATTING ERROR FOUND              00007470
  415      PSAO      ALARM MESSAGE BUFFER QUEUE FULL                 00007480
  416      PSAO      ALARM MESSAGE QUEUE FULL                        00007490
  417      PSAO      ALARM MESSAGE BUFFER QUEUE EMPTY                00007500
  418      PSIO      ALARM MESSAGE BUFFER QUEUE EMPTY                00007510
  419      PSIO      ALARM MESSAGE QUEUE FULL                        00007520
  420                NOT USED                                        00007530
  421      PSRP      TRANS. NO. CONVERSION ERROR                     00007540
                                                                       00007550
AUXILIARY FUNCTIONS FATAL ERRORS                                       00007560
                                                                       00007570
ERROR #    MODULE    ERROR                                           00007580
                                                                       00007590
  501      $INIT     FILE MANAGER SUB-SYSTEM LOAD ERROR              00007600
  502      $INIT     FILE MANAGER BUFFER QUEUE EMPTY                 00007610
  503      $INIT     UNEXPECTED F.M. COMPLETION CODE RECEIVED        00007620
  504      $INIT     FILE MANAGER BUFFER QUEUE FULL                  00007630
  505      $INIT     PSC COMM. SUB-SYSTEM LOAD ERROR                 00007640
  506      $INIT     SCANNER INPUT SUB-SYSTEM LOAD ERROR             00007650
  507      $INIT     OPERATOR INTERFACE SUB-SYSTEM LOAD ERROR        00007660
  508      $INIT     ALARM MESSAGE BUFFER QUEUE EMPTY                00007670
  509      $INIT     ALARM MESSAGE QUEUE FULL                        00007680
                                                                       00007690
09AL   TIME OUT TASK (TIMRTSK)                                        00007700
                                                                       00007710
       WHEN THE OUTPUT TASKS ARE STARTED A TIME OUT TASK IS ALSO       00007720
       STARTED. IF THE TIME OUT TASK TIMES OUT THE OUTPUT TASKS,       00007730
       IT IS ASSUMED THAT THERE IS A HARDWARE PROBLEM. TO AVOID        00007740
       HANGING UP THE OTHER ALARM MESSAGES THE PRINTEXTS ARE TER-      00007750
       MINATED THROUGH POSTS OF THE PRINT ECB'S OF EACH OUTPUT TASK.   00007760
                                                                       00007770
10AL   DATE AND TIME SUBROUTINE (DATES)                               00007780
                                                                       00007790
       THIS SUBROUTINE IS CALLED TO PUT THE DATE AND TIME INTO THE     00007800
       ALARM OUTPUT BUFFER. THE SUBROUTINE USES THE FIRST 18 BYTES     00007810
       OF THE BUFFER FOR THE DATE AND TIME. THE FORMAT IS:             00007820
                                                                       00007830
             . TIME    DATE                                          00007840
             XX:XX:XX XX/XX/XX                                         00007850
                                                                       00007860
11AL   MESSAGE COMPRESS FLAG                                          00007870
                                                                       00007880
       THIS FLAG (CRFLS) IS USED TO INDICATE IF A MESSAGE SHOULD BE    00007890
       COMPRESSED OR NOT.  IF THE FLAG IS A '1', THE MESSAGE IS COM-   00007900
       PRESSED. IF NOT, IT'S NOT.                                      00007910
                                                                       00007920
12AL   NON-COMPRESSABLE MESSAGES                                      00007930
                                                                       00007940
```

```
        NON-COMPRESSABLE MESSAGES ARE THOSE DISPLAYING BLANKS THAT ARE 00007950
        SIGNIFIGANT.  THESE MESSAGES ARE THOSE CONTAINING DATA SENT    00007960
        FROM THE PSC.                                                  00007970
                                                                       00007980
13AL    MESSAGE COMPRESSING SUBROUTINE (MOVTXT)                        00007990
                                                                       00008000
        THIS SUBROUTINE WILL COMPRESS A STRING OF BLANKS IN AN ALARM   00008010
        MESSAGE DOWN TO A SINGLE BLANK.  ALL OCCURRENCES OF BLANKS     00008020
        IN THE MESSAGES ARE COMPRESSED.                                00008030
                                                                       00008040


*********************************************************************00000010
*                                                                     00000020
* FILE MANAGER DATA DICTIONARY INDEX                                  00000030
*                                                                     00000040
*********************************************************************00000050
                                                                       00000060
I/O #              DESCRIPTION                                         00000070
                                                                       00000080
01FM    BATCH FILE DEFINITION                                          00000090
02FM    RESIDENT BATCH FILE DEFINITION                                 00000100
03FM    TRANSACTION DIRECTORY DEFINITION                               00000110
04FM    TRANSACTION FILE DEFINITION                                    00000120
05FM    ORDER FILE DEFINITION                                          00000130
06FM    SORT TRANSLATION FILE DEFINITION                               00000140
07FM    DISKETTE FILE DEFINITION                                       00000150
08FM    FILE MANAGER USER CALLING SEQUENCE                             00000160
09FM    FILE MANAGER INPUT BUFFER FORMAT                               00000170
10FM    FILE MANAGER CALL CODES                                        00000180
11FM    FILE MANAGER COMPLETION CODES                                  00000190
12FM    ADD BATCH RECORD CALL FORMAT                                   00000200
13FM    ADD TRANSACTION RECORD CALL FORMAT                             00000210
14FM    DELETE BATCH CALL FORMAT                                       00000220
15FM    CHANGE BATCH STATUS CALL FORMAT                                00000230
16FM    CHANGE TRANSACTION STATUS FROM TERMINAL CALL FORMAT            00000240
17FM    CHANGE TRANSACTION STATUS FROM SCANNERS CALL FORMAT            00000250
18FM    READ BATCH RECORD VIA BATCH I.D. CALL FORMAT                   00000260
19FM    READ TRANSACTION RECORD VIA TRANSACTION NO. CALL FORMAT        00000270
20FM    READ TRANSACTION DIRECTORY RECORD VIA TRANS. NO. CALL FORMAT   00000280
21FM    READ TRANSACTION FILE BLOCK CALL FORMAT                        00000290
22FM    READ ORDER RECORD VIA ORDER I.D. CALL FORMAT                   00000300
23FM    READ ORDER RECORD VIA RELATIVE ORDER NO. CALL FORMAT           00000310
24FM    COLD START FILE INITIALIZATION CALL FORMAT                     00000320
25FM    FILE MANAGER ENTRY EVENT                                       00000330
26FM    CALL CODE DISPATCH TABLE                                       00000340
27FM    FILE MANAGER COMPLETION EVENT                                  00000350
28FM    READ/WRITE FILE BLOCK                                          00000360
29FM    ORDER I.D. TABLE                                               00000370
30FM    NOT USED                                                       00000380
31FM    BATCH ADDITION COUNTS                                          00000390
32FM    BATCH DELETION QUEUE                                           00000400
33FM    RESIDENT BATCH FILE SEARCH                                     00000410
34FM    FILE MANAGER RESOURCE DEFINITION                               00000420
35FM    FILE MANAGER PANIC CALL                                        00000430
36FM    RECORD BUFFER QUEUE                                            00000440
37FM    SECTOR BUFFER QUEUE                                            00000450
38FM    ADD BATCH COMPLETE                                             00000460
39FM    SYSTEM PARAMETER FILE DEFINITION                               00000470
40FM    SYSTEM ERROR COUNTS                                            00000480


*********************************************************************00000490
*                                                                     00000500
* FILE MANAGER DATA REFERENCES                                        00000510
*                                                                     00000520
*********************************************************************00000530
                                                                       00000540
I/O #              MODULES REFERENCING                                 00000550
                                                                       00000560
01FM    FMATR,FMABR,FMDEL,FMDID,FMBID,FMCBS,FMRTB,FMINT,FMROL,DISCGB   00000570
        FMOLD,FMADC,DIATS,DIBSR,DISTB,$INITIAL                         00000580
```

```
02FM    FMATR,FMABR,FMDEL,FMCBS,FMINT,FMDLD,FMCTS,FMDRB,SCIND,FMADC,      00000590
        DIRDB,DITSR,DIWSR,DICCFB                                          00000600
03FM    FMATR,FMCTS,FMTID,FMRTD,SCIND,FMTSR                               00000610
04FM    FMATR,FMCTS,FMRTB,FMTID,SCTSU,SCIND,FMDLD,FMRTD,DICCFT,DICCGT,    00000620
        $INITIAL                                                          00000630
05FM    FMATR,FMCTS,FMTRB,FMTID,SCTSU,DICCFD,DICCGD,FMADC,$INITIAL        00000640
06FM    FMINT,SCIND,DIASL,DICCWS,DIRSL,DISTR,AXCNFG                       00000650
07FM    AXBLDD,DIRDB,DIWRB                                                00000660
08FM    FM,PSRP,SCTSU,$INIT,FMATR,DIATS,DITSR,DICCAB,DICCAT,DICCCB,       00000670
        DICCDB,DICCFD,DICCGB,DICCGD,DICCGT,DICCST,DIOSR                   00000680
09FM    FMGR,FMRWB,PSRP,$INIT,FMATR,DICCFI,DICCGI,SCTSU                   00000690
10FM    SCTSU                                                             00000700
11FM    SCTSU                                                             00000710
12FM    FMABR,DICCAB                                                      00000720
13FM    FMATR,DICCAT                                                      00000730
14FM    FMATR,FMDEL,DICCDB                                                00000740
15FM    FMCBS,DICCCB                                                      00000750
16FM    FMCTS,DIATS                                                       00000760
17FM    FMCTS,PSRP,SCTSU                                                  00000770
18FM    FMBID,DICCGB                                                      00000780
19FM    FMTID,DICCGT                                                      00000790
20FM    FMRID,DITSR,DIOSR                                                 00000800
21FM    FMRTB,DICCST                                                      00000810
22FM    FMDID,DICCGD                                                      00000820
23FM    FMPDL,DICCFD                                                      00000830
24FM    FMINT,$INIT                                                       00000840
25FM    FMGR,FM                                                           00000850
26FM    FMGR                                                              00000860
27FM    FMGR,FM                                                           00000870
28FM    FMABR,FMDEL,FMDID,FMBID,FMCBS,FMCTS,FMRTB,FMINT                   00000880
        FMTID,FMRDL,FMRTD,FMRWB,FMADC                                     00000890
29FM    FMATR                                                             00000900
30FM                                                                      00000910
31FM    FMABR                                                             00000920
32FM    FMDEL,FMINT,FMDLD                                                 00000930
33FM    FMDID,FMBID,FMCBS,FMRTB,FMRDL,FMSRB                               00000940
34FM    FMDLD,FM                                                          00000950
35FM    FMPAN,FMATR,FMDEL,FMRWB                                           00000960
36FM    DICCGR,DICCFR                                                     00000970
37FM    DICCGS,DICCFS                                                     00000980
38FM    FMADC,FMATR                                                       00000990
39FM    $INITIAL,PSOC,DIASL,DIRDB,DIRSL,DISTR,AXCNFG,SCINIT,ALMPG,DISER00001000
40FM    DISER,DIRSE,SCIND,PSRP                                            00001010
```

```
*************************************************************************00001020
*                                                                       00001030
* FILE MANAGER DATA DICTIONARY ENTRIES                                  00001040
*                                                                       00001050
*************************************************************************00001060
                                                                        00001070
I/O #                DESCRPTION                                         00001080
                                                                        00001090
01FM    BATCH FILE DEFINITION                                           00001100
                                                                        00001110
        THE BATCH FILE CONTAINS 64 BYTE RECORDS WITH                    00001120
        4 RECORDS PER SECTOR. TOTAL FILE FOR 11 BATCHES                 00001130
        OCCUPIES 3 SECTORS. THIS FILE RESIDES ON DISK AS                00001140
        A CONTIGUOUS FILE CALLED "BATCH". THE LAYOUT OF                 00001150
        EACH 64 BYTE RECORD FOLLOWS.                                    00001160
                                                                        00001170
        INDEX        FIELD              SIZE (BYTES)        TYPE        00001180
                                                                        00001190
        IBSTA   BATCH STATUS                 1             BINARY      00001200
        IBBID   BATCH I.D.                   3             EBCDIC      00001210
        IBHDR   BATCH HEADER                50             EBCDIC      00001220
        IBORD   NO. OF ORDERS                2             BINARY      00001230
        IBXAC   NO. OF TRANSACTIONS          2             BINARY      00001240
                SPARE                        6                         00001250
                                                                        00001260
        NOTE: VALID BATCH STATUS VALUES FOLLOW:                         00001270
                                                                        00001280
```

```
        EQJ     MEANING                                                 00001290
                                                                        00001300
        BADD    ADDING                                                  00001310
        BDEL    DELETION                                                00001320
        BFRE    UNUSED RECORD                                           00001330
        BPEN    PENDING                                                 00001340
        BACT    ACTIVE                                                  00001350
        BCMP    COMPLETE                                                00001360
                                                                        00001370
02FM    RESIDENT BATCH FILE DEFINITION                                  00001380
                                                                        00001390
        THE RESIDENT BATCH FILE CONTAINS 4 BYTE RECORDS.                00001400
        TOTAL SPACE ALLOCATED FOR 11 BATCHES IS 44 BYTES.               00001410
        THIS FILE RESIDES IN $SYSCOM AS A CONTIGUOUS                    00001420
        FILE. THE LAYOUT FO EACH 4 BYTE RECORD FOLLOWS:                 00001430
                                                                        00001440
        INDEX       FIELD              SIZE (BYTES)      TYPE           00001450
                                                                        00001460
        IBSTA   BATCH STATUS                1           BINARY         00001470
        IBBID   BATCH I.D.                  3           EBCDIC         00001480
                                                                        00001490
        NOTE: VALID BATCH STATUS VALUES FOLLOW:                         00001500
                                                                        00001510
        -EQU-   -MEANING-                                               00001520
                                                                        00001530
        BADD    ADDING                                                  00001540
        BDEL    DELETING                                                00001550
        BFRE    UNUSED RECORD                                           00001560
        BPEN    PENDING                                                 00001570
        BACT    ACTIVE                                                  00001580
        BCMP    COMPLETE                                                00001590
                                                                        00001600
03FM    TRANSACTION DIRECTORY DEFINITION                                00001610
                                                                        00001620
        TRANSACTION DIRECTORY CONTAINS 4 BYTE RECORDS WITH              00001630
        64 RECORDS PER SECTOR. TOTAL DIRECORY FOR                       00001640
        999,999 ENTRIES OCCUPIES 15,625 SECTORS. THIS                   00001650
        DIRECTORY RESIDES ON DISK AS A CONTIGUOUS FILE                  00001660
        CALLED "XDIR". THE LAYOUT OF EACH 4 BYTE RECORD                 00001670
        FOLLOWS:                                                        00001680
                                                                        00001690
        INDEX       FIELD              SIZE (BYTES)      TYPE           00001700
                                                                        00001710
        IDRLX   REL. TRANSACTION NO.        2           BINARY         00001720
        IDRLB   REL. BATCH NO.              1           BINARY         00001730
        IDDST   DESTINATION                 1           BINARY         00001740
                                                                        00001750
        NOTE: IF RECORD IS UNUSED, ALL BITS ARE SET.                    00001760
                                                                        00001770
04FM    TRANSACTION FILE DEFINITION                                     00001780
                                                                        00001790
        THE TRANSACTION FILE CONTAINS 18 BYTE RECORDS WITH              00001800
        14 RECORDS PER SECTOR. EACH BATCH MAY CONTAIN 9,002             00001810
        RECORDS (643 SECTORS). THIS FILE RESIDES ON DISK                00001820
        AS A CONTIGUOUS FILE CALLED "XACT" CONTAINING SPACE             00001830
        FOR 11 BATCHES FOR A TOTAL OF 7,073 SECTORS.                    00001840
        THE LAYOUT OF EACH 18 BYTE RECORD FOLLOWS:                      00001850
                                                                        00001860
        INDEX       FIELD              SIZE (BYTES)      TYPE           00001870
                                                                        00001880
        IXXNO   TRANSACTION NO.             4           BINARY         00001890
        IXSTA   STATUS                      1           BINARY         00001900
        IXRLO   RELATIVE ORDER NO.          1           BINARY         00001910
        IXPRD   PRODUCT                     6           EBCDIC         00001920
        IXDST   DESTINATION                 1           BINARY         00001930
        IXSRC   SOURCE                      5           EBCDIC         00001940
                                                                        00001950
        NOTE: VALID TRANSACTION STATUS VALUES FOLLOW:                   00001960
                                                                        00001970
        -EQU-   -MEANING-                                               00001980
                                                                        00001990
        XNPK    NOT PICKED                                              00002000
        XINS    IN-SORTATION                                            00002010
```

```
        XSTG    STAGED FOR SHIPPING                                     00002020
        XSTO    STOCKED-OUT                                             00002030
        XXSR    EXCESS REPACK                                           00002040
                                                                        00002050
05FM    ORDER FILE DEFINITION                                           00002060
                                                                        00002070
        ORDER FILE CONTAINS 20 BYTE RECORDS WITH 12 RECORDS             00002080
        PER SECTOR. TOTAL FILE FOR 36 RECORDS PER BATCH WITH            00002090
        SPACE ALLOCATED FOR 11 BATCHES OCCUPIES 33 SECTORS.             00002100
        THIS FILE RESIDES ON DISK AS A CONTIGUOUS FILE CALLED           00002110
        "ORDER". THE LAYOUT OF EACH 20 BYTE RECORD FOLLOWS.             00002120
                                                                        00002130
        INDEX   FIELD                   SIZE (BYTES)        TYPE        00002140
                                                                        00002150
        IORLO   RELATIVE ORDER NO.          2               BINARY      00002160
        IOOID   ORDER I.D.                  6               EBCDIC      00002170
        IOQNP   QTY. NOT PICKED             2               BINARY      00002180
        IOQIS   QTY. IN SORTATION           2               BINARY      00002190
        IOQSS   QTY. STAGED FOR SHIP.       2               BINARY      00002200
        IOQXS   QTY. EXCESS REPACK          2               BINARY      00002210
        IOQSO   QTY. STOCK-OUT              2               BINARY      00002220
        IOQMS   QTY. MIS-SORT               2               BINARY      00002230
                                                                        00002240
06FM    SORT TRANSLATION FILE DEFINITION                                00002250
                                                                        00002260
        THE SORT TRANSLATION FILE CONSISTS OF 1 SECTOR                  00002270
        OF 256 BYTES ON DISK CALLED "XLATE" OF WHICH THE FIRST 100      00002280
        BYTES ARE USED. THESE ENTRIES CORRESPOND DIRECTLY TO THE        00002290
        SORT DESTINATION VALUES 0 THRU 99. (DESTINATION 0 IS            00002300
        NOT USED) NORMALLY THESE ENTRIES ARE EACH ZERO.                 00002310
        IF A TRANSLATION IS IN EFFECT, THE NEW (NON-ZERO)               00002320
        DESTINATION (1-99) IS PLACED IN THE RELATIVE BYTE               00002330
        CORRESPONDING TO THE OLD DESTINATION.                           00002340
                                                                        00002350
        ON IPL, THE DISK-RESIDENT SORT TRANSLATION FILE                 00002360
        DATA IS COPIED INTO $SYSCOM AS A 50 WORD (100                   00002370
        BYTE) MEMORY RESIDENT TABLE. SUBSEQUENT UPDATES                 00002380
        TO THIS DATA ARE MADE TO THE MEMORY-RESIDENT                    00002390
        AND DISK-RESIDENT FILES.                                        00002400
                                                                        00002410
07FM    DISKETTE FILE DEFINITION                                        00002420
                                                                        00002430
        EACH DISKETTE CONTAINING DATA TO BE READ BY THE                 00002440
        DAS OR DATA WRITTEN BY THE DAS FOLLOWS THE IBM                  00002450
        STANDARD FOR INFORMATION INTERCHANGE (ONE-                      00002460
        SIDED, 128 BYTE SECTORS). EACH DISKETTE                         00002470
        FILE CONTAINS 24 BYTE RECORDS WITH 5                            00002480
        RECORDS PER SECTOR. RECORDS 1, 2, AND                           00002490
        3 HAVE UNIQUE LAYOUTS WITH RECORDS 4                            00002500
        THRU N HAVING AN IDENTICAL FORMAT. THESE                        00002510
        LAYOUTS FOLLOW:                                                 00002520
                                                                        00002530
        REC.#       FIELD               SIZE (BYTES)        TYPE        00002540
                                                                        00002550
          1     NO. OF RECS. IN FILE        4               BINARY      00002560
                BATCH I.D.                  3               EBCDIC      00002570
                HEADER (PART 1)            17               EBCDIC      00002580
                                                                        00002590
          2     HEADER (PART 2)            24               EBCDIC      00002600
                                                                        00002610
          3     HEADER (PART 3)             9               EBCDIC      00002620
                UNUSED                     15               EBCDIC      00002630
                                                                        00002640
        4-N     TRANSACTION NO.             4               BINARY      00002650
                ORDER I.D.                  6               EBCDIC      00002660
                PRODUCT                     5               EBCDIC      00002670
                DESTINATION                 2               EBCDIC      00002680
                SOURCE                      5               EBCDIC      00002690
                STATUS                      1               EBCDIC      00002700
                                                                        00002710
        NOTE: NO. OF RECS IN FILE IN RECORD 1 INCLUDES                  00002720
              RECORDS 1, 2, AND 3.                                      00002730
                                                                        00002740
```

```
08FM    FILE MANAGER USER CALLING SEQUENCE                              00002750
                                                                        00002760
        FOR ANY TASK TO USE THE FILE MANAGEMENT PROGRAM THE             00002770
        FOLLOWING CALLING SEQUENCE MUST BE USED:                        00002780
                                                                        00002790
                MOVE #1, BUFADR                                         00002800
                CALL FM                                                 00002810
                                                                        00002820
        WHERE BUFADR IS THE ADDRESS OF A 16 WORD BUFFER                 00002830
        CONTAINING INPUT DATA AND FM IS A SUBROUTINE                    00002840
        INCLUDED IN THE CALLING PROGRAM WHICH                           00002850
        PERFORMS ALL INTERFACE FUNCTIONS. NOTE:                         00002860
        #1 WILL STILL CONTAIN THE INPUT BUFFER                          00002870
        ADDRESS UPON RETURN FROM SUBROUTINE FM.                         00002880
                                                                        00002890
09FM    FILE MANAGER INPUT BUFFER FORMAT                                00002900
                                                                        00002910
                *     CALL CODE     *                                   00002920
                *  COMPLETION CODE  *                                   00002930
                *    PARAMETER 1    *                                   00002940
                *    PARAMETER 2    *                                   00002950
                *    PARAMETER 3    *  SIZE = 16 WORDS                  00002960
                *    PARAMETER 4    *                                   00002970
                *    PARAMETER N    *                                   00002980
                *                   *                                   00002990
                                                                        00003000
                FOR VALID CALL CODES SEE: 10FM                          00003010
                                                                        00003020
                                                                        00003030
        NOTE: ALL BUFFERS ARE TO BE OBTAINED FROM                       00003040
        A BUFFER QUEUE DEFINED IN $SYSCOM FOR                           00003050
        THIS PURPOSE                                                    00003060
                                                                        00003070
10FM    FILE MANAGER CALL CODES                                         00003080
                                                                        00003090
        FABR    ADD BATCH RECORD                                        00003100
        FATR    ADD TRANSACTION RECORD                                  00003110
        FDEL    DELETE BATCH                                            00003120
        FCBS    CHANGE BATCH STATUS                                     00003130
        FCTST   CHANGE TRANSACTION STATUS (FROM TERMINALS)              00003140
        FCTSS   CHANGE TRANSACTION STATUS (FROM SCANNERS)               00003150
        FBID    READ BATCH RECORD VIA BATCH I.D.                        00003160
        FTID    READ TRANSACTION RECORD VIA TRANSACTION NUMBER          00003170
        FRTD    READ TRANSACTION DIRECTORY RECORD VIA TRANSACTION NO.   00003180
        FRTB    READ TRANSACTION FILE BLOCK                             00003190
        FOID    READ ORDER RECORD VIA ORDER I.D.                        00003200
        FROL    READ ORDER RECORD VIA RELATIVE ORDER NO.                00003210
        FINT    COLD START FILE INITIALIZATION                          00003220
                                                                        00003230
11FM    FILE MANAGER COMPLETION CODES                                   00003240
                                                                        00003250
        FOK     SUCCESSFUL                                              00003260
        FBX     BATCH FILE SPACE EXHAUSTED                              00003270
        FTX     TRANSACTION FILE SPACE EXHAUSTED                        00003280
        FBA     BATCH ALREADY ON FILE                                   00003290
        FTA     TRANSACTION ALREADY ON FILE                             00003300
        FIS     INVALID STATUS                                          00003310
        FBN     BATCH NOT ON FILE                                       00003320
        FIR     INVALID RELATIVE RECORD/BLOCK NUMBER                    00003330
        FNB     TRANSACTION NOT IN BATCH                                00003340
        FTN     TRANSACTION NOT ON FILE                                 00003350
        FOX     ORDER FILE SPACE EXHAUSTED                              00003360
        FON     ORDER NOT ON FILE                                       00003370
        FCC     INVALID CALL CODE                                       00003380
        FAD     BATCH ALREADY BEING ADDED                               00003390
                                                                        00003400
12FM    CODE: FABR                      ADD BATCH RECORD                00003410
                                                                        00003420
        INPUT BUFFER LAYOUT:                                            00003430
                                                                        00003440
```

```
        WORD #                  CONTENTS                                00003450
                                                                        00003460
          1                     CALL CODE (FABR)                        00003470
          2                     IGNORE                                  00003480
          3                     $SYSCOM ADR. OF BATCH I.D. DATA         00003490
                                (SEE BELOW FOR DATA DEFINTION)          00003500
                                                                        00003510
RETURN BUFFER LAYOUT:                                                   00003520
                                                                        00003530
        WORD #                  CONTENTS                                00003540
                                                                        00003550
          1                     UNCHANGED                               00003560
          2                     COMPLETION CODE                         00003570
          3                     UNCHANGED                               00003580
                                                                        00003590
COMPLETION CODES:                                                       00003600
                                                                        00003610
        CODE                    MEANING                                 00003620
                                                                        00003630
        FOK                     SUCCESSFUL                              00003640
        FBX                     BATCH FILE SPACE EXHAUSTED              00003650
        FBA                     BATCH ALREADY ON FILE                   00003660
        FCC                     INVALID CALL CODE                       00003670
        FAD                     BATCH ALREADY BEING ADDED               00003680
                                                                        00003690
BATCH IDENTIFICATION DATA DEFINITION                                    00003700
                                                                        00003710
THE BATCH IDENTIFICATION DATA IS USED TO ADD A BATCH                    00003720
RECORD TO THE BATCH FILE. THE MEMORY ADDRESS OF THIS                    00003730
DATA IS PASSED TO THE FILE MANAGER IN THE ADD BATCH                     00003740
CALL. THE SIZE OF THIS DATA AREA IS 54 BYTES OF                         00003750
CONTIGUOUS MEMORY.  THE LAYOUT OF THIS DATA AREA                        00003760
FOLLOWS:                                                                00003770
                                                                        00003780
            FIELD               SIZE (BYTES)        TYPE                00003790
        SPARE                       1                                   00003800
        BATCH I.D.                  3               EBCDIC              00003810
        BATCH HEADER               50               EBCDIC              00003820
                                                                        00003830
13FM    CODE: FATR                  ADD TRANSACTION RECORD              00003840
                                                                        00003850
        INPUT BUFFER LAYOUT:                                            00003860
                                                                        00003870
        WORD #                  CONTENTS                                00003880
                                                                        00003890
          1                     CALL CODE (FATR)                        00003900
          2                     IGNORE                                  00003910
          3                     $SYSCOM ADR. OF TRANSACTION I.D         00003920
                                (SEE BELOW FOR DATA DEFINITION)         00003930
          4                     ZERO = FIRST/NEXT REC. NON-ZERO         00003940
                                = LAST REC.                             00003950
                                                                        00003960
        RETURN BUFFER LAYOUT:                                           00003970
                                                                        00003980
        WORD #                  CONTENTS                                00003990
                                                                        00004000
          1                     UNCHANGED                               00004010
          2                     COMPLETION CODE                         00004020
          3-4                   UNCHANGED                               00004030
                                                                        00004040
        COMPLETION CODES:                                               00004050
                                                                        00004060
        CODE                    MEANING                                 00004070
                                                                        00004080
        FOK                     SUCCESSFUL                              00004090
        FTA                     TRANSACTION ALREADY ON FILE             00004100
        FTX                     TRANSACTION FILE SPACE EXHAUSTE         00004110
        FOX                     ORDER FILE SPACE EXHAUSTED              00004120
        FCC                     INVALID CALL CODE                       00004130
                                                                        00004140
        NOTE: FTA, FTX, FOX CAUSE AN AUTO. DELETE OF                    00004150
              THE BATCH BEING ADDED.                                    00004160
                                                                        00004170
```

```
TRANSACTION IDENTIFICATION DATA DEFINITION                          00004180
                                                                    00004190
THE TRANSACTION IDENTIFICATION DATA IS USED TO                      00004200
ADD A TRANSACTION RECORD TO THE TRANSACTION FILE.                   00004210
THE MEMORY ADDRESS OF THIS DATA AREA IS PASSED TO                   00004220
THE FILE MANAGER IN THE ADD TRANSACTION CALL. THE                   00004230
SIZE OF THIS DATA AREA IS 24 BYTES OF CONTIGUOUS                    00004240
MEMORY. THE LAYOUT OF THIS DATA AREA FOLLOWS:                       00004250
                                                                    00004260
            FIELD              SIZE (BYTES)        TYPE             00004270
                                                                    00004280
        TRANSACTION NO.             4             BINARY           00004290
        ORDER I.D.                  6             EBCDIC           00004300
        PRODUCT I.D.                5             EBCDIC           00004310
        SORT DESTINATION            1             BINARY           00004320
        SOURCE                      5             EBCDIC           00004330
        STATUS                      2             BINARY           00004340
                                                                    00004350
14FM    CODE: FDEL                  DELETE BATCH                    00004360
                                                                    00004370
        INPUT BUFFER LAYOUT:                                        00004380
                                                                    00004390
            WORD #                  CONTENTS                        00004400
                                                                    00004410
              1                     CALL CODE (FDEL)                00004420
              2                     IGNORE                          00004430
             3-N                    RELATIVE BATCH NO(S) OF BATCH   00004440
                                    (ES) TO BE DELETED              00004450
             N+1                    ZERO                            00004460
                                                                    00004470
        RETURN BUFFER LAYOUT:                                       00004480
                                                                    00004490
            WORD #                  CONTENTS                        00004500
                                                                    00004510
              1                     UNCHANGED                       00004520
              2                     COMPLETION CODE                 00004530
            3-N+1                   UNCHANGED                       00004540
                                                                    00004550
        COMPLETION CODES:                                           00004560
                                                                    00004570
            CODE                    MEANING                         00004580
                                                                    00004590
            FOK                     SUCCESSFUL                      00004600
            FIR                     INVALID RELATIVE RECORD/        00004610
                                    BLOCK NUMBER                    00004620
            FCC                     INVALID CALL CODE .             00004630
                                                                    00004640
15FM    CODE: FCBS                  CHANGE BATCH STATUS             00004650
                                                                    00004660
        INPUT BUFFER LAYOUT:                                        00004670
                                                                    00004680
            WORD #                  CONTENTS                        00004690
                                                                    00004700
              1                     CALL CODE (FCBS)                00004710
              2                     IGNORE                          00004720
             3-4                    BATCH I.D.                      00004730
              5                     NEW STATUS (BPEN=PENDING,       00004740
                                    BACT=ACTIVE, BCMP=COMPLETE)     00004750
                                                                    00004760
        RETURN BUFFER LAYOUT:                                       00004770
                                                                    00004780
            WORD #                  CONTENTS                        00004790
                                                                    00004800
              1                     UNCHANGED                       00004810
              2                     COMPLETION CODE                 00004820
             3-5                    UNCHANGED                       00004830
                                                                    00004840
        COMPLETION CODES:                                           00004850
                                                                    00004860
            CODE                    MEANING                         00004870
                                                                    00004880
            FOK                     SUCCESSFUL                      00004890
            FIS                     INVALID STATUS                  00004900
```

```
            FBN     BATCH NOT ON FILE                                   00004910
            FCC     INVALID CALL CODE                                   00004920
                                                                        00004930
16FM        CODE: FCTST     CHANGE TRANSACTION STATUS                   00004940
                            FROM TERMINAL                               00004950
                                                                        00004960
            INPUT BUFFER LAYOUT:                                        00004970
                                                                        00004980
                    WORD #  CONTENTS                                    00004990
                                                                        00005000
                      1     CALL CODE                                   00005010
                      2     IGNORE                                      00005020
                    3-4     TRANSACTION NUMBER                          00005030
                    5-6     BATCH I.D.                                  00005040
                      7     STATUS (XNPK=NOT PICKED,XINS                00005050
                            =IN SORTATION,XSTO=STOCK-                   00005060
                            OUT,XXSR=EXCESS REPACK,XSTG                 00005070
                            =STAGED FOR SHIPMENT)                       00005080
                                                                        00005090
            RETURN BUFFER LAYOUT:                                       00005100
                                                                        00005110
                    WORD #  CONTENTS                                    00005120
                                                                        00005130
                      1     UNCHANGED                                   00005140
                      2     COMPLETION CODE                             00005150
                    3-7     UNCHANGED                                   00005160
                                                                        00005170
            COMPLETION CODES:                                           00005180
                                                                        00005190
                    CODE    MEANING                                     00005200
                                                                        00005210
                    FOK     SUCCESSFUL                                  00005220
                    FIS     INVALID STATUS                              00005230
                    FNB     TRANSACTION NOT IN BATCH                    00005240
                    FTN     TRANSACTION NOT ON FILE                     00005250
                    FCC     INVALID CALL CODE                           00005260
                                                                        00005270
17FM        CODE: FCTSS     CHANGE TRANSACTION STATUS                   00005280
                            FROM SCANNERS                               00005290
                                                                        00005300
            INPUT BUFFER LAYOUT:                                        00005310
                                                                        00005320
                    WORD #  CONTENTS                                    00005330
                                                                        00005340
                      1     CALL CODE (FCTSS)                           00005350
                      2     IGNORE                                      00005360
                    3-4     TRANSACTION NUMBER                          00005370
                      5     STATUS (XINS=IN SORTATION,                  00005380
                            XSTO=STOCK-OUT,XXSR=EXCESS                  00005390
                            REPACK,XSTG=STAGED FOR                      00005400
                            SHIPMENT,XMIS=MIS-SORT)                     00005410
                    6-7     TRANSACTION DIRECTORY ENTRY                 00005420
                            FOR STATUS OF IN-SORTATION.                 00005430
                      8     MIS-SORT DEST. FOR STATUS                   00005440
                            OF MIS-SORT                                 00005450
                                                                        00005460
            RETURN BUFFER LAYOUT:                                       00005470
                                                                        00005480
                    WORD #  CONTENTS                                    00005490
                                                                        00005500
                      1     UNCHANGED                                   00005510
                      2     COMPLETION CODE                             00005520
                    3-5     UNCHANGED                                   00005530
                                                                        00005540
            COMPLETION CODES:                                           00005550
                                                                        00005560
                    CODE    MEANING                                     00005570
                                                                        00005580
                    FOK     SUCCESSFUL                                  00005590
                    FIS     INVALID STATUS                              00005600
                    FTN     TRANSACTION NOT ON FILE                     00005610
                    FCC     INVALID CALL CODE                           00005620
                                                                        00005630
```

```
18FM    CODE: FBID                      READ BATCH RECORD VIA           00005640
                                        BATCH I.D.                      00005650
                                                                        00005660
        INPUT BUFFER LAYOUT:                                            00005670
                                                                        00005680
              WORD #                    CONTENTS                        00005690
                                                                        00005700
                1                       CALL CODE (FBID)                00005710
                2                       IGNORE                          00005720
               3-4                      BATCH I.D.                      00005730
                5                       RECORD BUFFER ADDRESS (IN       00005740
                                        $SYSCOM)(BUFFER SIZE > OR =     00005750
                                        64 BYTES)                       00005760
                                                                        00005770
        RETURN BUFFER LAYOUT:                                           00005780
                                                                        00005790
              WORD #                    CONTENTS                        00005800
                                                                        00005810
                1                       UNCHANGED                       00005820
                2                       COMPLETION CODE                 00005830
               3-5                      UNCHANGED                       00005840
                                                                        00005850
        COMPLETION CODES:                                               00005860
                                                                        00005870
              CODE                      MEANING                         00005880
                                                                        00005890
              FOK                       SUCCESSFUL                      00005900
              FBN                       BATCH NOT ON FILE               00005910
              FCC                       INVALID CALL CODE               00005920
                                                                        00005930
19FM    CODE: FTID                      READ TRANSACTION RECORD         00005940
                                        VIA TRANSACTION NO.             00005950
                                                                        00005960
        INPUT BUFFER LAYOUT:                                            00005970
                                                                        00005980
              WORD #                    CONTENTS                        00005990
                                                                        00006000
                1                       CALL CODE (FTID)                00006010
                2                       IGNORE                          00006020
               3-4                      TRANSACTION NUMBER              00006030
                5                       RECORD BUFFER ADDRESS (IN       00006040
                                        $SYSCOM)(BUFFER >  OR = 18      00006050
                                        BYTES)                          00006060
                                                                        00006070
        RETURN BUFFER LAYOUT:                                           00006080
                                                                        00006090
              WORD #                    CONTENTS                        00006100
                                                                        00006110
                1                       UNCHANGED                       00006120
                2                       COMPLETION CODE                 00006130
               3-5                      UNCHANGED                       00006140
                                                                        00006150
        COMPLETION CODES:                                               00006160
                                                                        00006170
              CODE                      MEANING                         00006180
                                                                        00006190
              FOK                       SUCCESSFUL                      00006200
              FTN                       TRANSACTION NOT ON FILE         00006210
              FCC                       INVALID CALL CODE               00006220
                                                                        00006230
20FM    CODE: FRTD                      READ TRANSACTION DIRECTORY      00006240
                                        RECORD VIA TRANSACTION NO.      00006250
                                                                        00006260
        INPUT BUFFER LAYOUT:                                            00006270
                                                                        00006280
              WORD #                    CONTENTS                        00006290
                                                                        00006300
                1                       CALL CODE (FRTD)                00006310
                2                       IGNORE                          00006320
               3-4                      TRANSACTION NUMBER              00006330
                5                       RECORD BUFFER ADDRESS (IN       00006340
                                        $SYSCOM)(BUFFER > OR = 4 BYTES) 00006350
                                                                        00006360
```

```
      RETURN BUFFER LAYOUT:                                          00006370
                                                                     00006380
              WORD #                  CONTENTS                       00006390
                                                                     00006400
                1                     UNCHANGED                      00006410
                2                     COMPLETION CODE                00006420
               3-5                    UNCHANGED                      00006430
                                                                     00006440
      COMPLETION CODES:                                              00006450
                                                                     00006460
              CODE                    MEANING                        00006470
                                                                     00006480
              FOK                     SUCCESSFUL                     00006490
              FTN                     TRANSACTION NOT ON FILE        00006500
              FCC                     INVALID CALL CODE              00006510
                                                                     00006520
21FM  CODE: FRTB                      READ TRANSACTION FILE BLOCK    00006530
                                                                     00006540
      INPUT BUFFER LAYOUT:                                           00006550
                                                                     00006560
              WORD #                  CONTENTS                       00006570
                                                                     00006580
                1                     CALL CODE (FRTB)               00006590
                2                     IGNORE                         00006600
               3-4                    BATCH I.D.                     00006610
                5                     RELATIVE BLOCK NUMBER (1-NO.   00006620
                                      OF TRANS. BLOCKS USED)         00006630
                6                     BLOCK BUFFER ADDRESS (IN       00006640
                                      $SYSCOM)(BUFFER > OR = 256 BYTE00006650
                                                                     00006660
      RETURN BUFFER LAYOUT:                                          00006670
                                                                     00006680
              WORD #                  CONTENTS                       00006690
                                                                     00006700
                1                     UNCHANGED                      00006710
                2                     COMPLETION CODE                00006720
               3-6                    UNCHANGED                      00006730
                                                                     00006740
      COMPLETION CODES:                                              00006750
                                                                     00006760
              CODE                    MEANING                        00006770
                                                                     00006780
              FOK                     SUCCESSFUL                     00006790
              FBN                     BATCH NOT ON FILE              00006800
              FIR                     INVALID RELATIVE RECORD/       00006810
                                      BLOCK NUMBER                   00006820
              FCC                     INVALID CALL CODE              00006830
                                                                     00006840
22FM  CODE: FOID                      READ ORDER RECORD VIA          00006850
                                      ORDER I.D.                     00006860
                                                                     00006870
      INPUT BUFFER LAYOUT:                                           00006880
                                                                     00006890
              WORD #                  CONTENTS                       00006900
                                                                     00006910
                1                     CALL CODE (FOID)               00006920
                2                     IGNORE                         00006930
               3-4                    BATCH I.D.                     00006940
               5-7                    ORDER I.D.                     00006950
                8                     RECORD BUFFER ADDRESS (IN      00006960
                                      $SYSCOM)(BUFFER SIZE > OR =    00006970
                                      18 BYTES)                      00006980
                                                                     00006990
      RETURN BUFFER LAYOUT:                                          00007000
                                                                     00007010
              WORD #                  CONTENTS                       00007020
                                                                     00007030
                1                     UNCHANGED                      00007040
                2                     COMPLETION CODE                00007050
               3-8                    UNCHANGED                      00007060
                                                                     00007070
      COMPLETION CODES:                                              00007080
                                                                     00007090
```

```
                CODE                    MEANING                                 00007100
                                                                                00007110
                FOK                     SUCCESSFUL                              00007120
                FBN                     BATCH NOT ON FILE                       00007130
                FON                     ORDER NOT ON FILE                       00007140
                FCC                     INVALID CALL CODE                       00007150
                                                                                00007160
23FM        CODE: FROL                  READ ORDER RECORD VIA                   00007170
                                        RELATIVE ORDER NO.                      00007180
                                                                                00007190
            INPUT BUFFER LAYOUT:                                                00007200
                                                                                00007210
                WORD #                  CONTENTS                                00007220
                                                                                00007230
                  1                     CALL CODE (FROL)                        00007240
                  2                     IGNORE                                  00007250
                 3-4                    BATCH I.D.                              00007260
                  5                     RELATIVE ORDER NUMBER (1-               00007270
                                        NO. OF ORDERS IN BATCH)                 00007280
                  6                     RECORD BUFFER ADDRESS (IN               00007290
                                        $SYSCOM)(BUFFER SIZE > OR =             00007300
                                        18 BYTES)                               00007310
                                                                                00007320
            RETURN BUFFER LAYOUT:                                               00007330
                                                                                00007340
                WORD #                  CONTENTS                                00007350
                                                                                00007360
                  1                     UNCHANGED                               00007370
                  2                     COMPLETION CODE                         00007380
                 3-6                    UNCHANGED                               00007390
                                                                                00007400
            COMPLETION CODES:                                                   00007410
                                                                                00007420
                CODE                    MEANING                                 00007430
                                                                                00007440
                FOK                     SUCCESSFUL                              00007450
                FBN                     BATCH NOT ON FILE                       00007460
                FIR                     INVALID RELATIVE RECORD/                00007470
                                        BLOCK NUMBER                            00007480
                FCC                     INVALID CALL CODE                       00007490
                                                                                00007500
24FM        CODE: FINT                  COLD START FILE                         00007510
                                        INITIALIZATION                          00007520
                                                                                00007530
            INPUT BUFFER LAYOUT:                                                00007540
                                                                                00007550
                WORD #                  CONTENTS                                00007560
                                                                                00007570
                  1                     CALL CODE (FINT)                        00007580
                  2                     IGNORE                                  00007590
                                                                                00007600
            RETURN BUFFER LAYOUT:                                               00007610
                                                                                00007620
                WORD #                  CONTENTS                                00007630
                                                                                00007640
                  1                     UNCHANGED                               00007650
                  2                     COMPLETION CODE                         00007660
                                                                                00007670
            COMPLETION CODES:                                                   00007680
                                                                                00007690
                CODE                    MEANING                                 00007700
                                                                                00007710
                FOK                     SUCCESSFUL                              00007720
                FCC                     INVALID CALL CODE                       00007730
                                                                                00007740
25FM        FILE MANAGER ENTRY EVENT                                            00007750
                                                                                00007760
            THIS EVENT IS DEFINED IN $SYSCOM AND IS USED                        00007770
            BY ALL CALLERS TO PASS THE ADDRESS OF A                             00007780
            BUFFER CONTAINING INPUT DATA TO THE FILE                            00007790
            MGT. PROGRAM                                                        00007800
                                                                                00007810
```

```
26FM    FILE MANAGER CALL CODE DISPATCH TABLE FORMAT                    00007820
                                                                        00007830
        *  ADRS. OF WORK ROUTINE FOR CALL CODE 1  *                     00007840
        *  ADRS. OF WORK ROUTINE FOR CALL CODE 2  *                     00007850
        *  ADRS. OF WORK ROUTINE FOR CALL CODE 3  *                     00007860
        *  ADRS. OF WORK ROUTINE FOR CALL CODE 4  *                     00007870
        *  ADRS. OF WORK ROUTINE FOR CALL CODE 5  *                     00007880
        *  ADRS. OF WORK ROUTINE FOR CALL CODE N  *                     00007890
                                                                        00007900
27FM    FILE MANAGER COMPLETION EVENT                                   00007910
                                                                        00007920
        THIS EVENT IS DEFINED IN $SYSCOM AND IS USED                    00007930
        TO PASS THE ADDRESS OF A BUFFER CONTAINING                      00007940
        THE FILE MANAGER OUTPUT DATA BACK TO THE                        00007950
        CALLER. THIS BUFFER IS THE SAME ONE USED BY                     00007960
        THE CALLER FOR THE FILE MANAGER ENTRY EVENT.                    00007970
                                                                        00007980
28FM    READ/WRITE FILE BLOCK                                           00007990
                                                                        00008000
        TO READ OR WRITE A SECTOR FROM/TO ONE OF THE                    00008010
        DISK FILES SPECIFIED BELOW, SUBROUTINE                          00008020
        FMRWB IS CALLED WITH THE FOLLOWING INPUT                        00008030
        PARAMETERS:                                                     00008040
                                                                        00008050
                CALL PARAMETERS                                         00008060
                                                                        00008070
                READ/WRITE INDICATOR                                    00008080
                BUFFER ADDRESS (256 BYTE BUFFER) FOR READ/WRITE         00008090
                RELATIVE BLOCK NUMBER                                   00008100
                FILE CODE (SEE BELOW)                                   00008110
                BUFFER ADDRESS SUPPLIED BY FM USER                      00008120
                                                                        00008130
                FILE CODES:                                             00008140
                                                                        00008150
                CODE                    FILE                            00008160
                                                                        00008170
                XDIRF                   TRANSACTION DIRECTORY           00008180
                XACTF                   TRANSACTION FILE                00008190
                BATCHF                  BATCH FILE                      00008200
                ORDERF                  ORDER FILE                      00008210
                XLATEF                  TRANSLATION FILE                00008220
                ORDERI                  BUILD RESIDENT ORDER FILE       00008230
                ORDERU                  UPDATE RESIDENT ORDER FILE      00008240
                                                                        00008250
29FM    ORDER I.D. TABLE                                                00008260
                                                                        00008270
        THE ORDER I.D. TABLE IS A LOCAL TABLE                           00008280
        CONTAINING ORDER I.D.'S AND COUNTS OF                           00008290
        TRANSACTIONS PER ORDER FOR ALL ORDERS                           00008300
        FOUND IN BATCH BEING ADDED TO SYSTEM.                           00008310
                                                                        00008320
31FM    BATCH ADDITION COUNTS                                           00008330
                                                                        00008340
        TWO COUNTERS ARE USED BY ADD TRANSACTION                        00008350
        (FMATR) TO INDICATE THE NO. OF                                  00008360
        TRANSACTIONS AND NO. OF ORDERS ADDED                            00008370
        THUS FAR IN THE BATCH ADDITION PROCESS.                         00008380
                                                                        00008390
32FM    BATCH DELETION QUEUE                                            00008400
                                                                        00008410
        A FIFO QUEUE OF ONE-WORD ENTRIES                                00008420
        CONTAINING RELATIVE BATCH NOS. OF                               00008430
        BATCHES TO BE DELETED BY FMOLD.                                 00008440
                                                                        00008450
33FM    RESIDENT BATCH FILE SEARCH                                      00008460
                                                                        00008470
        TO OBTAIN THE RELATIVE BATCH NO. GIVEN                          00008480
        A BATCH I.D. CALL FMSRB WITH THE BATCH                          00008490
        I.D. AS A PARAMETER. FMSRB SEARCHES                             00008500
        THE RESIDENT BATCH FILE AND RETURNS                             00008510
        EITHER THE RELATIVE BATCH NO. OR A                              00008520
        ZERO VALUE IF THE BATCH IS NOT ON FILE.                         00008530
                                                                        00008540
```

34FM FILE MANAGER RESOURCE DEFINITION 00008550

A QUEUE CONTROL BLOCK DEFINED IN $SYSCOM USED TO FORCE SERIAL USE OF THE FILE MANAGER PROGRAM. 00008560 00008570 00008580 00008590

35FM FILE MANAGER PANIC CALL 00008600

TO PROCESS A PANIC WITHIN THE FILE MANAGER SUB-SYSTEM THE USER SHOULD CALL THE SUBROUTINE FMPAN WITH THE PANIC CODE IN REGISTER 1. (#1) NO RETURN WILL BE MADE FROM THIS CALL. 00008610 00008620 00008630 00008640 00008650 00008660

36FM RECORD BUFFER QUEUE 00008670

THE RECORD BUFFER QUEUE IS DEFINED IN $SYSCOM AND CONTAINS A POOL OF 64 BYTE BUFFERS. THESE BUFFERS ARE USED TO PASS COPIES OF RECORDS TO AND FROM THE FILE MANAGER. 00008680 00008690 00008700 00008710 00008720 00008730

37FM SECTOR BUFFER QUEUE 00008740

THE SECTOR BUFFER QUEUE IS DEFINED IN $SYSCOM AND CONTAINS A POOL OF 256 BYTE BUFFERS. THESE BUFFERS ARE USED TO PASS COPIES OF DISK FILE SECTORS TO AND FROM THE FILE MANAGER. 00008750 00008760 00008770 00008780 00008790 00008800

38FM CODE: FADC ADD BATCH COMPLETE 00008810

00008820

INPUT BUFFER LAYOUT: 00008830

00008840

WORD #	CONTENTS	00008850
		00008860
1	CALL CODE (FADC)	00008870
2	IGNORE	00008880

00008890

RETURN BUFFER LAYOUT: 00008900

00008910

WORD #	CONTENTS	00008920
		00008930
1	UNCHANGED	00008940
2	COMPLETION CODE	00008950

00008960

COMPLETION CODES: 00008970

00008980

CODE	MEANING	00008990
		00009000
FOK	SUCCESSFUL	00009010
FBN	BATCH NOT ON FILE	00009020
FCC	INVALID CALL CODE	00009030

00009040

39FM SYSTEM PARAMETER FILE DEFINITION 00009050

00009060

THE SYSTEM PARAMETER FILE CONSISTS OF 1 SECTOR OF 256 BYTES ON DISK CALLED "PARMFILE" OF WHICH THE FIRST 58 BYTES ARE USED. THE LAYOUT OF THIS FILE FOLLOWS. 00009070 00009080 00009090

00009100

INDEX	FIELD	SIZE(BYTES)	TYPE	00009110
				00009120
SNOS	# OF SHIPPING LINES	2	BINARY	00009130
SREC	RECIRC DESTINATION	2	BINARY	00009140
SERD	ERROR DESTINATION	2	BINARY	00009150
SNOI	# OF INDUCTION LINES	2	BINARY	00009160
SOUT	# OF STOCK-OUT SCANNERS	2	BINARY	00009170
XRPK	# OF EXCESS REPACK SCNRS	2	BINARY	00009180
SHIP	# OF SHIPPING SCANNERS	2	BINARY	00009190
LOGD	# OF LOG DEVICES	2	BINARY	00009200
LOGMR	LOG MESSAGE ROUTING TABLE	42	BINARY	00009210

00009220

FOR FURTHER DEFINITION OF THE LOG MESSAGE ROUTING TABLE ENTRIES, SEE 04AL. 00009230 00009240

00009250

```
       ON IPL, THE DISK-RESIDENT SYSTEM PARAMETER FILE DATA IS          00009260
       COPIED INTO $SYSCOM AS A 29 WORD (58 BYTE) MEMORY RESIDENT       00009270
       TABLE. THE INDEX INTO THE MEMEORY RESIDENT TABLE IS THE          00009280
       SAME AS THE DISK FILE INDEX.                                     00009290
                                                                        00009300
40FM   SYSTEM ERROR COUNTS                                             00009310
                                                                        00009320
       THE SYSTEM ERROR COUNTS RESIDE IN $SYSCOM AND CONTAIN THE        00009330
       NUMBER OF ERRORS FOR EACH CONDITION. THE MIS-SORT COUNT          00009340
       TABLE CONTAINS 1 WORD FOR EACH SORT LINE IN THE SYSTEM.          00009350
       THESE COUNTS CAN BE CLEARED BY OPERATOR COMMAND OR BY            00009360
       AN IPL. THE LAYOUT OF THESE COUNTS FOLLOW:                       00009370
                                                                        00009380
       INDEX        FIELD                  SIZE(BYTES)      TYPE      00009390
                                                                        00009400
       MSRT     MIS-SORT COUNT TABLE          100          BINARY      00009410
       NRED     NO-READ COUNT TABLE             6          BINARY      00009420
       RJCT     REJECT LINE COUNT               2          BINARY      00009430
       PREC     RECIRCS FROM PSC                2          BINARY      00009440
       SCRC     SCANNER ASSIGNED RECIRCS        2          BINARY      00009450
                                                                        00009460


*************************************************************************00000010
*                                                                       00000020
* OPERATOR INTERFACE DATA DICTIONARY INDEX                              00000030
*                                                                       00000040
*************************************************************************00000050
                                                                        00000060
I/O #    DESCRIPTION                                                    00000070
                                                                        00000080
010I     OPERATOR INTERFACE COMMANDS                                    00000090
020I     ERROR MESSAGES                                                 00000100
030I     REQUEST MESSAGES                                               00000110
040I     DESCRIPTIVE MESSAGES                                           00000120
050I     WORK SCHEDULE REPORT LAYOUT                                    00000130
060I     BATCH STATUS REPORT LAYOUT                                     00000140
070I     ORDER STATUS REPORT LAYOUT                                     00000150
080I     TRANSACTION STATUS REPORT LAYOUT                               00000160
090I     SORT LINE ROUTING REPORT LAYOUT                                00000170
100I     LOCAL FLAG DEFINITIONS                                         00000180
110I     GLOBAL FLAG DEFINITIONS                                        00000190
120I     BATCH LOCAL DATA                                               00000200
130I     ORDER LOCAL DATA                                               00000210
140I     TRANSACTION LOCAL DATA                                         00000220
150I     FIND BATCH CALL                                                00000230
160I     GET BATCH CALL                                                 00000240
170I     FIND ORDER CALL                                                00000250
180I     GET ORDER CALL                                                 00000260
190I     FIND TRANSACTION CALL                                          00000270
200I     SEQUENTIAL TRANSACTION CALL                                    00000280
210I     GET TRANSACTION CALL                                           00000290
220I     READ BATCH CALL                                                00000300
230I     WRITE BATCH CALL                                               00000310
240I     ADD BATCH CALL                                                 00000320
250I     ADD TRANSACTION CALL                                           00000330
260I     OPERATOR INTERFACE CONTROL LOCAL DATA                          00000340
270I     WORK SCHEDULE REPORT LOCAL DATA                                00000350
280I     BATCH STATUS REPORT LOCAL DATA                                 00000360
290I     ORDER STATUS REPORT LOCAL DATA                                 00000370
300I     TRANSACTION STATUS REPORT LOCAL DATA                           00000380
310I     SORT TRANSLATION REPORT LOCAL DATA                             00000390
320I     READ BATCH LOCAL DATA                                          00000400
330I     WRITE BATCH LOCAL DATA                                         00000410
340I     START BATCH LOCAL DATA                                         00000420
350I     END BATCH LOCAL DATA                                           00000430
360I     DELETE BATCH LOCAL DATA                                        00000440
370I     REROUTE SHIPPING LINE LOCAL DATA                               00000450
380I     RESTORE SHIPPING LINE LOCAL DATA                               00000460
390I     OPEN PSC COMMUNICATIONS LOCAL DATA                             00000470
400I     ASSIGN TRANSACTION STATUS LOCAL DATA                           00000480
410I     RECIRC LOCAL DATA                                              00000490
420I     HOLD LOCAL DATA                                                00000500
```

```
430I      BATCH VERIFICATION DISPLAY LAYOUT                                00000510
440I      TRANSACTION VERIFICATION DISPLAY LAYOUT                          00000520
450I      CHANGE BATCH STATUS CALL                                         00000530
460I      WRITE SORT TRANSLATION TABLE CALL                                00000540
470I      HELP COMMAND LAYOUT                                              00000550
480I      GET IPM CALL                                                     00000560
490I      GET RECORD BUFFER CALL                                           00000570
500I      FREE IPM CALL                                                    00000580
510I      FREE RECORD BUFFER CALL                                          00000590
520I      SEND ALARM MESSAGE CALL                                          00000600
530I      DELETE BATCH CALL                                                00000610
540I      PANIC CALL                                                       00000620
550I      GET BLOCK BUFFER CALL                                            00000630
560I      FREE BLOCK BUFFER CALL                                           00000640
570I      DISKETTE COMMAND WORD FLAG DEFINITIONS                           00000650
580I      SYSTEM ERROR REPORT LAYOUT                                       00000660
590I      SYSTEM ERROR REPORT LOCAL DATA                                   00000670
600I      MIS-SORT REPORT LOCAL DATA                                       00000680
610I      MIS-SORT REPORT LAYOUT                                           00000690
                                                                           00000700


*************************************************************************00000710
*                                                                        00000720
* OPERATOR INTERFACE MODULE REFERENCES                                   00000730
*                                                                        00000740
*************************************************************************00000750
                                                                         00000760
I/O #     DESCRIPTION                                                    00000770
                                                                         00000780
010I      OICTL                                                          00000790
020I:1    OICTL                                                          00000800
     :2   OICCAB(OIRDB)                                                  00000810
     :3   OICCAT(OIRDB)                                                  00000820
     :4   OICCAB(OIRDB)                                                  00000830
     :5   OICCAT(OIRDB)                                                  00000840
     :6   OIATS                                                         00000850
     :7   OIBSR,OIOSR,OITSR,OIWRB,OISTB,OIENB,OIDLB,OIATS,OIMSR         00000860
     :8   OITSR,OIATS                                                   00000870
     :9   OICCAT(OIRDB)                                                 00000880
     :10  OICCGO(OIOSR),OICCGO(OIMSR)                                   00000890
     :11  OIASL,OIRSL                                                   00000900
     :12  OIBSR,OIOSR,OICCGO(OIOSR),OICCGO(OIMSR),OIMSR                 00000910
     :13  OIATS,OITSR                                                   00000920
     :14  OICCRB(OIRDB),OICCWB(OIWRB)                                   00000930
     :15  OIRDB                                                         00000940
     :16  OIRDB                                                         00000950
     :17  OIDLB                                                         00000960
     :18  OIDLB                                                         00000970
     :19  OIATS                                                         00000980
     :20  INITIAL                                                       00000990
     :21  INITIAL                                                       00001000
     :22  INITIAL                                                       00001010
     :23  OIASL                                                         00001020
     :24  OIRSL                                                         00001030
030I:1    OIWSR,OIBSR,OIOSR,OITSR,OISTR,OISER,OIMSR                      00001040
     :2   OIBSR,OIOSR,OIWRB,OISTB,OIENB,OIDLB,OIATS,OIMSR               00001050
     :3   OIOSR,OIMSR                                                   00001060
     :4   OIOSR,OIMSR                                                   00001070
     :5   OIOSR                                                         00001080
     :6   OIRDB,OIWRB                                                   00001090
     :7   OIRDB                                                         00001100
     :8   OIWRB                                                         00001110
     :9   OISTB                                                         00001120
     :10  OIENB                                                         00001130
     :11  OIDLB                                                         00001140
     :12  OIDLB                                                         00001150
     :13  OITSR,OIATS                                                   00001160
     :14  OIATS                                                         00001170
     :15  OIASL,OIRSL                                                   00001180
     :16  OIASL                                                         00001190
     :17  OIATS                                                         00001200
     :18  OISER                                                         00001210
```

```
    :19  OIRSE                                                          00001220
    :20  OIRSE                                                          00001230
    :21  OIRSE                                                          00001240
    :22  OIRSE                                                          00001250
    :23  OIRSE                                                          00001260
    :24  OIRSE                                                          00001270
040I:1   OICTL                                                          00001280
    :2   OICTL                                                          00001290
    :3   OIWSR                                                          00001300
    :4   OIWSR,OIDLB                                                    00001310
    :5   OIBSR                                                          00001320
    :6   OIOSR                                                          00001330
    :7   OITSR                                                          00001340
    :8   OISTR                                                          00001350
    :9   OIRDB                                                          00001360
    :10  OIWRB                                                          00001370
    :11  OISTB                                                          00001380
    :12  OIENB                                                          00001390
    :13  OIDLB                                                          00001400
    :14  OIATS                                                          00001410
    :15  OIASL                                                          00001420
    :16  OIRSL                                                          00001430
    :17  OIPSC                                                          00001440
    :18  OIPSC                                                          00001450
    :19  OIREC                                                          00001460
    :20  OIHLT                                                          00001470
    :21  OIHLP                                                          00001480
    :22  OIASL,OIRSL                                                    00001490
    :23  OIRDB                                                          00001500
    :24  OIWRB                                                          00001510
    :25  OIRDB,OIWRB                                                    00001520
    :26  OIATS                                                          00001530
    :27  INIT                                                          00001540
    :28  INIT                                                          00001550
    :29  INITIAL                                                       00001560
    :30  INITIAL                                                       00001570
    :31  OISER                                                         00001580
    :32  OISER                                                         00001590
    :33  OIRSE                                                         00001600
    :34  OIRSE                                                         00001610
    :35  OIMSR                                                         00001620
    :36  OIMSR                                                         00001630
    :37  OIOSR                                                         00001640
050I     OIWSR                                                         00001650
060I     OIBSR                                                         00001660
070I     OIOSR                                                         00001670
080I     OITSR                                                         00001680
090I     OISTR                                                         00001690
100I     OICTL,OIWSR,OIBSR,OIOSR,OITSR,OISTR,OIRDB,OIWRB,OISTB,OIENB,  00001700
         OIDLB,OIATS,OIASL,OIRSL,OIPSC,OIMSR                           00001710
110I     OICTL,OIRDB,OIWRB                                             00001720
120I     OIWSR,OIBSR,OIOSR,OITSR,OIRDB,OIWRB,OISTB,OIENB,OIDLB,OIATS,  00001730
         OIMSR                                                         00001740
130I     OIBSR,OIOSR,OIWRB,OIMSR                                       00001750
140I     OIOSR,OITSR,OIRDB,OIWRB,OIATS,OIMSR                           00001760
150I     OIWSR                                                         00001770
160I     OICCFB,OIBSR,OIWRB,OISTB,OIENB,OIDLB,OIATS,OIMSR              00001780
170I     OIBSR,OIOSR,OIWRB,OIDLB,OIMSR                                 00001790
180I     OIOSR,OIMSR                                                   00001800
190I     OIOSR,OIWRB,OIMSR                                             00001810
200I     OICCFT                                                        00001820
210I     OITSR,OIATS                                                   00001830
220I     OIRDB                                                         00001840
230I     OIWRB                                                         00001850
240I     OIRDB                                                         00001860
250I     OIRDB                                                         00001870
260I     OICTL                                                         00001880
270I     OIWSR                                                         00001890
280I     OIBSR                                                         00001900
290I     OIOSR                                                         00001910
300I     OITSR                                                         00001920
310I     OISTR                                                         00001930
320I     OIRDB                                                         00001940
```

```
330I     OIWRB                                                          00001950
340I     OISTB                                                          00001960
350I     OIENB                                                          00001970
360I     OIDLB                                                          00001980
370I     OIASL                                                          00001990
380I     OIRSL                                                          00002000
390I     OIPSC                                                          00002010
400I     OIATS                                                          00002020
410I     OIREC                                                          00002030
420I     OIHLT                                                          00002040
430I     OIRDB,OIWRB,OISTB,OIENB,OIDLB,OIATS                            00002050
440I     OIATS                                                          00002060
450I     OISTB,OIENB                                                    00002070
460I     OIASL,OIRSL                                                    00002080
470I     OIHLP                                                          00002090
480I     OIWSR,OIBSR,OIOSR,OITSR,OIRDB,OISTB,OIENB,OIDLB,OIATS,OIWRB,   00002100
         OIMSR                                                          00002110
490I     OIWSR,OIBSR,OIOSR,OITSR,OIRDB,OISTB,OIENB,OIDLB,OIATS,OIMSR    00002120
500I     OIWSR,OIBSR,OIOSR,OITSR,OIRDB,OISTB,OIENB,OIDLB,OIATS,OIWRB,   00002130
         OIMSR                                                          00002140
510I     OIWSR,OIBSR,OIOSR,OITSR,OIRDB,OISTB,OIENB,OIDLB,OIATS,OIMSR    00002150
520I     OISTB,OIENB                                                    00002160
530I     OIDLB,OIRDB                                                    00002170
540I     OITSR,OIPSC,OIATS,OICCGB,OICCCB,OICCRB,OICCAB,OICCWB,OICCFD,   00002180
         OICCGO,OICCST,OICCGT,OICCAT,OICCWS,OICCSA,OICCDB,FMATR,OIMSR,  00002190
         OIOSR                                                          00002200
550I     OICCST                                                         00002210
560I     OICCST                                                         00002220
570I     OIRDB,OIWRB                                                    00002230
580I     OISER                                                          00002240
590I     OISER                                                          00002250
600I     OIMSR                                                          00002260
610I     OIMSR                                                          00002270
                                                                        00002280

*************************************************************************00002290
*                                                                       00002300
* OPERATOR INTERFACE DATA DICTIONARY ENTRIES                            00002310
*                                                                       00002320
*************************************************************************00002330
                                                                        00002340
I/O #                DESCRIPTION                                        00002350
                                                                        00002360
010I     OPERATOR INTERFACE COMMANDS:                                   00002370
                                                                        00002380
         CATEGORY:        COMMAND:  DESCRIPTION:                        00002390
                                                                        00002400
         STATUS REPORTS   BATCH     BATCH STATUS REPORT                00002410
                          ORDER     ORDER STATUS REPORT                00002420
                          TRANSACT  TRANSACTION STATUS REPORT          00002430
                          SCHEDULE  WORK SCHEDULE REPORT               00002440
                          ROUTE     SORT LINE ROUTING REPORT           00002450
                          MISSORT   MISSORT EXCEPTIONS                 00002460
                                                                       00002470
         BATCH CONTROL    READ      READ BATCH FROM DISKETTE           00002480
                          START     START BATCH PROCESSING             00002490
                          END       END BATCH PROCESSING               00002500
                          WRITE     WRITE BATCH TO DISKETTE            00002510
                          DELETE    DELETE BATCH INFORMATION           00002520
                                                                       00002530
         TRANSACTION      ASSIGN    ASSIGN TRANSACTION STATUS          00002540
                                                                       00002550
         SCANNER INPUT    RECIRC    RECIRCULATE ON SCANNER ERROR       00002560
                          HOLD      HOLD ON SCANNER ERROR              00002570
                                                                       00002580
         SORT LINE        REROUTE   ASSIGN ALTERNATE SORT LINE         00002590
                          RESTORE   RESTORE SORT LINE                  00002600
                                                                       00002610
         PSC CONTROL      OPEN      OPEN PSC COMMUNICATIONS            00002620
                                                                       00002630
         DIAGNOSTIC       HELP      OPERATOR COMMAND LIST              00002640
                          RESET     RESET SYSTEM ERROR COUNTS          00002650
                          ERROR     DISPLAY SYSTEM ERROR COUNTS        00002660
                                                                       00002670
020I     ERROR MESSAGES:                                                00002680
```

```
                                                                00002690
1: PROGRAM LOAD ERROR XX                                        00002700
                                                                00002710
        UNABLE TO LOAD AN OPERATOR COMMAND PROGRAM FOR EXECU-   00002720
        TION. EDX COMPLETION CODE (XX) FOR LOAD STATEMENT       00002730
        DESCRIBES CAUSE OF ERROR.                               00002740
                                                                00002750
2: BATCH FILE SPACE EXHAUSTED                                   00002760
                                                                00002770
        ATTEMPT TO ADD TOO MANY BATCH RECORDS. BATCH NOT        00002780
        ADDED.                                                  00002790
                                                                00002800
3: TRANSACTION FILE SPACE EXHAUSTED                             00002810
                                                                00002820
        ATTEMPT TO ADD TOO MANY TRANSACTIONS RECORDS. BATCH     00002830
        AUTOMATICALLY DELETED.                                  00002840
                                                                00002850
4: BATCH ALREADY ON FILE XXX                                    00002860
                                                                00002870
        ATTEMPT TO ADD BATCH ID (XXX) FAILED BECAUSE A DUP-     00002880
        LICATE BATCH ID WAS ENCOUNTERED.                        00002890
                                                                00002900
5: TRANSACTION ALREADY ON FILE XXXXXX                           00002910
                                                                00002920
        ATTEMPT TO ADD TRANSACTION ID (XXXXXX) FAILED BECAUSE   00002930
        A DUPLICATE TRANSACTIN ID WAS ENCOUNTERED.              00002940
                                                                00002950
6: INVALID STATUS X                                             00002960
                                                                00002970
        ATTEMPT TO CHANGE EITHER BATCH OR TRANSACTION STATUS    00002980
        FAILED BECAUSE OF INVALID STATUS CODE. OLD STATUS NOT   00002990
        MODIFIED.                                               00003000
                                                                00003010
7: BATCH NOT ON FILE XXX                                        00003020
                                                                00003030
        ATTEMPT TO ACCESS BATCH ID (XXX) INFORMATION NOT CUR-   00003040
        RENTLY AVAILABLE.                                       00003050
                                                                00003060
8: TRANSACTION NOT ON FILE XXXXXX                               00003070
                                                                00003080
        ATTEMPT TO ACCESS TRANSACTION ID (XXXXXX) INFORMATION   00003090
        NOT CURRENTLY AVAILABLE.                                00003100
                                                                00003110
9: ORDER FILE SPACE EXHAUSTED                                   00003120
                                                                00003130
        ATTEMPT TO ADD TOO MANY ORDER RECORDS. BATCH AUTOMA-    00003140
        TICALLY DELETED.                                        00003150
                                                                00003160
10: ORDER NOT ON FILE XXXXXX                                    00003170
                                                                00003180
        ATTEMPT TO ACCESS ORDER ID (XXXXXX) INFORMATION NOT     00003190
        CURRENTLY AVAILABLE.                                    00003200
                                                                00003210
11: INVALID SORT LINE NUMBER XX                                 00003220
                                                                00003230
        ATTEMPT TO ASSIGN OR RESTORE A SORT LINE OUTSIDE        00003240
        KNOWN RANGE.                                            00003250
                                                                00003260
12: BATCH NO LONGER ON FILE - REPORT TERMINATED                 00003270
                                                                00003280
        BATCH INFORMATION WAS DELETED DURING REPORT GENER-      00003290
        ATION BUT AFTER BATCH ID WAS VERIFIED.                  00003300
                                                                00003310
13: INVALID TRANSACTION NUMBER                                  00003320
                                                                00003330
        TRANSACTION NUMBER OUT OF RANGE.                        00003340
                                                                00003350
14: DISKETTE I/O ERROR OCCURRED                                 00003360
                                                                00003370
        ATTEMPT TO READ OR WRITE A DISKETTE DATA BLOCK          00003380
        FAILED.                                                 00003390
                                                                00003400
15: TRANSACTION VALIDATION ERROR                                00003410
                                                                00003420
```

```
      ATTEMPT TO ADD A TRANSACTION RECORD FAILED BECAUSE OF A      00003430
      DATA VALIDATION CHECK REJECTED EITHER; TRANSACTION ID,       00003440
      DESTINATION, OR STATUS CODE.                                 00003450
                                                                   00003460
   16: DISKETTE FORMAT ERROR                                       00003470
                                                                   00003480
      ATTEMPT TO ACCESS A NON-STANDARD IBM SYSTEM 3 DISKETTE.      00003490
                                                                   00003500
   17: BATCH MUST BE PENDING OR COMPLETE TO BE DELETED - COMMAND   00003510
      ABORTED                                                      00003520
                                                                   00003530
      ATTEMPTED TO DELETE AN ACTIVE BATCH. THE COMMAND IS ABORTED  00003540
                                                                   00003550
   18: ALL CURRENT BATCHES ARE ACTIVE - COMMAND ABORTED            00003560
                                                                   00003570
      ATTEMPTED TO DELETE ALL BATCHES WHEN ALL BATCHES ON FILE     00003580
      WERE ACTIVE. THE COMMAND IS ABORTED.                         00003590
                                                                   00003600
   19: TRANSACTION XXXXXX NOT IN BATCH XXX                         00003610
                                                                   00003620
      ATTEMPT TO CHANGE A TRANSACTION STATUS THAT IS NOT           00003630
      IN THE REQUESTED BATCH.                                      00003640
                                                                   00003650
   20: ERROR IN BATCH XXX                                          00003660
                                                                   00003670
      ERROR WAS FOUND IN THE TRANSACTION FILE AT SYSTEM            00003680
      START UP TIME. STATUS OF BATCH IS QUESTIONABLE.              00003690
                                                                   00003700
   21: DISK ERROR WHILE PROCESSING BATCH XXX                       00003710
                                                                   00003720
      ATTEMPT TO READ A PORTION OF THE TRANSACTION FILE            00003730
      AT SYSTEM START UP TIME FAILED. STATUS OF BATCH              00003740
      IS QUESTIONABLE.                                             00003750
                                                                   00003760
   22: ERROR WHILE READING BATCH FILE                              00003770
                                                                   00003780
      ATTEMPT TO READ THE BATCH FILE AT SYSTEM START UP            00003790
      TIME FAILED. STATUS OF SYSTEM IS QUESTIONABLE.               00003800
                                                                   00003810
   23: ATTEMPT TO ASSIGN A SORT LINE TO ITSELF                     00003820
                                                                   00003830
      AN ATTEMPT WAS MADE TO REROUTE A SORT LINE TO IT-            00003840
      SELF USING THE REROUTE COMMAND. NO ACTION IS TAKEN           00003850
      AND THE COMMAND IS ABORTED.                                  00003860
                                                                   00003870
   24: NO PRIOR ASSIGNMENT OF SORT LINE                            00003880
                                                                   00003890
      AN ATTEMPT WAS MADE TO RESTORE A SORT LINE THAT              00003900
      HAD NOT BEEN REROUTED USING THE RESTORE COMMAND.             00003910
      NO ACTION IS TAKEN AND THE COMMAND IS ABORTED.               00003920
                                                                   00003930
030I  REQUEST MESSAGES:                                            00003940
                                                                   00003950
    1: DO YOU WANT HARD COPY?                                      00003960
                                                                   00003970
         Y          USER TERMINAL DEQUEUED AND SYSTEM PRINTER     00003980
                    ENQUEUED. ALL FUTURE MESSAGES LISTED ON       00003990
                    SYSTEM PRINTER.                               00004000
                                                                  00004010
         N          NO ACTION.                                    00004020
                                                                  00004030
    2: ENTER BATCH ID:                                            00004040
                                                                  00004050
         XXX        3-CHARACTER ALPHANUMERIC BATCH ID.            00004060
                                                                  00004070
    3: ALL ORDERS ?                                               00004080
                                                                  00004090
         Y          ALL ORDER INFORMATION FOR A PREVIOUSLY RE-    00004100
                    QUESTED BATCH ID, WILL BE ACCESSED.           00004110
                                                                  00004120
         N          NO ACTION.                                    00004130
                                                                  00004140
    4: ENTER ORDER ID:                                            00004150
                                                                  00004160
```

```
        XXXXXX    6-CHARACTER ALPHANUMERIC ORDER ID.                  00004170
                                                                      00004180
5: LIST EXCEPTIONS ?                                                  00004190
                                                                      00004200
        Y         ALL TRANSACTION INFORMATION NOT "STAGED FOR          00004210
                  SHIPMENT" FOR A PREVIOUSLY REQUESTED BATCH          00004220
                  ID AND ORDER ID, WILL BE LISTED.                    00004230
                                                                      00004240
        N         NO ACTION.                                          00004250
                                                                      00004260
6: DISKETTE MOUNTED ?                                                 00004270
                                                                      00004280
        Y         ACCESS OF DISKETTE ATTEMPTED.                       00004290
                                                                      00004300
        N         NO ACTION.                                          00004310
                                                                      00004320
7: OK TO READ ?                                                       00004330
                                                                      00004340
        Y         DISKETTE BATCH INFORMATION READ.                    00004350
                                                                      00004360
        N         READ BATCH (FROM DISKETTE) COMMAND TERM-            00004370
                  INATED.                                             00004380
                                                                      00004390
8: CORRECT BATCH ?                                                    00004400
                                                                      00004410
        Y         BATCH INFORMATION FOR A PREVIOUSLY REQUEST-         00004420
                  ED BATCH ID IS WRITTEN TO DISKETTE.                 00004430
                                                                      00004440
        N         WRITE BATCH (TO DISKETTE) COMMAND TERMIN-           00004450
                  ATED.                                               00004460
                                                                      00004470
9: START BATCH ?                                                      00004480
                                                                      00004490
        Y         BATCH STATUS FOR PREVIOUSLY REQUESTED BATCH         00004500
                  ID IS CHANGED.                                      00004510
                                                                      00004520
        N         BATCH STATUS NOT CHANGED.                           00004530
                                                                      00004540
10: END BATCH ?                                                       00004550
                                                                      00004560
        Y         BATCH STATUS FOR PREVIOUSLY REQUESTED BATCH         00004570
                  ID IS CHANGED.                                      00004580
                                                                      00004590
        N         BATCH STATUS NOT CHANGED.                           00004600
                                                                      00004610
11: ALL BATCHES ?                                                     00004620
                                                                      00004630
        Y         ALL BATCH INFORMATION CURRENTLY AVAILABLE           00004640
                  MAY BE DELETED.                                     00004650
                                                                      00004660
        N         NO ACTION.                                          00004670
                                                                      00004680
12: DELETE BATCH ?                                                    00004690
                                                                      00004700
        Y         ALL BACH INFORMATION ASSOCIATED WITH THE            00004710
                  PREVIOUSLY DISPLAYED BATCH ID WILL BE DES-          00004720
                  TROYED.                                             00004730
                                                                      00004740
        N         NO ACTION.                                          00004750
                                                                      00004760
13: ENTER TRANSACTION ID:                                             00004770
                                                                      00004780
        XXXXXX    6-DIGIT NUMERIC TRANSACTION ID, IN THE              00004790
                  RANGE 1 THROUGH 999999.                             00004800
                                                                      00004810
14: ENTER NEW STATUS NUMBER:                                          00004820
                                                                      00004830
        X         SINGLE DIGIT NUMERIC TRANSACTION STATUS, IN         00004840
                  THE RANGE 1 THROUGH 5.                              00004850
                                                                      00004860
15: ENTER OLD SORT LINE:                                              00004870
                                                                      00004880
        XX        2-DIGIT NUMERIC SORT LINE NUMBER, IN THE            00004890
                  RANGE 1 THROUGH 99.                                 00004900
                                                                      00004910
```

```
16: ENTER NEW SORT LINE:                                          00004920
                                                                  00004930
          XX          2-DIGIT NUMERIC SORT LINE NUMBER, IN THE    00004940
                      RANGE 1 THROUGH 99.                         00004950
                                                                  00004960
17: TYPE "CANCEL" TO TERMINATE:                                   00004970
                                                                  00004980
          CANCEL      COMMAND ABORTED                             00004990
                                                                  00005000
          <ENTER>     CONTINUE PROCESSING                         00005010
                                                                  00005020
18: LIST MISSORT COUNTS ?                                         00005030
                                                                  00005040
          Y           MISSORTS COUNTS LISTED                      00005050
                                                                  00005060
          N           NO ACTION                                   00005070
                                                                  00005080
19: CLEAR ALL COUNTS ?                                            00005090
                                                                  00005100
          Y           ALL ERROR COUNTS CLEARED                    00005110
                                                                  00005120
          N           NO ACTION                                   00005130
                                                                  00005140
20: CLEAR MISSORT COUNTS ?                                        00005150
                                                                  00005160
          Y           MISSORT COUNTS CLEARED                      00005170
                                                                  00005180
          N           NO ACTION                                   00005190
                                                                  00005200
21: CLEAR SCANNER NO-READ COUNTS ?                                00005210
                                                                  00005220
          Y           NO-READ COUNTS CLEARED                      00005230
                                                                  00005240
          N           NO ACTION                                   00005250
                                                                  00005260
22: CLEAR REJECT COUNT ?                                          00005270
                                                                  00005280
          Y           REJECT COUNT CLEARED                        00005290
                                                                  00005300
          N           NO ACTION                                   00005310
                                                                  00005320
23: CLEAR RECIRCULATION COUNTS ?                                  00005330
                                                                  00005340
          Y           RECIRCULATION COUNTS CLEARED                00005350
                                                                  00005360
          N           NO ACTION                                   00005370
                                                                  00005380
24: CLEAR LOST FROM TRACKING COUNT ?                              00005390
                                                                  00005400
          Y           LFT COUNT CLEARED                           00005410
                                                                  00005420
          N           NO ACTION                                   00005430
                                                                  00005440
043I      DESCRIPTIVE MESSAGES:                                   00005450
                                                                  00005460
 1: DISKETTE ALREADY IN USE                                       00005470
 2: RETRY COMMAND LATER                                           00005480
 3: WORK SCHEDULE REPORT                                          00005490
 4: NO CURRENT BATCH INFORMATION                                  00005500
 5: BATCH STATUS REPORT                                           00005510
 6: ORDER STATUS REPORT                                           00005520
 7: TRANSACTION STATUS REPORT                                     00005530
 8: SORT TRANSLATION REPORT                                       00005540
 9: READ BATCH COMMAND                                            00005550
10: WRITE BATCH COMMAND                                           00005560
11: START BATCH COMMAND                                           00005570
12: END BATCH COMMAND                                             00005580
13: DELETE BATCH COMMAND                                          00005590
14: ASSIGN TRANSACTION STATUS COMMAND                             00005600
15: ASSIGN SHIPPING LINE COMMAND                                  00005610
16: RESTORE SHIPPING LINE COMMAND                                 00005620
17: OPEN PSC COMMUNICATIONS COMMAND                               00005630
18: PSC COMMUNICATIONS INITIATED                                  00005640
19: RECIRC ON SCANNER ERROR                                       00005650
```

```
20: HOLD ON SCANNER ERROR                                                00005660
21: HELP COMMAND                                                         00005670
22: SHIPPING LINE ASSIGNED                                               00005680
23: READ BATCH COMMAND ABORTED                                           00005690
24: WRITE BATCH COMMAND ABORTED                                          00005700
25: REMOVE DISKETTE                                                      00005710
26: STATUS CHANGE COMPLETE                                               00005720
27: SYSTEM STARTED                                                       00005730
28: SET DATE AND TIME USING COMMAND $T                                   00005740
29: SYSTEM INITIALIZATION IN PROGRESS                                    00005750
30: STATUS OF SYSTEM IS QUESTIONABLE                                     00005760
31: SYSTEM ERROR REPORT                                                  00005770
32: M I S - S O R T S                                                    00005780
33: RESET ERROR COUNTS COMMAND                                           00005790
34: COUNTS CLEARED                                                       00005800
35: MIS-SORT REPORT                                                      00005810
36: NO MIS-SORTS IN THIS ORDER                                           00005820
37: NO EXCEPTIONS IN THIS ORDER                                          00005830
                                                                         00005840
050I    WORK SCHEDULE REPORT LAYOUT:                                     00005850
                                                                         00005860
        REPORT FIELD:                  FROM:                             00005870
                                                                         00005880
 1:  DATE                              270I TIME                         00005890
 2:  TIME                              270I TIME                         00005900
 3:  PAGE                              270I PAGECNT                      00005910
 4:  BATCH ID                          120I BID                          00005920
 5:  BATCH STATUS                      120I BSTATUS, BSTBL               00005930
 6:  ORDER COUNT                       120I BOCNT                        00005940
 7:  TRANSACTION COUNT                 120I BTCNT                        00005950
 8:  HEADER INFORMATION                120I BHDR                         00005960
                                                                         00005970
*  SEE REPORT LAYOUT SHEET                                               00005980
                                                                         00005990
060I    BATCH STATUS REPORT LAYOUT:                                      00006000
                                                                         00006010
        REPORT FIELD:                  FROM:                             00006020
                                                                         00006030
 1:  DATE                              280I TIME                         00006040
 2:  TIME                              280I TIME                         00006050
 3:  PAGE                              280I PAGECNT                      00006060
 4:  BATCH ID                          120I BID                          00006070
 5:  BATCH STATUS                      120I BSTATUS, BSTBL               00006080
 6:  ORDER COUNT                       120I BOCNT                        00006090
 7:  TRANSACTION COUNT                 120I BTCNT                        00006100
 8:  HEADER INFORMATION                120I BHDR                         00006110
 9:  BATCH % COMPLETE                  120I BPCOM                        00006120
                                                                         00006130
*  SEE REPORT LAYOUT SHEET                                               00006140
                                                                         00006150
070I    ORDER STATUS REPORT LAYOUT:                                      00006160
                                                                         00006170
        REPORT FIELD:                  FROM:                             00006180
                                                                         00006190
 1:  DATE                              290I TIME                         00006200
 2:  TIME                              290I TIME                         00006210
 3:  PAGE                              290I PAGECNT                      00006220
 4:  BATCH ID                          120I BID                          00006230
 5:  BATCH STATUS                      120I BSTATUS, BSTBL               00006240
 6:  ORDER COUNT                       120I BOCNT                        00006250
 7:  TRANSACTION COUNT                 290I RTCNT                        00006260
 8:  HEADER INFORMATION                120I BHDR                         00006270
 9:  ORDER NUMBER                      130I OCNT                         00006280
10:  ORDER ID                          130I OID                          00006290
11:  QUANTITY PLANNED                  130I OTCNT                        00006300
12:  QUANTITY STAGED                   130I OQSTG                        00006310
13:  QUANTITY NOT SHIPPED              130I OQNSTG                       00006320
14:  QUANTITY STOCK-OUT                130I OQSTO                        00006330
15:  QUANTITY EXCESS REPACK            130I OQXRP                        00006340
16:  QUANTITY IN SORTATION             130I OQISORT                      00006350
17:  QUANTITY NOT PICKED               130I OQNPKD                       00006360
18:  QUANTITY MIS-SORT                 130I OQMIS                        00006370
19:  ORDER % COMPLETE                  130I OPCOM                        00006380
20:  TRANSACTION NUMBER                290I RTCNT                        00006390
```

```
        21:  TRANSACTION ID                  140I TID                          00006400
        22:  TRANSACTION STATUS              140I TSTATUS                      00006410
        23:  PRODUCT ID                      140I TPROD                        00006420
        24:  SOURCE LOCATION                 140I TSOURCE                      00006430
        25:  DESTINATION                     140I TDEST                        00006440
        26:  MIS-SORT DESTINATION                                              00006450
                                                                               00006460
        * SEE REPORT LAYOUT SHEET                                              00006470
                                                                               00006480
080I    TRANSACTION STATUS REPORT LAYOUT:                                      00006490
                                                                               00006500
             REPORT FIELD:                   FROM:                             00006510
                                                                               00006520
         1:  DATE                            300I TIME                         00006530
         2:  TIME                            300I TIME                         00006540
         3:  PAGE                            300I PAGECNT                      00006550
         4:  BATCH ID                        120I BID                          00006560
         5:  BATCH STATUS                    120I BSTATUS, BSTBL               00006570
         6:  ORDER COUNT                     120I BOCNT                        00006580
         7:  HEADER INFORMATION              120I BHDR                         00006590
         8:  TRANSACTION ID                  140I TID                          00006600
         9:  TRANSACTION STATUS              140I TSTATUS                      00006610
        10:  PRODUCT ID                      140I TPROD                        00006620
        11:  SOURCE LOCATION                 140I TSOURCE                      00006630
        12:  DESTINATION                     140I TDEST                        00006640
        13:  MIS-SORT DESTINATION                TMDEST                        00006650
                                                                               00006660
        * SEE REPORT LAYOUT SHEET                                              00006670
                                                                               00006680
090I    SORT LINE ROUTING REPORT LAYOUT:                                       00006690
                                                                               00006700
             REPORT FIELD:                   FROM:                             00006710
                                                                               00006720
         1:  DATE                            310I TIME                         00006730
         2:  TIME                            310I TIME                         00006740
         3:  PAGE                            310I PAGECNT                      00006750
         4:  SORT LINE                       06FM                              00006760
         5:  ALTERNATE                       06FM                              00006770
                                                                               00006780
        * SEE REPORT LAYOUT SHEET                                              00006790
                                                                               00006800
100I    LOCAL FLAG DEFINITIONS:                                                00006810
                                                                               00006820
             LABEL:    DESCRIPTION:              VALUE: BYTES: TYPE:           00006830
                                                                               00006840
         1:  FLGDEF    LOCAL FLAG DEFINITION            2    BINARY           00006850
         2:  FLGOK     (GENERAL PURPOSE "OK")    0001   0    EQUATE           00006860
         3:  FLGHCY    HARDCOPY DEVICE           0002   0    EQUATE           00006870
         4:  FLGBFD    BATCH FOUND               0004   0    EQUATE           00006880
         5:  FLBSPOB   SPECIFIC ORDER OR BATCH   0008   0    EQUATE           00006890
         6:  FLGOSQ    ORDER SEQUENCE PROCESSING 0010  0    EQUATE           00006900
         7:  FLGEXC    EXCEPTION PROCESSING      0020   0    EQUATE           00006910
         8:  FLGDEL    DELETION PROCESSING       0040   0    EQUATE           00006920
         9:  FLGLOR    LIST ORDER RECORD         0080   0    EQUATE           00006930
        10:  FLGLTR    LIST TRANSACTION RECORD   0100   0    EQUATE           00006940
        11:  FLGABO    BATCH READ/WRITE ABORTED  0200   0    EQUATE           00006950
        12:  FLGLR     LAST RECORD PROCESSING    0400   0    EQUATE           00006960
        13:  FLGRAB    REPORT ABORTED            0800   0    EQUATE           00006970
                                                                               00006980
110I    GLOBAL FLAG DEFINITIONS:                                               00006990
                                                                               00007000
             LABEL:    DESCRIPTION:              VALUE: BYTES: TYPE:           00007010
                                                                               00007020
         1:  OIFLG     GLOBAL FLAG DEFINITION           2    BINARY           00007030
         2:  OIDIU     DISKETTE IN USE           0001   0    EQUATE           00007040
                                                                               00007050
             MODULE REFERENCE: OICTL, OIROB, OIWRB.                            00007060
                                                                               00007070
120I    BATCH LOCAL DATA:                                                      00007080
                                                                               00007090
             LABEL:    DESCRIPTION:                   BYTES: TYPE:           00007100
                                                                               00007110
         1:  BRDA      RESIDENT DIRECTORY ADDRESS       2    BINARY           00007120
         2:  BMAX      MAXIMUM BATCHES ALLOWED          2    BINARY           00007130
```

```
 3:  BCNT      BATCH COUNTER                          2    BINARY   00007140
 4:  BID       BATCH ID                               6    EBCDIC   00007150
 5:  BSTATUS   BATCH STATUS                           2    BINARY   00007160
 6:  BHDR      BATCH HEADER                          52    EBCDIC   00007170
 7:  BOCNT     BATCH ORDER COUNT                      2    BINARY   00007180
 8:  BTCNT     BATCH TRANSACTION COUNT                2    BINARY   00007190
 9:  BRDES     RESIDENT DIRECTORY ENTRY SIZE          2    BINARY   00007200
10:  BIPM      IPM BUFFER ADDRESS                     2    BINARY   00007210
11:  BSRB      SYSCOM RECORD BUFFER ADDRESS           2    BINARY   00007220
12:  BPANIC1   PANIC CODE 1                           2    BINARY   00007230
13:  BPANIC2   PANIC CODE 2                           2    BINARY   00007240
14:  BPANIC3   PANIC CODE 3                           2    BINARY   00007250
15:  BPCOM     BATCH PERCENT COMPLETE                 8    BINARY   00007260
16:  BESTBL    BATCH ENGLISH STATUS TABLE            48    EBCDIC   00007270
17:  BHDR1     BATCH REPORT HEADER                   82    EBCDIC   00007280
18:  BHDR2     BATCH REPORT HEADER                   30    EBCDIC   00007290
19:  BHDR3     BATCH REPORT HEADER                   50    EBCDIC   00007300
20:  BDTL      BATCH REPORT DETAIL INFORMATION       32    EBCDIC   00007310
21:  BFTN      BATCH REPORT FOOTNOTE                 34    EBCDIC   00007320
                                                                    00007330
130I  ORDER LOCAL DATA:                                             00007340
                                                                    00007350
      LABEL:   DESCRIPTION                          BYTES: TYPE:    00007360
                                                                    00007370
 1:   OID      ORDER ID                               8    EBCDIC   00007380
 2:   OCNT     ORDER COUNTER                          2    BINARY   00007390
 3:   ORNO     ORDER RELATIVE NUMBER                  2    BINARY   00007400
 4:   OTCNT    ORDER TRANSACTION COUNT                2    BINARY   00007410
 5:   OQSTG    ORDER QUANTITY STAGED                  2    BINARY   00007420
 6:   OQNSTG   ORDER QUANTITY NOT STAGED              2    BINARY   00007430
 7:   OQNPKD   ORDER QUANTITY NOT PICKED              2    BINARY   00007440
 8:   OQISORT  ORDER QUANTITY IN SORTATION            2    BINARY   00007450
 9:   OQXRP    ORDER QUANTITY EXCESS REPACK           2    BINARY   00007460
10:   OQSTO    ORDER QUANTITY STOCK OUT               2    BINARY   00007470
11:   OQMIS    ORDER QUANTITY MIS-SORT                2    BINARY   00007480
12:   OPCOM    ORDER QUANTITY PERCENT COMPLETE        8    BINARY   00007490
13:   OIPM     IPM BUFFER ADDRESS                     2    BINARY   00007500
14:   OSRB     SYSCOM RECORD BUFFER ADDRESS           2    BINARY   00007510
15:   OPANIC1  PANIC CODE 1                           2    BINARY   00007520
16:   OPANIC2  PANIC CODE 2                           2    BINARY   00007530
17:   OPANIC3  PANIC CODE 3                           2    BINARY   00007540
18:   OHDR1    ORDER REPORT HEADER                   82    EBCDIC   00007550
19:   OHDR2    ORDER REPORT HEADER                   82    EBCDIC   00007560
20:   ODTL     ORDER REPORT DETAIL INFORMATION       80    EBCDIC   00007570
                                                                    00007580
140I  TRANSACTION LOCAL DATA:                                       00007590
                                                                    00007600
      LABEL:   DESCRIPTION:                         BYTES: TYPE:    00007610
                                                                    00007620
 1:   TCNT     TRANSACTION COUNTER                    2    BINARY   00007630
 2:   TID      TRANSACTION ID                         8    BINARY   00007640
 3:   TSTATUS  TRANSACTION STATUS                     2    BINARY   00007650
 4:   TRON     TRANSACTION RELATIVE ORDER NUMBER      2    BINARY   00007660
 5:   TPROD    TRANSACTION PRODUCT ID                 6    EBCDIC   00007670
 6:   TSOURCE  TRANSACTION SOURCE                     6    EBCDIC   00007680
 7:   TDEST    TRANSACTION DESTINATION                2    BINARY   00007690
 8:   TFBLK    TRANSACTION FILE BLOCK NUMBER          2    BINARY   00007700
 9:   TFOFF    TRANSACTION FILE ENTRY OFFSET          2    BINARY   00007710
10:   TFRS     FILE RECORD SIZE                       2    BINARY   00007720
11:   TFBF     TRANSACTION FILE BLOCKING FACTOR       2    BINARY   00007730
12:   TIPM     IPM BUFFER ADDRESS                     2    BINARY   00007740
13:   TSRB     SYSCOM RECORD BUFFER ADDRESS           2    BINARY   00007750
14:   TPANIC1  PANIC CODE 1                           2    BINARY   00007760
15:   TPANIC2  PANIC CODE 2                           2    BINARY   00007770
16:   TPANIC3  PANIC CODE 3                           2    BINARY   00007780
17:   TESTBL   TRANSACTION ENGLISH STATUS TABLE      65    EBCDIC   00007790
18:   THDR1    TRANSACTION REPORT HEADER             64    EBCDIC   00007800
19:   THDR2    TRANSACTION REPORT HEADER             78    EBCDIC   00007810
20:   TDTL     TRANSACTION DETAIL INFORMATION        74    EBCDIC   00007820
                                                                    00007830
150I  FIND BATCH CALL: DICCFB                                       00007840
                                                                    00007850
      LABEL:   DESCRIPTION:                         BYTES: TYPE:    00007860
                                                                    00007870
```

```
INPUT:                                                                  00007880
                                                                        00007890
 1:  BRDA      RESIDENT DIRECTORY ADDRESS          2    BINARY        00007900
 2:  BRDES     RESIDENT DIRECTORY ENTRY SIZE       2    BINARY        00007910
 3:  FLGDEF    LOCA FLAG WORD                      2    BINARY        00007920
 4:  BMAX      MAXIMUM BATCHES ALLOWED             2    BINARY        00007930
 5:  BCNT      BATCH COUNTER TO START TABLE        2    BINARY        00007940
               SEARCH                                                   00007950
                                                                        00007960
LOCAL:                                                                  00007970
                                                                        00007980
 1:  SAVE.1    TEMPORARY SAVE #1                   2    BINARY        00007990
                                                                        00008000
OUTPUT:                                                                 00008010
                                                                        00008020
 1:  TFLG      TEMPORARY FLAG                      2    BINARY        00008030
 2:  SAVE.2    TEMPORARY SAVE #2                   2    BINARY        00008040
                                                                        00008050
OUTPUT:                                                                 00008060
                                                                        00008070
 1:  BID       BATCH ID                            3    EBCDIC        00008080
 2:  FLGDEF    UPDATED LOCAL FLAG WORD             2    BINARY        00008090
 3:  BCNT      UPDATED BATCH COUNTER               2    BINARY        00008100
                                                                        00008110
163I  GET BATCH CALL: DICCGB                                            00008120
                                                                        00008130
     LABEL:    DESCRIPTION:                  BYTES: TYPE:             00008140
                                                                        00008150
INPUT:                                                                  00008160
                                                                        00008170
 1:  BID       BATCH ID                            3    EBCDIC        00008180
 2:  BIPM      IPM BUFFER ADDRESS                  2    BINARY        00008190
 3:  BSRB      SYSCOM RECORD BUFFER ADDRESS        2    BINARY        00008200
 4:  BPANIC1   BAD CALL CODE PANIC CODE            2    BINARY        00008210
 5:  BPANIC2   BAD RETURN CODE PANIC CODE          2    BINARY        00008220
 6:  FLGDEF    LOCAL FLAG WORD                     2    BINARY        00008230
                                                                        00008240
LOCAL:                                                                  00008250
                                                                        00008260
 1:  SAVE.1    TEMPORARY SAVE #1                   2    BINARY        00008270
                                                                        00008280
OUTPUT:                                                                 00008290
                                                                        00008300
 1:  BSTATUS   BATCH STATUS                        2    BINARY        00008310
 2:  BHDR      BATCH HEADER                       50    EBCDIC        00008320
 3:  BOCNT     BATCH ORDER COUNTER                 2    BINARY        00008330
 4:  BTCNT     BATCH TRANSACTION COUNTER           2    BINARY        00008340
 5:  PANIC     UPDATED PANIC WORK AREA             2    BINARY        00008350
 6:  FLGDEF    UPDATED LOCAL FLAG WORD             2    BINARY        00008360
                                                                        00008370
170I  FIND ORDER CALL: DICCFO                                           00008380
                                                                        00008390
     LABEL:    DESCRIPTION                   BYTES: TYPE:             00008400
                                                                        00008410
 1:  OIPM      IPM BUFFER ADDRESS                  2    BINARY        00008420
 2:  BID       BATCH ID                            3    EBCDIC        00008430
 3:  OCNT      ORDER NUMBER                        2    BINARY        00008440
 4:  OSRB      SYSCOM RECORD BUFFER ADDRESS        2    BINARY        00008450
 5:  FLGDEF    LOCAL FLAG WORD                     2    BINARY        00008460
 6:  OPANIC1   BAD CALL CODE PANIC CODE            2    BINARY        00008470
 7:  OPANIC2   BAD RELATIVE RECORD PANIC CODE      2    BINARY        00008480
                                                                        00008490
LOCAL:                                                                  00008500
                                                                        00008510
 1:  SAVE.1    TEMPORARY SAVE #1                   2    BINARY        00008520
                                                                        00008530
OUTPUT:                                                                 00008540
                                                                        00008550
 1:  OID       ORDER ID                            6    EBCDIC        00008560
 2:  ORNO      ORDER RELATIVE NUMBER               2    BINARY        00008570
 3:  OQSTB     ORDER QUANTITY STAGED               2    BINARY        00008580
 4:  OQNPKD    ORDER QUANTITY NOT PICKED           2    BINARY        00008590
 5:  OQISRT    ORDER QUANTITY IN SORTATION         2    BINARY        00008600
 6:  OQXRP     ORDER QUANTITY EXCESS REPACK        2    BINARY        00008610
```

```
 7:  OQSTO     ORDER QUANTITY STOCK OUT          2    BINARY  00008620
 8:  OQMIS     ORDER QUANTITY MIS-SORT           2    BINARY  00008630
 9:  FLGDEF    UPDATED LOCAL FLAG WORD           2    BINARY  00008640
10:  OCNT      UPDATED ORDER NUMBER              2    BINARY  00008650
11:  PANIC     UPDATED PANIC WORK AREA           2    BINARY  00008660
                                                              00008670
180I  GET ORDER CALL: OICCGO                                  00008680
                                                              00008690
      LABEL:   DESCRIPTION:                    BYTES: TYPE:   00008700
                                                              00008710
      INPUT:                                                  00008720
                                                              00008730
 1:   OIPM     IPM BUFFER ADDRESS                2    BINARY  00008740
 2:   BID      BATCH ID                          3    EBCDIC  00008750
 3:   OID      ORDER ID                          6    EBCDIC  00008760
 4:   OSRB     SYSCOM RECORD BUFFER ADDRESS      2    BINARY  00008770
 5:   FLGDEF   LOCAL FLAG WORD                   2    BINARY  00008780
 6:   OPANIC1  BAD CALL CODE PANIC CODE          2    BINARY  00008790
 7:   OPANIC2  BAD RETURN CODE PANIC CODE        2    BINARY  00008800
                                                              00008810
      LOCAL:                                                  00008820
                                                              00008830
 1:   SAVE.1   TEMPORARY SAVE #1                 2    BINARY  00008840
 2:   BNRTXT   BATCH NO LONGER THERE MESSAGE     50   EBCDIC  00008850
 3:   ONTXT    ORDER NOT ON FILE MESSAGE         32   EBCDIC  00008860
                                                              00008870
      OUTPUT:                                                 00008880
                                                              00008890
 1:   OID      ORDER ID                          6    EBCDIC  00008900
 2:   ORNO     ORDER RELATIVE NUMBER             2    BINARY  00008910
 3:   OQSTG    ORDER QUANTITY STAGED             2    BINARY  00008920
 4:   OQNPKD   ORDER QUANTITY NOT PICKED         2    BINARY  00008930
 5:   OQISORT  ORDER QUANTITY IN SORTATION       2    BINARY  00008940
 6:   OQXRP    ORDER QUANTITY EXCESS REPACK      2    BINARY  00008950
 7:   OQSTO    ORDER QUANTITY STOCK OUT          2    BINARY  00008960
 8:   OQMIS    ORDER QUANTITY MIS-SORT           2    BINARY  00008970
 9:   FLGDEF   UPDATED LOCAL FLAG WORD           2    BINARY  00008980
10:   PANIC    UPDATED PANIC WORK AREA           2    BINARY  00008990
                                                              00009000
190I  FIND TRANSACTION CALL: OICCFT                           00009010
                                                              00009020
      LABEL:   DESCRIPTION                     BYTES: TYPE:   00009030
                                                              00009040
      INPUT:                                                  00009050
                                                              00009060
 1:   TCNT     TRANSCTION COUNTER TO START ACCESS 2    BINARY  00009070
 2:   BTCNT    BATCH TRANSACTION COUNTER         2    BINARY  00009080
 3:   TSBB     SYSCOM BLOCK BUFFER ADDRESS       2    BINARY  00009090
 4:   TFOFF    TRANSACTION FILE ENTRY OFFSET     2    BINARY  00009100
 5:   ORNO     ORDER RELATIVE NUMBER             2    BINARY  00009110
 6:   FLGDEF   LOCAL FLAG WORD                   2    BINARY  00009120
                                                              00009130
      LOCAL:                                                  00009140
                                                              00009150
 1:   T1FLG    TEMPORARY OK FLAG                 2    BINARY  00009160
 2:   T2FLG    TEMPORARY ORDER SEQUENCE FLAG     2    BINARY  00009170
 3:   SAVE.2   TEMPORARY SAVE #2                 2    BINARY  00009180
                                                              00009190
      OUTPUT:                                                 00009200
                                                              00009210
 1:   TID      TRANSACTION ID                    4    BINARY  00009220
 2:   TSTATUS  TRANSACTION STATUS                2    BINARY  00009230
 3:   TRON     TRANSACTION RELATIVE ORDER NUMBER 2    BINARY  00009240
 4:   TPROD    TRANSACTION PRODUCT ID            6    EBCDIC  00009250
 5:   TSOURCE  TRANSACTION SOURCE                6    EBCDIC  00009260
 6:   TDEST    TRANSACTION DESTINATION           2    BINARY  00009270
 7:   FLGDEF   UPDATED LOCAL FLAG WORD           2    BINARY  00009280
 8:   TCNT     UPDATED TRANSACTION COUNTER       2    BINARY  00009290
                                                              00009300
200I  SEQUENTIAL TRANSACTION CALL: OICCST                     00009310
                                                              00009320
      LABEL:   DESCRIPTION:                    BYTES: TYPE:   00009330
                                                              00009340
```

```
INPUT:                                                                  00009350
                                                                        00009360
 1:  TCNT      TRANSACTION COUNTER TO ACCESS FILE  2    BINARY         00009370
 2:  TFBF      TRANSACTION FILE BLOCKING FACTOR    2    BINARY         00009380
 3:  TFOFF     TRANSACTION FILE ENTRY OFFSET       2    BINARY         00009390
 4:  TFRS      TRANSACTION FILE RECORD SIZE        2    BINARY         00009400
 5:  TFBLK     TRANSACTION FILE BLOCK NUMBER       2    BINARY         00009410
 6:  TIPM      IPM BUFFER ADDRESS                  2    BINARY         00009420
 7:  BID       BATCH ID                            3    EBCDIC         00009430
 8:  SBB       SYSCOM BLOCK BUFFER ADDRESS         2    BINARY         00009440
 9:  TPANIC1   BAD CALL CODE PANIC CODE            2    BINARY         00009450
10:  TPANIC2   BAD BLOCK NUMBER PANIC CODE         2    BINARY         00009460
11:  TPANIC3   BAD RETURN CODE PANIC CODE          2    BINARY         00009470
12:  BOCNT     BATCH ORDER COUNT                   2    BINARY         00009480
13:  BTCNT     BATCH TRANSACTION COUNT             2    BINARY         00009490
                                                                        00009500
LOCAL:                                                                  00009510
                                                                        00009520
 1:  TMP1      TEMPORARY WORK AREA                 2    BINARY         00009530
 2:  SAVE.1    TEMPORARY SAVE #1                   2    BINARY         00009540
 3:  SAVE.2    TEMPORARY SAVE #2                   2    BINARY         00009550
 4:  DATBUF    TRANSACTION SECTOR BUFFER         256    BINARY         00009560
                                                                        00009570
OUTPUT:                                                                 00009580
                                                                        00009590
 1:  TCNT      UPDATED TRANSACTION COUNTER         2    BINARY         00009600
 2:  TFBLK     UPDATED FILE BLOCK NUMBER           2    BINARY         00009610
 3:  SBB       UPDATED SYSCOM BLOCK BUFFER ADDR.   2    BINARY         00009620
 4:  TSBB      UPDATED TRANSACTION BLOCK BUFR      2    BINARY         00009630
 5:  PANIC     UPDATED PANIC WORK AREA             2    BINARY         00009640
 6:  FLGDEF    UPDATED LOCAL FLAG WORD             2    BINARY         00009650
 7:  OCNT      UPDATED ORDER COUNTER               2    BINARY         00009660
                                                                        00009670
210I  GET TRANSACTION CALL: DICCGT                                      00009680
                                                                        00009690
      LABEL:   DESCRIPTION:                  BYTES:  TYPE:             00009700
                                                                        00009710
INPUT:                                                                  00009720
                                                                        00009730
 1:  TIPM      IPM BUFFER ADDRESS                  2    BINARY         00009740
 2:  TID       TRANSACTION ID                      4    BINARY         00009750
 3:  TSRB      SYSCOM RECORD BUFFER ADDRESS        2    BINARY         00009760
 4:  TPANIC1   BAD CALL CODE PANIC CODE            2    BINARY         00009770
 6:  TPANIC2   BAD RETURN CODE PANIC CODE          2    BINARY         00009780
 7:  FLGDEF    LOCAL FLAG WORD                     2    BINARY         00009790
                                                                        00009800
LOCAL:                                                                  00009810
                                                                        00009820
 1:  SAVE.1    TEMPORARY SAVE #1                   2    BINARY         00009830
                                                                        00009840
OUTPUT:                                                                 00009850
                                                                        00009860
 1:  TSTATUS   TRANSACTION STATUS                  2    BINARY         00009870
 2:  TRON      TRANSACTION RELATIVE ORDER NUMBER   2    BINARY         00009880
 3:  TPROD     TRANSACTIONPRODUCT ID               6    EBCDIC         00009890
 4:  TSOURCE   TRANSACTION SOURCE                  6    EBCDIC         00009900
 5:  TDEST     TRANSACTION DESTINATION             2    BINARY         00009910
 6:  PANIC     UPDATED PANIC WORK AREA             2    BINARY         00009920
 7:  FLGDEF    UPDATED LOCAL FLAG WORD             2    BINARY         00009930
                                                                        00009940
220I  READ BATCH CALL: DICCRB                                           00009950
                                                                        00009960
      LABEL:   DESCRIPTION:                  BYTES:  TYPES:            00009970
                                                                        00009980
INPUT:                                                                  00009990
                                                                        00010000
 1:  DCMD      DISKETTE COMMAND WORD               2    BINARY         00010010
 2:  TEMP      TEMPORARY WORK AREA               2-8    BINARY         00010020
 3:  DPANIC1   BAD CALL CODE PANIC CODE            2    BINARY         00010030
 4:  DPANIC2   BAD RETURN CODE PANIC CODE          2    BINARY         00010040
 5:  BSRB      SYSCOM RECORD BUFFER                2    BINARY         00010050
 6:  FLGDEF    LOCAL FLAG WORD                     2    BINARY         00010060
                                                                        00010070
```

```
LOCAL:                                                                  00010080
                                                                        00010090
 1:  RBTXT1     FORMAT ERROR MESSAGE                    28    EBCDIC   00010100
 2:  RBTXT1     I/O ERROR MESSAGE                       34    EBCDIC   00010110
                                                                        00010120
OUTPUT:                                                                 00010130
                                                                        00010140
 1:  BSRB       UPDATED RECORD BUFFER                    2    BINARY   00010150
 2:  PANIC      UPDATED PANIC WORK AREA                  2    BINARY   00010160
 3:  FLGDEF     UPDATED LOCAL FLAG WORD                  2    BINARY   00010170
                                                                        00010180
230I  WRITE BATCH CALL: DICCWB                                          00010190
                                                                        00010200
     LABEL:    DESCRIPTION:                          BYTES:  TYPE:     00010210
                                                                        00010220
INPUT:                                                                  00010230
                                                                        00010240
 1:  DCMD       DISKETTE COMMAND WORD                    2    BINARY   00010250
 2:  DIWRB      PROGRAM TASK WORD ADDRESS                2    BINARY   00010260
 3:  DPANIC1    BAD CALL CODE PANIC CODE                 2    BINARY   00010270
 4:  DPANIC2    BAD RETURN CODE PANIC CODE               2    BINARY   00010280
 5:  BID        BATCH ID                                 3    EBCDIC   00010290
 6:  BHDR       BATCH HEADER                            50    EBCDIC   00010300
 7:  TID        TRANSACTION ID                           4    BINARY   00010310
 8:  OID        ORDER ID                                 6    EBCDIC   00010320
 9:  TPROD      TRANSACTION PRODUCT ID                   6    EBCDIC   00010330
10:  TDEST      TRANSACTION DESTINATION                  2    BINARY   00010340
11:  TSOURCE    TRANSACTION SOURCE                       6    EBCDIC   00010350
12:  TSTATUS    TRANSACTION STATUS                       2    BINARY   00010360
13:  FLGDEF     LOCAL FLAG WORD                          2    BINARY   00010370
                                                                        00010380
LOCAL:                                                                  00010390
                                                                        00010400
 1:  TEMP       TEMPORARY WORK AREA                      2    BINARY   00010410
 2:  SAVE.1     TEMPORARY SAVE #1                        2    BINARY   00010420
 3:  WBTXT1     I/O ERROR MESSAGE                       34    EBCDIC   00010430
                                                                        00010440
OUTPUT:                                                                 00010450
                                                                        00010460
 1:  PANIC      UPDATED PANIC WORK AREA                  2    BINARY   00010470
 2:  TCNT       UPDATED TRANSACTION COUNTER              2    BINARY   00010480
 3:  FLGDEF     UPDATED LOCAL FLAG WORD                  2    BINARY   00010490
                                                                        00010500
240I  ADD BATCH CALL: DICCAB                                            00010510
                                                                        00010520
     LABEL:    DESCRIPTION:                          BYTES:  TYPE:     00010530
                                                                        00010540
INPUT:                                                                  00010550
                                                                        00010560
 1:  BIPM       IPM BUFFER ADDRESS                       2    BINARY   00010570
 2:  BID        BATCH ID                                 3    EBCDIC   00010580
 3:  BHDR       BATCH HEADER                            50    EBCDIC   00010590
 4:  BSRB       SYSCOM RECOD BUFFER ADDRESS              2    BINARY   00010600
 5:  BPANIC1    BATCH ALREADY ADDING PANIC CODE          2    BINARY   00010610
 6:  BPANIC2    BAD CALL CODE PANIC CODE                 2    BINARY   00010620
 7:  BPANIC3    BAD RETURN CODE PANIC CODE               2    BINARY   00010630
 8:  FLGDEF     LOCAL FLAG WORD                          2    BINARY   00010640
                                                                        00010650
LOCAL:                                                                  00010660
                                                                        00010670
 1:  SAVE.1     TEMPORARY SAVE #1                        2    BINARY   00010680
 2:  ABTXT1     SPACE EXHAUSTED MESSAGE                 34    EBCDIC   00010690
 3:  ABTXT2     ALREADY ON FILE MESSAGE                 32    EBCDIC   00010700
                                                                        00010710
OUTPUT:                                                                 00010720
                                                                        00010730
 1:  PANIC      UPDATED PANIC WORK AREA                  2    BINARY   00010740
 2:  FLGDEF     UPDATED LOCAL FLAG WORD                  2    BINARY   00010750
                                                                        00010760
250I  ADD TRANSACTION CALL: DICCAT                                      00010770
                                                                        00010780
     LABEL:    DESCRIPTION:                          BYTES:  TYPE:     00010790
                                                                        00010800
```

```
INPUT:                                                                  00010810
                                                                        00010820
 1:  TIPM      IPM BUFFER ADDRESS                    2    BINARY       00010830
 2:  TSRB      SYSCOM RECORD BUFFER ADDRESS          2    BINARY       00010840
 3:  TPANIC1   BAD CALL CODE PANIC CODE              2    BINARY       00010850
 4:  TPANIC2   BAD RETURN CODE PANIC CODE            2    BINARY       00010860
 5:  TID       TRANSACTION ID                        4    BINARY       00010870
 6:  OID       ORDER ID                              6    EBCDIC       00010880
 7:  TPROD     TRANSACTION PRODUCT ID                6    EBCDIC       00010890
 8:  TDEST     TRANSACTION DESTINATION               2    BINARY       00010900
 9:  TSOURCE   TRANSACTION SOURCE                    6    EBCDIC       00010910
10:  TSTATUS   TRANSACTION STATUS                    2    BINARY       00010920
11:  FLGDEF    LOCAL FLAG WORD                       2    BINARY       00010930
                                                                        00010940
LOCAL:                                                                  00010950
                                                                        00010960
 1:  TFLG      TEMPORARY FLAG WORD                   2    BINARY       00010970
 2:  SAVE.1    TEMPORARY SAVE #1                     2    BINARY       00010980
 3:  ATTXT1    ALREADY ON FILE MESSAGE              36    EBCDIC       00010990
 4:  ATTXT2    TRANSACTION SPACE EXHAUSTED MSG      40    EBCDIC       00011000
 5:  ATTXT3    ORDER SPACE EXHAUSTED MESSAGE        34    EBCDIC       00011010
                                                                        00011020
OUTPUT:                                                                 00011030
                                                                        00011040
 1:  PANIC     UPDATED PANIC WORK AREA               2    BINARY       00011050
 2:  FLGDEF    UPDATED LOCAL FLAG WORD               2    BINARY       00011060
                                                                        00011070
260I  OPERATOR INTERFACE CONTROL LOCAL DATA:                            00011080
                                                                        00011090
      LABEL:   DESCRIPTION:                    BYTES: TYPE:            00011100
                                                                        00011110
 1:  OIEND     END ECB OF OPERATOR INTERFACE         3    BINARY       00011120
 2:  MSGLDER   LOAD ERROR MESSAGE                   26    EBCDIC       00011130
 3:  MSGDIU    DISKETTE IN USE MESSAGE               30    EBCDIC       00011140
 4:  MSGRTRY   RETRY COMMAND MESSAGE                 22    EBCDIC       00011150
 5:  PANIC     PANIC WORK AREA                       2    BINARY       00011160
 6:  OITSK1CD  LOAD TASK #1 CODE                     2    BINARY       00011170
 7:  OITSK2CD  LOAD TASK #2 CODE                     2    BINARY       00011180
 8:  OITSK3CD  LOAD TASK #3 CODE                     2    BINARY       00011190
 9:  OITSK4CD  LOAD TASK #4 CODE                     2    BINARY       00011200
10:  LDRES     LOAD PROGRAM QUEUE CONTROL BLOCK     10    BINARY       00011210
                                                                        00011220
270I  WORK SCHEDULE REPORT LOCAL DATA:                                  00011230
                                                                        00011240
      LABEL:   DESCRIPTION:                    BYTES: TYPE:            00011250
                                                                        00011260
 1:  FLGDEF    LOCAL FLAG WORD                       2    BINARY       00011270
 2:  IPM       IPM BUFFER ADDRES                     2    BINARY       00011280
 3:  IPMPC1    IPM PANIC CODE 1                      2    BINARY       00011290
 4:  IPMPC2    IPM PANIC CODE 2                      2    BINARY       00011300
 5:  SRB       SYSCOM RECORD BUFFER ADDRESS          2    BINARY       00011310
 6:  SRBPC1    SRB PANIC CODE 1                      2    BINARY       00011320
 7:  SRBPC2    SRB PANIC CODE 2                      2    BINARY       00011330
 8:  TIME      TABLE:HH,MM,SS,MO,DA,YR              12    BINARY       00011340
 9:  PAGECNT   REPORT PAGE COUNT                     2    BINARY       00011350
10:  LINECNT   REPORT LINE COUNT                     2    BINARY       00011360
11:  TEMP      TEMPORARY WORK AREA                   4    BINARY       00011370
12:  PANIC     PANIC CODE WORK AREA                  2    BINARY       00011380
13:  MSGWSR    BANNER                               22    EBCDIC       00011390
14:  FLG1      AUXILIARY FLAG WORD                   2    BINARY       00011400
15:  CRTCNT    CRT MAXIMUM SCREEN SIZE               2    BINARY       00011410
                                                                        00011420
280I  BATCH STATUS REPORT LOCAL DATA:                                   00011430
                                                                        00011440
      LABEL:   DESCRIPTION:                    BYTES: TYPE:            00011450
                                                                        00011460
 1:  FLGDEF    LOCAL FLAG WORD                       2    BINARY       00011470
 2:  IPM       IPM BUFFER ADDRES                     2    BINARY       00011480
 3:  IPMPC1    IPM PANIC CODE 1                      2    BINARY       00011490
 4:  IPMPC2    IPM PANIC CODE 2                      2    BINARY       00011500
 5:  SRB       SYSCOM RECORD BUFFER ADDRESS          2    BINARY       00011510
 6:  SRBPC1    SRB PANIC CODE 1                      2    BINARY       00011520
 7:  SRBPC2    SRB PANIC CODE 2                      2    BINARY       00011530
 8:  TIME      TABLE:HH,MM,SS,MO,DA,YR              12    BINARY       00011540
```

```
 9:  PAGECNT   REPORT PAGE COUNT                    2    BINARY  00011550
10:  LINECNT   REPORT LINE COUNT                    2    BINARY  00011560
11:  CRTCNT    CRT MAXIMUM SCREEN SIZE              2    BINARY  00011570
12:  PRINTCNT  PRINTER MAXIMUM LINE SIZE            2    BINARY  00011580
13:  TEMP      TEMPORARY WORK AREA                 10    BINARY  00011590
14:  PANIC     PANIC CODE WORK AREA                 2    BINARY  00011600
15:  MSGBSR    BANNER                              22    EBCDIC  00011610
                                                                 00011620
29DI    ORDER STATUS REPORT LOCAL DATA:                          00011630
                                                                 00011640
     LABEL:    DESCRIPTION:                    BYTES: TYPE:      00011650
                                                                 00011660
 1:  FLGDEF    LOCAL FLAG WORD                      2    BINARY  00011670
 2:  IPM       IPM BUFFER ADDRES                    2    BINARY  00011680
 3:  IPMPC1    IPM PANIC CODE 1                     2    BINARY  00011690
 4:  IPMPC2    IPM PANIC CODE 2                     2    BINARY  00011700
 5:  SRB       SYSCOM RECORD BUFFER ADDRESS         2    BINARY  00011710
 6:  SRBPC1    SRB PANIC CODE 1                     2    BINARY  00011720
 7:  SRBPC2    SRB PANIC CODE 2                     2    BINARY  00011730
 8:  SBB       SYSCOM BLOCK BUFFER ADDRESS          2    BINARY  00011740
 9:  SBBPC1    SBB PANIC CODE 1                     2    BINARY  00011750
10:  RTCNT     REPORT TRANSACTION COUNTER           2    BINARY  00011760
11:  TIME      TABLE:HH,MM,SS,MO,DA,YR             12    BINARY  00011770
12:  PAGECNT   REPORT PAGE COUNT                    2    BINARY  00011780
13:  LINECNT   REPORT LINE COUNT                    2    BINARY  00011790
14:  CRTCNT    CRT MAXIMUM SCREEN SIZE              2    BINARY  00011800
15:  PRINTCNT  PRINTER MAXIMUM LINE SIZE            2    BINARY  00011810
16:  TEMP      TEMPORARY WORK AREA                 10    BINARY  00011820
17:  PANIC     PANIC CODE WORK AREA                 2    BINARY  00011830
18:  MSGOSR    BANNER                              22    EBCDIC  00011840
19:  MSGABORT  REPORT ABORTED                      48    EBCDIC  00011850
                                                                 00011860
30DI    TRANSACTION STATUS REPORT LOCAL DATA:                    00011870
                                                                 00011880
     LABEL:    DESCRIPTION:                    BYTES: TYPE:      00011890
                                                                 00011900
 1:  FLGDEF    LOCAL FLAG WORD                      2    BINARY  00011910
 2:  IPM       IPM BUFFER ADDRES                    2    BINARY  00011920
 3:  IPMPC1    IPM PANIC CODE 1                     2    BINARY  00011930
 4:  IPMPC2    IPM PANIC CODE 2                     2    BINARY  00011940
 5:  SRB       SYSCOM RECORD BUFFER ADDRESS         2    BINARY  00011950
 6:  SRBPC1    SRB PANIC CODE 1                     2    BINARY  00011960
 7:  SRBPC2    SRB PANIC CODE 2                     2    BINARY  00011970
 8:  TIME      TABLE:HH,MM,SS,MO,DA,YR             12    BINARY  00011980
 9:  PAGECNT   REPORT PAGE COUNT                    2    BINARY  00011990
10:  LINECNT   REPORT LINE COUNT                    2    BINARY  00012000
11:  TEMP      TEMPORARY WORK AREA                 12    BINARY  00012010
12:  PANIC     PANIC CODE WORK AREA                 2    BINARY  00012020
13:  MSGTSR    BANNER                              28    EBCDIC  00012030
14:  MSGFTN    TRANSACTION NOT ON FILE             32    EBCDIC  00012040
15:  CRTCNT    CRT MAXIMUM SCREEN SIZE              2    BINARY  00012050
16:  DZERO     DWORD CONSTANT 0                     4    BINARY  00012060
17:  TRANSMAX  DWORD CONSTANT 999999                4    BINARY  00012070
                                                                 00012080
31DI    SORT TRANSLATION REPORT LOCAL DATA:                      00012090
                                                                 00012100
     LABEL:    DESCRIPTION:                    BYTES: TYPE:      00012110
                                                                 00012120
 1:  TIME      TABLE:HH,MM,SS,MO,DA,YR             12    BINARY  00012130
 2:  PAGECNT   REPORT PAGE COUNT                    2    BINARY  00012140
 3:  TEMP      TEMPORARY WORK AREA                 10    BINARY  00012150
 4:  MSGSTR    BANNER                              26    EBCDIC  00012160
 5:  SLCNT     SORT LINE COUNT                      2    BINARY  00012170
 6:  MAXSL     MAXIMUM SORT LINES                   2    BINARY  00012180
 7:  SLRBASE   ROW SORT LINE BASE NUMBER            2    BINARY  00012190
 8:  ROWBASE   ROW SORT LINE BASE ADDRESS           2    BINARY  00012200
 9:  SLCBASE   COLUMN SORT LINE BASE NUMBER         2    BINARY  00012210
10:  COLBASE   COLUMN SORT LINE BASE ADDRESS        2    BINARY  00012220
11:  SHDR1     REPORT HEADER LINE 1                82    EBCDIC  00012230
12:  SHDR2     REPORT HEADER LINE 2                16    EBCDIC  00012240
13:  SHDR3     REPORT HEADER LINE 3                16    EBCDIC  00012250
14:  SDTL      REPORT DETAIL LINE                  16    EBCDIC  00012260
15:  FLGDEF    LOCAL FLAG WORD                      2    BINARY  00012270
16:  CRTCNT    CRT MAXIMUM SCREEN SIZE              2    BINARY  00012280
                                                                 00012290
```

```
320I    READ BATCH LOCAL DATA:                                                  00012300
                                                                                00012310
          LABEL:    DESCRIPTION:                          BYTES: TYPE:        00012320
                                                                                00012330
        1:  FLGDEF    LOCAL FLAG WORD                        2    BINARY      00012340
        2:  IPM       IPM BUFFER ADDRES                      2    BINARY      00012350
        3:  IPMPC1    IPM PANIC CODE 1                       2    BINARY      00012360
        4:  IPMPC2    IPM PANIC CODE 2                       2    BINARY      00012370
        5:  SRB       SYSCOM RECORD BUFFER ADDRESS           2    BINARY      00012380
        6:  SRBPC1    SRB PANIC CODE 1                       2    BINARY      00012390
        7:  SRBPC2    SRB PANIC CODE 2                       2    BINARY      00012400
        8:  TEMP      TEMPORARY WORK AREA                    8    BINARY      00012410
        9:  PANIC     PANIC CODE WORK AREA                   2    BINARY      00012420
       10:  DCMD      DISKETTE COMMAND WORD                  2    BINARY      00012430
       11:  DPANIC1   DISKETTE PANIC CODE 1                  2    BINARY      00012440
       12:  DPANIC2   DISKETTE PANIC CODE 2                  2    BINARY      00012450
       13:  DZERO     DWORD CONSTANT 0                       4    BINARY      00012460
       14:  D999999   DWORD CONSTANT 999999                  4    BINARY      00012470
       15:  TVERCNT   TRANS. VALIDATION ERROR ADDR. CNT.     2.   BINARY      00012480
       16:  FLAG      DISKETTE MOUNTED FLAG                  2    BINARY      00012490
       17:  COMMA     CONSTANT ","                           2    EBCDIC      00012500
       18:  MSGTID    TRANSACTION NO.                       18    EBCDIC      00012510
       19:  MSGDEST   DESTINATION                           14    EBCDIC      00012520
       20:  MSGSTAT   STATUS                                 8    EBCDIC      00012530
       21:  MSGTVER   VALIDATION ERROR                      38    EBCDIC      00012540
                                                                                00012550
330I    WRITE BATCH LOCAL DATA:                                                 00012560
                                                                                00012570
          LABEL:    DESCRIPTION:                          BYTES: TYPE:        00012580
                                                                                00012590
        1:  FLGDEF    LOCAL FLAG WORD                        2    BINARY      00012600
        2:  IPM       IPM BUFFER ADDRES                      2    BINARY      00012610
        3:  IPMPC1    IPM PANIC CODE 1                       2    BINARY      00012620
        4:  IPMPC2    IPM PANIC CODE 2                       2    BINARY      00012630
        5:  SBB       SYSCOM BLOCK BUFFER ADDRESS            2    BINARY      00012640
        6:  SBBPC1    SBB PANIC CODE 1                       2    BINARY      00012650
        7:  SBBPC2    SBB PANIC CODE 2                       2    BINARY      00012660
        8:  TEMP      TEMPORARY WORK AREA                    8    BINARY      00012670
        9:  PANIC     PANIC CODE WORK AREA                   2    BINARY      00012680
       10:  DCMD      DISKETTE COMMAND WORD                  2    BINARY      00012690
       11:  DPANIC1   DISKETTE PANIC CODE 1                  2    BINARY      00012700
       12:  DPANIC2   DISKETTE PANIC CODE 2                  2    BINARY      00012710
       13:  ORTABLE   ORDER REFERENCE TABLE                210    EBCIDIC     00012720
       14:  FLAG      EXTRA FLAG WORD                        2    BINARY      00012730
       15:  SRB       SYSCOM RECORD BUFFER ADDRESS           2    BINARY      00012740
       16:  SRBPC1    SRB PANIC CODE 1                       2    BINARY      00012750
       17:  SRBPC2    SRB PANIC CODE 2                       2    BINARY      00012760
                                                                                00012770
340I    START BATCH LOCAL DATA:                                                 00012780
                                                                                00012790
          LABEL:    DESCRIPTION:                          BYTES: TYPE:        00012800
                                                                                00012810
        1:  FLGDEF    LOCAL FLAG WORD                        2    BINARY      00012820
        2:  IPM       IPM BUFFER ADDRES                      2    BINARY      00012830
        3:  IPMPC1    IPM PANIC CODE 1                       2    BINARY      00012840
        4:  IPMPC2    IPM PANIC CODE 2                       2    BINARY      00012850
        5:  SRB       SYSCOM RECORD BUFFER ADDRESS           2    BINARY      00012860
        6:  SRBPC1    SRB PANIC CODE 1                       2    BINARY      00012870
        7:  SRBPC2    SRB PANIC CODE 2                       2    BINARY      00012880
        8:  TEMP      TEMPORARY WORK AREA                    8    BINARY      00012890
        9:  PANIC     PANIC CODE WORK AREA                   2    BINARY      00012900
       10:  KAOBFQ    ALARM BUFFER QUEUE ADDRESS             2    BINARY      00012910
       11:  KAOALQ    ALARM QUEUE ADDRESS                    2    BINARY      00012920
       12:  ALARM     ALARM CODE                             2    BINARY      00012930
       13:  ALARMPC1  ALARM PANIC CODE 1                     2    BINARY      00012940
       14:  ALARMPC2  ALARM PANIC CODE 2                     2    BINARY      00012950
                                                                                00012960
350I    END BATCH LOCAL DATA:                                                   00012970
                                                                                00012980
          LABEL:    DESCRIPTION:                          BYTES: TYPE:        00012990
                                                                                00013000
        1:  FLGDEF    LOCAL FLAG WORD                        2    BINARY      00013010
        2:  IPM       IPM BUFFER ADDRES                      2    BINARY      00013020
```

```
     3:  IPMPC1     IPM PANIC CODE 1                        2    BINARY  00013030
     4:  IPMPC2     IPM PANIC CODE 2                        2    BINARY  00013040
     5:  SRB        SYSCOM RECORD BUFFER ADDRESS            2    BINARY  00013050
     6:  SRBPC1     SRB PANIC CODE 1                        2    BINARY  00013060
     7:  SRBPC2     SRB PANIC CODE 2                        2    BINARY  00013070
     8:  TEMP       TEMPORARY WORK AREA                     8    BINARY  00013080
     9:  PANIC      PANIC CODE WORK AREA                    2    BINARY  00013090
    10:  KAOBFQ     ALARM BUFFER QUEUE ADDRESS              2    BINARY  00013100
    11:  KAOALQ     ALARM QUEUE ADDRESS                     2    BINARY  00013110
    12:  ALARM      ALARM CODE                              2    BINARY  00013120
    13:  ALARMPC1   ALARM PANIC CODE 1                      2    BINARY  00013130
    14:  ALARMPC2   ALARM PANIC CODE 2                      2    BINARY  00013140
                                                                        00013150
360I DELETE BATCH LOCAL DATA:                                           00013160
                                                                        00013170
          LABEL:    DESCRIPTION:                          BYTES: TYPE:  00013180
                                                                        00013190
     1:  FLGDEF     LOCAL FLAG WORD                         2    BINARY  00013200
     2:  IPM        IPM BUFFER ADDRESS #1                   2    BINARY  00013210
     3:  IPMPC1     IPM PANIC CODE 1                        2    BINARY  00013220
     4:  IPMPC2     IPM PANIC CODE 2                        2    BINARY  00013230
     5:  SRB        SYSCOM RECORD BUFFER ADDRESS            2    BINARY  00013240
     6:  SRBPC1     SRB PANIC CODE 1                        2    BINARY  00013250
     7:  SRBPC2     SRB PANIC CODE 2                        2    BINARY  00013260
     8:  TEMP       TEMPORARY WORK AREA                     8    BINARY  00013270
     9:  PANIC      PANIC CODE WORK AREA                    2    BINARY  00013280
    10:  IPM2       IPM BUFFER ADDRESS #2                   2    BINARY  00013290
    11:  DBTABLE    DELETION BATCH TABLE ADDRESS            2    BINARY  00013300
    11:  BTCH       RELATIVE BATCH NUMBER                   2    BINARY  00013310
    12:  ACTBAT     ACTIVE BATCH FLAG WORD                  2    BINARY  00013320
    13:  QSTFLG     QUESTIONED FLAG WORD                    2    BINARY  00013330
    14:  PRTFLG     PRINT FLAG WORD                         2    BINARY  00013340
    15:  DMSG1      BATCH NOT ON FILE                      26    EBCDIC  00013350
    16:  DMSG2      BATCH MUST BE PENDING OR COMPLETE      72    EBCDIC  00013360
    17:  DMSG3      NO CURRENT BATCH INFORMATION           36    EBCDIC  00013370
    18:  DMSG4      ALL CURRENT BATCHES ARE ACTIVE         56    EBCDIC  00013380
                                                                        00013390
370I REROUTE SHIPPING LINE LOCAL DATA:                                  00013400
                                                                        00013410
          LABEL:    DESCRIPTION:                          BYTES: TYPE:  00013420
                                                                        00013430
     1:  PANIC      PANIC CODE WORK AREA                    2    BINARY  00013440
     2:  STB        SORT TRANSLATION BUFFER               260    BINARY  00013450
     3:  TEMP       TEMPORARY WORK AREA                     2    BINARY  00013460
     4:  SAVE       SAVE OLD SORT LINE                      2    BINARY  00013470
     5:  NEWSL      NEW SORT LINE ASSIGNMENT                2    BINARY  00013480
     6:  SLTXT1     INVALID SORT LINE                      34    EBCDIC  00013490
                                                                        00013500
380I RESTORE SHIPPING LINE LOCAL DATA:                                  00013510
                                                                        00013520
          LABEL:    DESCRIPTION:                          BYTES: TYPE:  00013530
                                                                        00013540
     1:  PANIC      PANIC CODE WORK AREA                    2    BINARY  00013550
     2:  STB        SORT TRANSLATION BUFFER               260    BINARY  00013560
                                                                        00013570
390I OPEN PSC COMMUNICATIONS LOCAL DATA:                                00013580
                                                                        00013590
          LABEL:    DESCRIPTION:                          BYTES: TYPE:  00013600
                                                                        00013610
     1:  OIPSCTX1                                          38    EBCDIC  00013620
     2:  OIPSCTX2                                          36    EBCDIC  00013630
     3:  PANIC      PANIC CODE WORK AREA                    2    BINARY  00013640
     4:  PPANIC1    PSC PANIC CODE 1                        2    BINARY  00013650
     5:  PPANIC2    PSC PANIC CODE 2                        2    BINARY  00013660
400I ASSIGN TRANSACTION STATUS LOCAL DATA:                              00013680
                                                                        00013690
          LABEL:    DESCRIPTION:                          BYTES: TYPE:  00013700
                                                                        00013710
     1:  FLGDEF     LOCAL FLAG WORD                         2    BINARY  00013720
     2:  IPM        IPM BUFFER ADDRES                       2    BINARY  00013730
     3:  IPMPC1     IPM PANIC CODE 1                        2    BINARY  00013740
     4:  IPMPC2     IPM PANIC CODE 2                        2    BINARY  00013750
```

```
 5:  SRB       SYSCOM RECORD BUFFER ADDRESS        2    BINARY  00013760
 6:  SRBPC1    SRB PANIC CODE 1                    2    BINARY  00013770
 7:  SRBPC2    SRB PANIC CODE 2                    2    BINARY  00013780
 8:  CRTCNT    CRT MAXIMUM SCREEN SIZE             2    BINARY  00013790
 9:  LINECNT   REPORT LINE COUNT                   2    BINARY  00013800
10:  TEMP      TEMPORARY WORK AREA                10    BINARY  00013810
11:  PANIC     PANIC CODE WORK AREA                2    BINARY  00013820
12:  TMX       END TRANSACTION RANGE ID            4    BINARY  00013830
13:  TNSTATUS  NEW TRANSACTION STATUS              2    BINARY  00013840
14:  MSGTNF    TRANSACTION NOT FOUND              32    EBCDIC  00013850
15:  MSGTRE    TRANSACTION RANGE ERROR            34    EBCDIC  00013860
16:  DZERO     DWORD CONSTANT 0                    4    BINARY  00013870
17:  DWONE     DWORD CONSTANT 1                    4    BINARY  00013880
18:  D999999   DWORD CONSTANT 999999               4    BINARY  00013890
                                                               00013900
410I  RECIRC LOCAL DATA:                                       00013910
                                                               00013920
      LABEL:   DESCRIPTION:                    BYTES: TYPE:    00013930
                                                               00013940
 1:   DIRECTXT  BANNER                          30    EBCDIC  00013950
                                                               00013960
420I  HOLD LOCAL DATA:                                         00013970
                                                               00013980
      LABEL:   DESCRIPTION:                    BYTES: TYPE:    00013990
                                                               00014000
 1:   DIHLTXT  BANNER                          28    EBCDIC  00014010
                                                               00014020
430I  BATCH VERIFICATION DISPLAY LAYOUT:                       00014030
                                                               00014040
      REPORT FIELD:            FROM:                           00014050
                                                               00014060
 1:   DATE                    LOCAL TIME                       00014070
 2:   TIME                    LOCAL TIME                       00014080
 3:   PAGE                    LOCAL PAGECNT                    00014090
 4:   BATCH ID                120I  BID                        00014100
 5:   BATCH STATUS            120I  BSTATUS, BSTBL              00014110
 6:   ORDER COUNT             120I  BOCNT                      00014120
 7:   TRANSACTION COUNT        120I  BTCNT                      00014130
 8:   HEADER INFORMATION       120I  BHDR                       00014140
                                                               00014150
     * SEE REPORT LAYOUT SHEET                                 00014160
                                                               00014170
440I  TRANSACTION VERIFICATION DISPLAY LAYOUT:                 00014180
                                                               00014190
      REPORT FIELD:            FROM:                           00014200
                                                               00014210
 1:   DATE                    LOCAL TIME                       00014220
 2:   TIME                    LOCAL TIME                       00014230
 3:   PAGE                    LOCAL PAGECNT                    00014240
 4:   TRANSACTION ID          140I  TID                        00014250
 5:   TRANSACTION STATUS      140I  TSTATUS                     00014260
 6:   PRODUCT ID              140I  TPROD                      00014270
 7:   SOURCE LOCATION         140I  TSOURCE                    00014280
 8:   DESTINATION             140I  TDEST                      00014290
                                                               00014300
     * SEE REPORT LAYOUT SHEET                                 00014310
                                                               00014320
450I  CHANGE BATCH STATUS CALL: DICCB                          00014330
                                                               00014340
      LABEL:   DESCRIPTION:                    BYTES: TYPE:    00014350
                                                               00014360
     INPUT:                                                    00014370
                                                               00014380
 1:   BIPM      IPM BUFFER ADDRESS                 2    BINARY  00014390
 2:   BID       BATCH ID                           3    EBCDIC  00014400
 3:   BSTATUS   BATCH STATUS                       2    BINARY  00014410
 4:   BPANIC1   BAD CALL CODE PANIC CODE           2    BINARY  00014420
 5:   BPANIC2   BAD STATUS PANIC CODE             2    BINARY  00014430
 6:   BPANIC3   BAD RETURN CODE PANIC CODE        2    BINARY  00014440
 7:   FLGDEF    LOCAL FLAG WORD                    2    BINARY  00014450
                                                               00014460
     LOCAL:                                                    00014470
                                                               00014480
```

```
 1:  SAVE.1     TEMPORARY SAVE #1                  2     BINARY  00014490
 2:  BNTXT      NOT ON FILE MESSAGE               28     EBCDIC  00014500
                                                                  00014510
OUTPUT:                                                           00014520
                                                                  00014530
 1:  PANIC      UPDATED PANIC WORK AREA            2     BINARY  00014540
 2:  FLGDEF     UPDATED LOCAL FLAG WORD            2     BINARY  00014550
                                                                  00014560
46DI  WRITE SORT TRANSLATION TABLE CALL: DICCWS                   00014570
                                                                  00014580
     LABEL:    DESCRIPTION                      BYTES: TYPE:     00014590
                                                                  00014600
INPUT:                                                            00014610
                                                                  00014620
 1:  STB        SORT TRANSLATION BUFFER          260     BINARY  00014630
 2:  DS1        DISK DATA SET                      2     BINARY  00014640
                                                                  00014650
LOCAL:                                                            00014660
                                                                  00014670
 1:  SAVE.2     TEMPORARY SAVE #2                  2     BINARY  00014680
                                                                  00014690
47DI  HELP COMMAND LAYOUT:                                        00014700
                                                                  00014710
* SEE REPORT LAYOUT SHEET                                         00014720
                                                                  00014730
48DI  GET IPM CALL: DICCGI                                        00014740
                                                                  00014750
USER MUST DEFINE IN PROGRAM LOCAL DATA:                           00014760
                                                                  00014770
     LABEL:    DESCRIPTION:                     BYTES: TYPE:     00014780
                                                                  00014790
INPUT:                                                            00014800
                                                                  00014810
 1:  IPMPC1     IPM BUFFER QUEUE EMPTY PANIC CODE  2     BINARY  00014820
                                                                  00014830
OUTPUT:                                                           00014840
                                                                  00014850
 1:  IPM        IPM BUFFER ADDRESS                 2     BINARY  00014860
 2:  PANIC      UPDATED PANIC WORK AREA            2     BINARY  00014870
                                                                  00014880
49DI  GET RECORD BUFFER CALL: DICCGR                              00014890
                                                                  00014900
     LABEL:    DESCRIPTION:                     BYTES: TYPE:     00014910
                                                                  00014920
INPUT:                                                            00014930
                                                                  00014940
 1:  SRBPC1     SRB BUFFER QUEUE EMPTY PANIC CODE  2     BINARY  00014950
                                                                  00014960
OUTPUT:                                                           00014970
                                                                  00014980
 1:  SRB        SYSCOM RECORD BUFFER ADDRESS       2     BINARY  00014990
 2:  PANIC      UPDATED PANIC WORK AREA            2     BINARY  00015000
                                                                  00015010
50DI  FREE IPM CALL: DICLFI                                       00015020
                                                                  00015030
     LABEL:    DESCRIPTION:                     BYTES: TYPE:     00015040
                                                                  00015050
INPUT:                                                            00015060
                                                                  00015070
 1:  IPM        IPM BUFFER ADDRESS                 2     BINARY  00015080
 2:  IPMPC2     IPM BUFFER QUEUE FULL PANIC CODE   2     BINARY  00015090
                                                                  00015100
LOCAL:                                                            00015110
                                                                  00015120
 1:  SAVE.2     TEMPORARY SAVE #2                  2     BINARY  00015130
                                                                  00015140
OUTPUT:                                                           00015150
                                                                  00015160
 1:  PANIC      UPDATED PANIC WORK AREA            2     BINARY  00015170
                                                                  00015180
51DI  FREE RECORD BUFFER CALL: DICCFR                             00015190
                                                                  00015200
     LABEL:    DESCRIPTION:                     BYTES: TYPE:     00015210
                                                                  00015220
```

```
INPUT:                                                                  00015230
                                                                        00015240
 1:  SRB       SYSCOM RECORD BUFFER ADDRESS          2    BINARY       00015250
 2:  SRBPC2    SRB BUFFER QUEUE FULL PANIC CODE      2    BINARY       00015260
                                                                        00015270
LOCAL:                                                                  00015280
                                                                        00015290
 1:  SAVE.2    TEMPORARY SAVE #2                     2    BINARY       00015300
                                                                        00015310
OUTPUT:                                                                 00015320
                                                                        00015330
 1:  PANIC     UPDATED PANIC WORK AREA               2    BINARY       00015340
                                                                        00015350
520I  SEND ALARM MESSAGE CALL: DICCSA                                   00015360
                                                                        00015370
      LABEL:   DESCRIPTION                        BYTES: TYPE:          00015380
                                                                        00015390
INPUT;                                                                  00015400
                                                                        00015410
 1:  ALARM     ALARM CODE                            2    BINARY       00015420
 2:  ALARMPC1  ALARM QUEUE EMPTY PANIC CODE          2    BINARY       00015430
 3:  ALARMPC2  ALARM QUEUE FULL PANIC CODE           2    BINARY       00015440
 4:  BID       BATCH ID                              3    EBCDIC       00015450
                                                                        00015460
LOCAL:                                                                  00015470
                                                                        00015480
 1:  SAVE.2    TEMPOARY SAVE #2                      2    BINARY       00015490
                                                                        00015500
OUTPUT:                                                                 00015510
                                                                        00015520
 1:  PANIC     UPDATED PANIC WORK AREA               2    BINARY       00015530
                                                                        00015540
530I  DELETE BATCH CALL: DICCDB                                         00015550
                                                                        00015560
      LABEL:   DESCRIPTION:                       BYTES: TYPE:          00015570
                                                                        00015580
INPUT:                                                                  00015590
                                                                        00015600
 1:  BIPM      IPM BUFFER ADDRESS                    2    BINARY       00015610
 2:  BPANIC1   BAD RELATIVE BATCH ID PANIC CODE      2    BINARY       00015620
 3:  BPANIC2   BAD CALL CODE PANIC CODE              2    BINARY       00015630
 4:  BPANIC3   BAD RETURN CODE PANIC CODE            2    BINARY       00015640
                                                                        00015650
LOCAL:                                                                  00015660
                                                                        00015670
 1:  SAVE.1    TEMPORARY SAVE #1                     2    BINARY       00015680
                                                                        00015690
OUTPUT:                                                                 00015700
                                                                        00015710
 1:  PANIC     UPDATED PANIC WORK AREA               2    BINARY       00015720
                                                                        00015730
540I  PANIC CALL: DICCPA                                                00015740
                                                                        00015750
      LABEL:   DESCRIPTION:                       BYTES: TYPE:          00015760
                                                                        00015770
INPUT:                                                                  00015780
                                                                        00015790
 1:  PANIC     PAIC CODE                             2    BINARY       00015800
                                                                        00015810
LOCAL:                                                                  00015820
                                                                        00015830
 1:  SAVE.2    TEMPORARY SAVE #2                     2    BINARY       00015840
 2:  NEVEVT    NEVER EVENT CONTROL BLOCK             6    BINARY       00015850
                                                                        00015860
550I  GET BLOCK BUFFER CALL: DICCGS                                     00015870
                                                                        00015880
      LABEL:   DESCRIPTION:                       BYTES: TYPE:          00015890
                                                                        00015900
LOCAL:                                                                  00015910
                                                                        00015920
 1:  SAVE.2    TEMPORARY SAVE #2                     2    BINARY       00015930
                                                                        00015940
OUTPUT:                                                                 00015950
                                                                        00015960
```

```
 1:  SBB       SYSCOM BLOCK BUFFER ADDRESS          2    BINARY  00015970
                                                                  00015980
560I  FREE BLOCK BUFFER CALL: DICCFS                              00015990
                                                                  00016000
      LABEL:  DESCRIPTION:                        BYTES: TYPE:    00016010
                                                                  00016020
INPUT:                                                            00016030
                                                                  00016040
 1:  SBB       SYSCOM BLOCK BUFFER ADDRESS          2    BINARY  00016050
 2:  SBBPC2    SBB BUFFER QUEUE FULL PANIC CODE     2    BINARY  00016060
                                                                  00016070
LOCAL:                                                            00016080
                                                                  00016090
 1:  SAVE.2    TEMPORARY SAVE #2                    2    BINARY  00016100
                                                                  00016110
OUTPUT:                                                           00016120
                                                                  00016130
 1:  PANIC     UPDATED PANIC WORK AREA              2    BINARY  00016140
                                                                  00016150
570I  DISKETTE COMMAND WORD FLAG DEFINITION:                      00016160
                                                                  00016170
      LABEL:  DESCRIPTION:              VALUE: BYTES: TYPE:       00016180
                                                                  00016190
 1:  DCMD      DISKETTE COMMAND WORD                2    BINARY  00016200
 2:  CMDBAT    WRITE BATCH HEADER        0001       0    EQUATE  00016210
 3:  CMDTRA    WRITE TRANSACTION DETAIL  0010       0    EQUATE  00016220
 4:  CMDLR     WRITE LAST RECORD         0011       0    EQUATE  00016230
 5:  CMDHDR    READ BATCH HEADER         0100       0    EQUATE  00016240
 6:  CMDDTL    READ TRANSACTION DETAIL   0101       0    EQUATE  00016250
                                                                  00016260
580I  SYSTEM ERROR REPORT LAYOUT                                  00016270
                                                                  00016280
      REPORT FIELD:              FROM:                            00016290
                                                                  00016300
 1:  DATE                       LOCAL TIME                        00016310
 2:  TIME                       LOCAL TIME                        00016320
 3:  PAGE                       PAGE COUNT                        00016330
 4:  TRANSACTIONS LOST          SEE 40FM                          00016340
 5:  TRANSACTIONS REJECTED      SEE 40FM                          00016350
 6:  TRANSACTIONS RECIRCED                                        00016360
      FROM INDUCTION            SEE 40FM                        00016370
      BY THE PSC                SEE 40FM                        00016380
      TOTAL                     SEE 40FM                        00016390
 7:  NO-READS                   SEE 40FM                          00016400
 8:  MIS-SORTS                  SEE 40FM                          00016410
                                                                  00016420
590I  SYSTEM ERROR REPORT LOCAL DATA                              00016430
                                                                  00016440
      LABEL:  DESCRIPTION:              BYTES: TYPE:              00016450
                                                                  00016460
 1:  HCFLG     HARD COPY FLAG WORD          2    BINARY          00016470
 2:  ABFLG     ABORT REPORT FLAG WORD       2    BINARY          00016480
 3:  TEMP      TEMPORARY STORAGE            4    BINARY          00016490
 4:  SORTLIN   NUMBER OF SORT LINES         2    BINARY          00016500
 5:  COL       NUMBER OF REPORT COLUMNS     2    BINARY          00016510
 6:  ROW       NUMBER OF REPORT ROWS        2    BINARY          00016520
 7:  LNCNT     LINE COUNTER                 2    BINARY          00016530
 8:  CRTCNT    LINES AVAILABLE ON CRT       2    BINARY          00016540
 9:  CNTER     SORT LINE COUNTER            2    BINARY          00016550
                                                                  00016560
600I  MIS-SORT REPORT LOCAL DATA:                                 00016570
                                                                  00016580
      LABEL:  DESCRIPTION:                        BYTES: TYPE:    00016590
                                                                  00016600
 1:  FLGDEF    LOCAL FLAG WORD                      2    BINARY  00016610
 2:  IPM       IPM BUFFER ADDRES                    2    BINARY  00016620
 3:  IPMPC1    IPM PANIC CODE 1                     2    BINARY  00016630
 4:  IPMPC2    IPM PANIC CODE 2                     2    BINARY  00016640
 5:  SRB       SYSCOM RECORD BUFFER ADDRESS         2    BINARY  00016650
 6:  SRBPC1    SRB PANIC CODE 1                     2    BINARY  00016660
 7:  SRBPC2    SRB PANIC CODE 2                     2    BINARY  00016670
 8:  SBB       SYSCOM BLOCK BUFFER ADDRESS          2    BINARY  00016680
 9:  SBBPC1    SBB PANIC CODE 1                     2    BINARY  00016690
10:  RTCNT     REPORT TRANSACTION COUNTER           2    BINARY  00016700
```

```
          11:  TIME       TABLE:HH,MM,SS,MO,DA,YR          12    BINARY  00016710
          12:  PAGECNT    REPORT PAGE COUNT                 2    BINARY  00016720
          13:  LINECNT    REPORT LINE COUNT                 2    BINARY  00016730
          14:  CRTCNT     CRT MAXIMUM SCREEN SIZE           2    BINARY  00016740
          15:  PRINTCNT   PRINTER MAXIMUM LINE SIZE         2    BINARY  00016750
          16:  TEMP       TEMPORARY WORK AREA              10    BINARY  00016760
          17:  PANIC      PANIC CODE WORK AREA              2    BINARY  00016770
          18:  MSGOSR     BANNER                           22    EBCDIC  00016780
          19:  MSGABORT   REPORT ABORTED                   48    EBCDIC  00016790
                                                                         00016800
610I      MIS-SORT REPORT LAYOUT:                                        00016810
                                                                         00016820
               REPORT FIELD:                FROM:                      00016830
                                                                         00016840
           1:  DATE                         290I TIME                  00016850
           2:  TIME                         290I TIME                  00016860
           3:  PAGE                         290I PAGECNT               00016870
           4:  BATCH ID                     120I BID                   00016880
           5:  BATCH STATUS                 120I BSTATUS, BSTBL        00016890
           6:  ORDER COUNT                  120I BOCNT                 00016900
           7:  TRANSACTION COUNT            290I RTCNT                 00016910
           8:  HEADER INFORMATION           120I BHDR                  00016920
           9:  ORDER NUMBER                 130I OCNT                  00016930
          10:  ORDER ID                     130I OID                   00016940
          11:  QUANTITY PLANNED             130I OTCNT                 00016950
          12:  QUANTITY STAGED              130I OQSTG                 00016960
          13:  QUANTITY NOT SHIPPED         130I OQNSTG                00016970
          14:  QUANTITY STOCK-OUT           130I OQSTO                 00016980
          15:  QUANTITY EXCESS REPACK       130I OQXRP                 00016990
          16:  QUANTITY IN SORTATION        130I OQISORT               00017000
          17:  QUANTITY NOT PICKED          130I OQNPKD                00017010
          18:  QUANTITY MIS-SORT            130I OQMIS                 00017020
          19:  ORDER % COMPLETE             130I OPCOM                 00017030
          20:  TRANSACTION NUMBER           290I RTCNT                 00017040
          21:  TRANSACTION ID               140I TID                   00017050
          22:  TRANSACTION STATUS           140I TSTATUS               00017060
          23:  PRODUCT ID                   140I TPROD                 00017070
          24:  SOURCE LOCATION              140I TSOURCE               00017080
          25:  DESTINATION                  140I TDEST                 00017090
          26:  MIS-SORT DESTINATION                                    00017100
                                                                       00017110
          * SEE REPORT LAYOUT SHEET                                    00017120
                                                                       00017130
HELP COMMAND
        STATUS REPORT COMMANDS          :        BATCH CONTROL COMMANDS
                                        :
SCHEDULE  - WORK SCHEDULE BATCHES       : READ     - READ BATCH FROM DISKETTE
BATCH     * BATCH/ORDER STATUS          : START    - START BATCH PROCESSING
ORDER     * ORDER/TRANSACTION STATUS    : END      - END BATCH PROCESSING
TRANSACT  - TRANSACTION STATUS          : WRITE    - WRITE BATCH TO DISKETTE
MISSORT   * MISSORT EXCEPTIONS          : DELETE   - DELETE BATCH INFORMATION
ROUTE     - SORT LINE TRANSLATIONS      : ASSIGN   - ASSIGN TRANSACTION STATUS
                                        :
> CANCEL  * ( TO TERMINATE REPORTS )    :
****************************************:****************************************
                                        :
        SYSTEM CONTROL COMMANDS         :      SORTATION CONTROL COMMANDS
                                        :
HELP      - LIST SYSTEM COMMANDS        : HOLD     - HOLD ON ERROR
RESET     - RESET ERROR COUNTS          : RECIRC   - RECIRCULATE ON ERROR
ERROR     - DISPLAY SYSTEM ERROR COUNTS : OPEN     - OPEN PSC COMMUNICATIONS
$T        - SET SYSTEM TIME             : REROUTE  - ASSIGN ALTERNATE SORT LINE
                                        : RESTORE  - RESTORE ORIGINAL SORT LINE
                                        :
-- BATCH --  NO. OF  NO. OF
ID.  STATUS  ORDERS  TRANS.  HEADER INFORMATION

S01  ACTIVE      3     150   SAMPLE BATCH OF 150 TRANSACTIONS
T02  ACTIVE      1       9   TEST BATCH T02
T03  PENDING     1       8   TEST BATCH T03
T04  PENDING     1       9   TEST BATCH T04
T05  COMPLETE    9       9   TEST BATCH T05
T06  PENDING     1       9   TEST BATCH T06
T07  PENDING     9       9   TEST BATCH T07
```

```
T08  COMPLETE    1          9   TEST BATCH T08
T09  COMPLETE    9          9   TEST BATCH T09
T10  PENDING     1          9   TEST BATCH T10

-- BATCH --  NO. OF  NO. OF
ID.  STATUS  ORDERS  TRANS.  HEADER INFORMATION

S01  ACTIVE      3      150   SAMPLE BATCH OF 150 TRANSACTIONS

      ORDER     QUANTITIES:   NOT      %     STOCK  EXCESS   IN      NOT     MIS-
 #     ID.     PLAN  STAGED  SHIP   COMP.   -OUT  REPACK  SORT.  PICKED   SORT

 1   000001      50     26     24   74.0       2       9      3       7       3
 2   000002      50     35     15   74.0       2       0      4       8       1
 3   000003      50     50      0  100.0       0       0      0       0       0

  BATCH PERCENT COMPLETE:  82.7

1    000001      50     26     24   74.0       2       9      3       7       3

        - TRANSACTION -                PRODUCT      SOURCE
      #   NUMBER    STATUS           IDENTIFICATION  LOCATION   DESTINATION

      1       16  MIS-SORTED           000087        00087      1 ( 2)
      2       91  EXCESS REPACK        000082        00082      1
      3       94  EXCESS REPACK        000085        00085      1
      4       97  MIS-SORTED           000088        00088      1 ( 3)
      5      100  EXCESS REPACK        000091        00091      1
      6      103  EXCESS REPACK        000094        00094      1
      7      106  EXCESS REPACK        000097        00097      1
      8      109  EXCESS REPACK        000100        00100      1
      9      112  EXCESS REPACK        000103        00103      1
     10      115  EXCESS REPACK        000106        00106      1
     11      118  EXCESS REPACK        000109        00109      1
     12      121  STOCK-OUT            000112        00112      1
     13      124  IN SORTATION         000115        00115      1
     14      127  NOT PICKED           000118        00118      1
     15      130  IN SORTATION         000121        00121      1
     16      133  IN SORTATION         000124        00124      1
     17      136  NOT PICKED           000127        00127      1
     18      139  NOT PICKED           000130        00130      1
     19      142  NOT PICKED           000133        00133      1
     20      145  STOCK-OUT            000136        00136      1
     21      148  NOT PICKED           000139        00139      1
     22      151  MIS-SORTED           000142        00142      1 ( 2)
     23      154  NOT PICKED           000145        00145      1
     24      157  NOT PICKED           000148        00148      1

2    000002      50     35     15   74.0       2       0      4       8       1

        - TRANSACTION -                PRODUCT      SOURCE
      #   NUMBER    STATUS           IDENTIFICATION  LOCATION   DESTINATION

      1      113  IN SORTATION         000104        00104      2
      2      116  IN SORTATION         000107        00107      2
      3      119  IN SORTATION         000110        00110      2
      4      122  STOCK-OUT            000113        00113      2
      5      125  STOCK-OUT            000116        00116      2
      6      128  NOT PICKED           000119        00119      2
      7      131  NOT PICKED           000122        00122      2
      8      134  NOT PICKED           000125        00125      2
      9      137  NOT PICKED           000128        00128      2
     10      140  NOT PICKED           000131        00131      2
     11      143  IN SORTATION         000134        00134      2
     12      149  NOT PICKED           000140        00140      2
     13      152  NOT PICKED           000143        00143      2
     14      155  NOT PICKED           000146        00146      2
     15      158  MIS-SORTED           000149        00149      2 ( 1)

3    000003      50     50      0  100.0       0       0      0       0       0

**** NO EXCEPTIONS IN THIS ORDER

**** REPORT COMPLETE
```

- TRANSACTION - NUMBER	STATUS	ORDER ID	PRODUCT ID	SOURCE LOCATION	DESTINATION
109	EXCESS REPACK	000001	000100	00100	1

#	ORDER ID.	QUANTITIES: PLAN	STAGED	NOT SHIP	% COMP.	STOCK -OUT	EXCESS REPACK	IN SORT.	NOT PICKED	MIS-SORT
1	000001	50	26	24	74.0	2	9	3	7	3

#	- TRANSACTION - NUMBER	STATUS	PRODUCT IDENTIFICATION	SOURCE LOCATION	DESTINATION
1	16	MIS-SORTED	000007	00007	1 (2)
2	97	MIS-SORTED	000088	00088	1 (3)
3	151	MIS-SORTED	000142	00142	1 (2)

#	ORDER ID.	QUANTITIES: PLAN	STAGED	NOT SHIP	% COMP.	STOCK -OUT	EXCESS REPACK	IN SORT.	NOT PICKED	MIS-SORT
2	000002	50	35	15	74.0	2	0	4	8	1

#	- TRANSACTION - NUMBER	STATUS	PRODUCT IDENTIFICATION	SOURCE LOCATION	DESTINATION
1	158	MIS-SORTED	000149	00149	2 (1)

#	ORDER ID.	QUANTITIES: PLAN	STAGED	NOT SHIP	% COMP.	STOCK -OUT	EXCESS REPACK	IN SORT.	NOT PICKED	MIS-SORT
3	000003	50	50	0	100.0	0	0	0	0	0

**** NO MIS-SORTS IN THIS ORDER

**** REPORT COMPLETE

SORT LINE	CHANGE TO	SORT LINE	CHANGE TO	SORT LINE	CHANGE TO
1		11		21 ---	19
2 ---	3	12			
3		13			
4		14			
5		15 ---	11		
6 ---	8	16			
7		17			
8		18			
9		19			
10		20			

- TRANSACTIONS - LOST	REJECTED	- SENT TO RECIRCULATION - BY PSC	FROM INDUCTION	TOTAL
0	3	2	5	7

SCANNER NO-READS SCANNER #1	SCANNER #2
0	1

M I S - S O R T S

LINE	COUNT	LINE	COUNT
1	••••• 1	18	••••• 0
2	••••• 2	19	••••• 0
3	••••• 1	20	••••• 0
4	••••• 0	21	••••• 0
5	••••• 0		
6	••••• 0		
7	••••• 0		
8	••••• 0		
9	••••• 0		
10	••••• 0		
11	••••• 0		

```
12 ..... 0
13 ..... 0
14 ..... 0
15 ..... 0
16 ..... 0
17 ..... 0
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
SYSTEM START-UP PROGRAM                                             00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                        - MODULE HISTORY -                          00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                     78-01709                00000130
                                                                    00000140
   SUB-SYSTEM:      AUXILIARY FUNCTIONS                             00000150
                                                                    00000160
   MODULE:          $INIT                   60                      00000170
                                                                    00000180
                                                                    00000190
                        - MODULE ABSTRACT -                         00000200
                                                                    00000210
   THIS MODULE IS THE SYSTEM START-UP PROGRAM. IT LOADS ALL         00000220
OTHER PROGRAMS IN THE SYSTEM AND CALLS THE FILE MANAGER             00000230
TO INITIALIZE THE FILES. AFTER IT HAS DONE ALL THIS, IT             00000240
PERFORMS A PROGSTOP.                                                00000250
WAIT FOR 1 SECOND                                                   00000260
LOAD THE ALARM MESSAGE PRINTING SUB-SYSTEM                          00000270
IF NO LOAD ERRORS ENCOUNTERED THEN                                  00000280
   LOAD THE FILE MANAGER SUB-SYSTEM                                 00000290
   IF LOAD ERROR ENCOUNTERED THEN                                   00000300
07AL  PANIC!                                                        00000310
08AL                                                                00000320
   (ELSE)                                                           00000330
   ENDIF                                                            00000340
09FM  GET A FILE MANAGER INPUT BUFFER                               00000350
   IF NONE AVAILABLE THEN                                           00000360
07AL  PANIC!                                                        00000370
08AL                                                                00000380
   (ELSE)                                                           00000390
   ENDIF                                                            00000400
24FM  SET UP BUFFER TO INITIALIZE THE FILES                         00000410
08FM  CALL THE FILE MANAGER                                         00000420
24FM  IF FILE MANAGER COMPLETION CODE NOT SUCCESSFUL THEN           00000430
07AL  PANIC!                                                        00000440
08AL                                                                00000450
   (ELSE)                                                           00000460
   ENDIF                                                            00000470
09FM  RETURN THE FILE MANAGER BUFFER TO FREE POOL                   00000480
   IF FREE POOL FULL THEN                                           00000490
07AL  PANIC!                                                        00000500
08AL                                                                00000510
   (ELSE)                                                           00000520
   ENDIF                                                            00000530
   LOAD THE PSC COMMUNICATIONS SUB-SYSTEM                           00000540
   IF LOAD ERROR ENCOUNTERED THEN                                   00000550
07AL  PANIC!                                                        00000560
08AL                                                                00000570
   (ELSE)                                                           00000580
   ENDIF                                                            00000590
   LOAD THE SCANNER INPUT SUB-SYSTEM                                00000600
   IF LOAD ERROR ENCOUNTERED THEN                                   00000610
07AL  PANIC!                                                        00000620
08AL                                                                00000630
   (ELSE)                                                           00000640
   ENDIF                                                            00000650
   LOAD THE OPERATOR INTERFACE SUB-SYSTEM                           00000660
   IF LOAD ERROR ENCOUNTERED THEN                                   00000670
07AL  PANIC!                                                        00000680
```

```
08AL                                                                    00000690
          (ELSE)                                                        00000700
          ENDIF                                                         00000710
02AL      GET AN ALARM MESSAGE BUFFER                                   00000720
          IF NONE AVAILABLE THEN                                        00000730
07AL         PANIC!                                                     00000740
08AL                                                                    00000750
          (ELSE)                                                        00000760
          ENDIF                                                         00000770
02AX      IF START-UP CODE = NEW IPL THEN                               00000780
03AL         SET BUFFER TO PRINT "SYSTEM STARTED" MSG.                  00000790
          ELSE                                                          00000800
03AL         SET BUFFER TO PRINT "POWER FAIL/RESTART" MSG.              00000810
          ENDIF                                                         00000820
01AL      PLACE BUFFER ON ALARM MESSAGE INPUT QUEUE                     00000830
          IF QUEUE FULL THEN                                            00000840
07AL         PANIC!                                                     00000850
08AL                                                                    00000860
          (ELSE)                                                        00000870
          ENDIF                                                         00000880
       (ELSE)                                                           00000890
       ENDIF                                                            00000900
       PROGSTOP                                                         00000910

                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
       SYSTEM INSTALLATION PROGRAM                                      00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                          - MODULE HISTORY -                            00000100
                                                                        00000110
                                                                        00000120
          PROJECT:        DAS                  78-01709               00000130
                                                                        00000140
          SUB-SYSTEM:     AUXILIARY FUNCTIONS                           00000150
                                                                        00000160
          MODULE:         $INITIAL             61                       00000170
                                                                        00000180
                                                                        00000190
                          - MODULE ABSTRACT -                           00000200
                                                                        00000210
          THIS MODULE REBUILDS THE ORDER FILE FROM THE TRANSACTION      00000220
       RECORDS IN THE TRANSACTION FILE FOR EVERY BATCH ON FILE. IT      00000230
       THEN LOADS THE SYSTEM START-UP PROGRAM INTO PARTITION 3 AND      00000240
       EXITS.                                                           00000250
                                                                        00000260
       INITIALIZATION                                                   00000270
02AX   SAVE START-UP CODE                                               00000280
       DISPLAY SYSTEM INITIALIZATION MESSAGE                            00000290
39FM   READ SYSTEM PARAMETER FILE                                       00000300
       IF THERE WHERE NO READ ERRORS THEN                               00000310
39FM      GET ADDRESS OF $SYSCOM PARAMETER TABLE                        00000320
39FM      MOVE SYSTEM PARAMETERS TO $SYSCOM                             00000330
01FM      READ THE BATCH FILE                                           00000340
          IF THERE WERE NO READ ERRORS, THEN                            00000350
01FM         DOUNTIL ALL RECORDS IN BATCH FILE HAVE BEEN TESTED         00000360
01FM            INDEX TO FIRST/NEXT BATCH RECORD                        00000370
01FM            IF BATCH STATUS = ACTIVE, PENDING, OR COMPLETE, THEN    00000380
01FM               SAVE RELATIVE BATCH, TRANSACTION COUNT, AND BATCH ID 00000390
03AX                                                                    00000400
                (ELSE)                                                  00000410
                ENDIF                                                   00000420
            ENDDO                                                       00000430
03AX        DOUNTIL END OF BATCH TABLE REACHED                          00000440
03AX           GET FIRST/NEXT RELATIVE BATCH AND TRANS COUNT FROM TBL   00000450
03AX           IF THIS IS NOT THE END OF THE TABLE, THEN                00000460
04FM              CALCULATE FIRST SECTOR IN TRANS FILE FOR THIS BATCH   00000470
04FM              CALCULATE NUMBER OF SECTORS IN THIS BATCH             00000480
01FM              MOVE NUMBER OF SECTORS IN BATCH TO 'SECTORS LEFT'     00000490
```

```
                    IF 'SECTORS LEFT' <= MAX SECTORS, THEN                  00000500
                       MOVE 'SECTORS LEFT' TO 'SECTOR COUNT'                00000510
                       ZERO 'SECTORS LEFT'                                  00000520
                    ELSE                                                    00000530
                       MOVE MAX SECTORS TO 'SECTOR COUNT'                   00000540
                       SUBTRACT MAX SECTORS FROM 'SECTORS LEFT'             00000550
                       UPDATE TRANSACTION FILE SECTOR POINTER               00000560
                    ENDIF                                                   00000570
04AX                SET FLAG TO FIRST BUFFER                                00000580
04FM                READ TO FIRST BUFFER 'SECTOR COUNT' SECTORS FROM ---    00000590
05AX                --- THE TRANSACTION FILE                                00000600
05AX                SET UP BUFFER POINTER FOR THE FIRST BUFFER              00000610
06AX                ZERO THE ORDER TABLE                                    00000620
07AX                DOUNTIL THE TRANS COUNT = 0, OR BATCH ERROR FLAG ---    00000630
08AX                                                                        00000640
                       --- SET, OR DISK ERROR FLAG SET                      00000650
                       WAIT FOR THE READ TO COMPLETE                        00000660
                       IF NO READ ERRORS, THEN                              00000670
                          IF 'SECTORS LEFT' NOT = 0, THEN                   00000680
                             IF 'SECTORS LEFT' <= MAX SECTORS, THEN         00000690
                                MOVE 'SECTORS LEFT' TO 'SECTOR COUNT'       00000700
                                ZERO 'SECTORS LEFT'                         00000710
                             ELSE                                           00000720
                                MOVE MAX SECTORS TO 'SECTOR COUNT'          00000730
                                SUBTRACT MAX SECTORS FROM 'SECTORS LEFT'    00000740
                                UPDATE TRANSACTION FILE SECTOR POINTER      00000750
                             ENDIF                                          00000760
                             IF FLAG SET TO FIRST BUFFER, THEN              00000770
05AX                            READ TO SECOND BUFFER 'SECTOR COUNT' ---    00000780
                                --- SECTORS FROM THE TRANSACTION FILE       00000790
04AX                            SET FLAG TO SECOND BUFFER                   00000800
                             ELSE                                           00000810
05AX                            READ TO FIRST BUFFER 'SECTOR COUNT' ---     00000820
                                --- SECTORS FROM THE TRANSACTION FILE       00000830
04AX                            SET FLAG TO FIRST BUFFER                    00000840
                             ENDIF                                          00000850
                          (ELSE)                                            00000860
                          ENDIF                                             00000870
05AX                      GET BUFFER POINTER                                00000880
05AX                      DOUNTIL END OF BUFFER OR TRANS COUNT = 0          00000890
04FM                         INDEX TO FIRST/NEXT TRANSACTION RECORD         00000900
04FM                         IF THE RECORD IS USED, THEN                    00000910
04FM                            GET REL. ORDER FROM RECORD                  00000920
04FM                            GET TRANSACTION STATUS FROM RECORD          00000930
06AX                            INDEX INTO ORDER TABLE BY REL. ORDER        00000940
06AX                            INDEX INTO STATUS COUNT FOR THIS ORDER      00000950
06AX                            INCREMENT STATUS COUNT                      00000960
                                DECREMENT TRANSACTION COUNT                 00000970
                             ELSE                                           00000980
                                SET TRANSACTION COUNT TO ZERO               00000990
07AX                            SET BATCH ERROR FLAG                        00001000
                                DISPLAY BATCH ERROR MESSAGE                 00001010
                             ENDIF                                          00001020
                          ENDDO                                             00001030
                          IF TRANSACTION COUNT NOT = 0, THEN                00001040
                             IF FLAG SET TO SECOND BUFFER, THEN             00001050
05AX                            SET POINTER TO SECOND BUFFER                00001060
                             ELSE                                           00001070
05AX                            SET POINTER TO FIRST BUFFER                 00001080
                             ENDIF                                          00001090
                          (ELSE)                                            00001100
                          ENDIF                                             00001110
                       ELSE                                                 00001120
08AX                      SET DISK ERROR FLAG                               00001130
                       ENDIF                                                00001140
                    ENDDO                                                   00001150
07AX                IF BATCH ERROR FLAG AND DISK ERROR FLG NOT SET, THEN    00001160
08AX                                                                        00001170
05FM                   CALCULATE FIRST ORDER SECTOR FOR CURRENT BATCH        00001180
05FM                   READ ORDER SECTORS FOR THIS BATCH                    00001190
                       IF NO READ ERRORS, THEN                              00001200
05FM                      DOUNTIL END OF ORDER RECORDS FOR THIS BATCH        00001210
06AX                                                                        00001220
```

```
05FM                        INDEX TO FIRST/NEXT ORDER RECORD                00001230
06AX                                                                        00001240
05FM                        UPDATE ORDER STATUS COUNTS IN RECORD            00001250
06AX                                                                        00001260
                          ENDDO                                             00001270
05FM                      WRITE ORDER SECTORS FOR THIS BATCH                00001280
                          IF WRITE ERRORS, THEN                             00001290
08AX                        SET DISK ERROR FLAG                             00001300
                          (ELSE)                                            00001310
                          ENDIF                                             00001320
                        ELSE                                                00001330
08AX                      SET DISK ERROR FLAG                               00001340
                        ENDIF                                               00001350
                      ELSE                                                  00001360
07AX                    RESET BATCH ERROR FLAG                              00001370
                      ENDIF                                                 00001380
                    (ELSE)                                                  00001390
                    ENDIF                                                   00001400
08AX                IF DISK ERROR FLAG SET, THEN                            00001410
                      DISPLAY DISK ERROR MESSAGE FOR THIS BATCH             00001420
                      DISPLAY WARNING                                       00001430
08AX                  RESET DISK ERROR FLAG                                 00001440
                    (ELSE)                                                  00001450
                    ENDIF                                                   00001460
                  ENDDO                                                     00001470
                ELSE                                                        00001480
                  DISPLAY DISK ERROR WHILE READING BATCH FILE MESSAGE       00001490
                  DISPLAY WARNING MESSAGE                                   00001500
                ENDIF                                                       00001510
              ELSE                                                          00001520
                DISPLAY UNABLE TO READ SYSTEM PARAMETERS                    00001530
                DISPLAY SYSTEM INITIALIZATION ABORTED                       00001540
              ENDIF                                                         00001550
              IF SYSTEM PARAMETERS READ O.K. THEN                           00001560
                LOAD SYSTEM START-UP PROGRAM                                00001570
              (ELSE)                                                        00001580
              ENDIF                                                         00001590
              PROGSTOP                                                      00001600
                                                                            00000010
                                                                            00000020
                                                                            00000030
                                                                            00000040
                                                                            00000050
              PSC OUTPUT CONTROL TASK                                       00000060
                                                                            00000070
                                                                            00000080
                                                                            00000090
                            - MODULE HISTORY -                              00000100
                                                                            00000110
                                                                            00000120
                 PROJECT:       DAS                  79-01709            00000130
                                                                            00000140
                 SUB-SYSTEM:    PSC COMMUNICATIONS                         00000150
                                                                            00000160
                 MODULE:        PSOC                 49                    00000170
                                                                            00000180
                                                                            00000190
                            - MODULE ABSTRACT -                             00000200
                                                                            00000210
                 THIS TASK PROCESSES ALL PSC OUTPUT MESSAGES. IT REMOVES THE 00000220
              INPUT BUFFER FROM THE PSC OUTPUT QUEUE AND IF THE COMMUNICATIONS00000230
              LINE IS OPEN, FORMATES THE MESSAGE, AND SENDS IT TO THE PSC.  00000240
              IT THEN WAITS ON THE TRANSMISSION COMPLETION CODE TO BE POSTED, 00000250
              AND WILL RETRY THE MESSAGE IF AN ERROR WAS ENCOUNTERED OR WILL 00000260
              CONTINUE COMMUNICATIONS WITH THE NEXT MESSAGE FOUND IN THE    00000270
              PSC OUTPUT QUEUE.                                             00000280
07PS          SET COMMUNICATIONS LINK = CLOSED                              00000290
              SET LAST COMPLETION CODE = IDLE RECEIVED                      00000300
08PS          RESET IDLE MESSAGE COUNT                                      00000310
09PS          CLEAR PSC RESPONSE COUNTS                                     00000320
10PS          RESET RESPONSE DUMP FLAG                                      00000330
13PS          RESET TOTAL NO. OF TIME-OUTS                                  00000340
14PS          RESET INVALID SEQUENCE COUNTER                                00000350
15PS          RESET INVALID CHECKSUM COUNTER                                00000360
```

```
39FM    GET SORT PARAMETERS FROM SYSCOM                                  00000370
        ATTACH THE PSC POLL TASK                                         00000380
        ATTACH PSC RESPONSE PROCESSING TASK                              00000390
        ATTACH PSC INPUT/OUTPUT TASK                                     00000400
        DOUNTIL CPU STOPPED                                              00000410
15SC       WAIT ON SCANNER SLOW DOWN EVENT                               00000420
28PS       WAIT ON PSC RESPONSE OVERFLOW EVENT                           00000430
           IF LAST COMPLETION CODE NOT = DATA RECEIVED OR --             00000440
07PS       -- COMMUNICATION LINK CLOSED THEN                             00000450
02PS          DOUNTIL DATA RECEIVED FROM PSC OUTPUT QUEUE                00000460
02PS             WAIT FOR PSC RUN EVENT TO BE POSTED                     00000470
02PS             RETRIEVE BUFFER FROM PSC OUTPUT QUEUE                   00000480
02PS             IF QUEUE EMPTY THEN                                     00000490
02PS                RESET PSC RUN EVENT                                  00000500
                 (ELSE)                                                  00000510
                 ENDIF                                                   00000520
              ENDDO                                                      00000530
           ELSE                                                          00000540
02PS          RETRIEVE BUFFER FROM PSC OUTPUT QUEUE                      00000550
01PS                                                                     00000560
              IF QUEUE EMPTY THEN                                        00000570
                 SET MESSAGE CODE = IDLE                                 00000580
              ELSE                                                       00000590
01PS             SET MESSAGE CODE = INPUT MESSAGE CODE                   00000600
                 SAVE DATA FROM INPUT BUFFER                             00000610
01PS             RETURN BUFFER TO FREE POOL                              00000620
                 IF FREE POOL FULL THEN                                  00000630
11PS                PANIC!                                               00000640
                 (ELSE)                                                  00000650
                 ENDIF                                                   00000660
              ENDIF                                                      00000670
           ENDIF                                                         00000680
07PS       IF COMMUNICATIONS LINK = CLOSED THEN                          00000690
              IF MESSAGE CODE = OPEN LINK THEN                           00000700
12PS             INITIALIZE THE SEQUENCE BYTES                           00000710
07PS             SET COMMUNICATIONS LINK = OPEN                          00000720
              ELSE                                                       00000730
                 SET MESSAGE CODE = NO MESSAGE                           00000740
              ENDIF                                                      00000750
           ELSE                                                          00000760
              IF MESSAGE CODE = OPEN LINK THEN                           00000770
                 SET MESSAGE CODE = NO MESSAGE                           00000780
              (ELSE)                                                     00000790
              ENDIF                                                      00000800
           ENDIF                                                         00000810
           IF MESSAGE CODE NOT = NO MESSAGE THEN                         00000820
12PS          TOGGLE LAST TRANSMITTED SEQUENCE BYTE                      00000830
16PS          CALL MESSAGE FORMATTING ROUTINE                            00000840
              IF NO ERROR RETURNED THEN                                  00000850
                 SET RETRY COUNT = MAX. NO. OF RETRIES                   00000860
08PS             SET IDLE MESSAGE COUNT = MAX. IDLE COUNT                00000870
                 DOWHILE RETRY COUNT > ZERO                              00000880
17PS                RESET TRANSMISSION COMPLETION EVENT                  00000890
                    ATTACH MESSAGE TIME-OUT TASK                         00000900
29PS                POST PSC INPUT/OUTPUT EVENT                          00000910
17PS                WAIT FOR TRANSMISSION COMPLETION EVENT               00000920
                    IF COMPLETION EVENT CODE = RETRY THEN                00000930
                       SUBTRACT 1 FROM RETRY COUNT                       00000940
                    ELSE                                                 00000950
                       SET RETRY COUNT LESS THAN ZERO                    00000960
                    ENDIF                                                00000970
                 ENDDO                                                   00000980
                 IF RETRY COUNT = 0 THEN                                 00000990
07PS                SET COMMUNICATION LINK = CLOSED                      00001000
02AL                GET ALARM MESSAGE BUFFER                             00001010
                    IF ONE RECEIVED THEN                                 00001020
03AL                   SET UP BUFFER TO PRINT "COMM. FAILURE" MSG.       00001030
01AL                   PLACE BUFFER ADRS. ON ALARM MSG. QUEUE            00001040
                       IF ALARM MSG. QUEUE FULL THEN                     00001050
11PS                      PANIC!                                         00001060
                       (ELSE)                                            00001070
                       ENDIF                                             00001080
                    ELSE                                                 00001090
```

```
11PS                    PANIC!                                          00001100
                      ENDIF                                             00001110
                   ELSE                                                 00001120
12PS                  TOGGLE LAST RECEIVED SEQUENCE BYTE                00001130
                   ENDIF                                                00001140
                ELSE                                                    00001150
11PS               PANIC!                                               00001160
                ENDIF                                                   00001170
             (ELSE)                                                     00001180
             ENDIF                                                      00001190
             IF LINK WAS JUST OPENED THEN                               00001200
02AL            GET ALARM MESSAGE BUFFER                                00001210
                IF ONE RECEIVED THEN                                    00001220
03AL               SET UP BUFFER TO PRINT "COMM. LINK OPEN" MSG.        00001230
01AL               PLACE BUFFER ADDRESS ON ALARM MSG. QUEUE             00001240
                   IF ALARM MSG. QUEUE FULL THEN                        00001250
11PS                  PANIC!                                            00001260
                   (ELSE)                                               00001270
                   ENDIF                                                00001280
                ELSE                                                    00001290
11PS               PANIC!                                               00001300
                ENDIF                                                   00001310
             (ELSE)                                                     00001320
             ENDIF                                                      00001330
          ENDDO                                                         00001340
          PROGSTOP                                                      00001350
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
      PSC RESPONSE PROCESSOR ROUTINE                                    00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
      PROJECT:          DAS                   73-01709                00000130
                                                                        00000140
      SUB-SYSTEM:       PSC COMMUNICATIONS                            00000150
                                                                        00000160
      MODULE:           PSRP                  53                      00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
      THIS TASK PROCESSES ALL VALID RESPONSES FROM THE PSC.             00000220
    IT CALLS THE FILE MANAGER TO UPDATE THE TRANSACTION                 00000230
    STATUS IF NEEDED. IT ALSO CALLS THE MESSAGE PROCESSING              00000240
    ROUTINE IF A MESSAGE IS TO BE PRINTED.                              00000250
    DOUNTIL CPU STOPPED                                                 00000260
24PS   DOUNTIL DATA RECEIVED FROM INPUT QUEUE                           00000270
30PS      WAIT FOR PSC RESPONSE PROCESSING EVENT                        00000280
          REMOVE AN ENTRY FROM THE INPUT QUEUE                          00000290
          IF QUEUE EMPTY THEN                                           00000300
28PS         IF PSC OVERFLOW FLAG SET THEN                              00000310
23PS            GET A PSC RESPONSE BUFFER                               00000320
                IF NONE AVAILABLE THEN                                  00000330
11PS               PANIC !                                              00000340
                ELSE                                                    00000350
22PS               MOVE LAST DATA RECEIVED INTO BUFFER                  00000360
28PS               POST PSC RESPONSE OVERFLOW EVENT                     00000370
                ENDIF                                                   00000380
             ELSE                                                       00000390
30PS            RESET PSC RESPONSE PROCESSING EVENT                     00000400
             ENDIF                                                      00000410
          (ELSE)                                                        00000420
          ENDIF                                                         00000430
       ENDDO                                                            00000440
25PS   IF MSG. TYPE RECEIVED = TRANSFER COMPLETE OR MIS-SORT THEN       00000450
          CLEAR STATUS FLAG                                             00000460
25PS      IF INPUT DATA WITHIN RANGE THEN                               00000470
25PS         IF MSG. TYPE REC. = TRANSFER COMPLETE THEN                 00000480
```

```
                        SET NEW STATUS = STAGED FOR SHIPPMENT                    00000490
25PS                    IF DEST. RECEIVED = REJECT THEN                          00000500
40FM                       ADD 1 TO TOTAL TRANS. SENT TO REJECT                  00000510
                        ELSE                                                     00000520
25PS                       IF DEST. RECEIVED = RECIRCULATION THEN                00000530
40FM                          ADD 1 TO TOTAL TRANS. SENT TO RECIRCULATION        00000540
                           (ELSE)                                                00000550
                           ENDIF                                                 00000560
                        ENDIF                                                    00000570
                     ELSE                                                        00000580
                        SET NEW STATUS = MIS-SORT                                00000590
40FM                    ADD 1 TO NUMBER OF MIS-SORTS AT THIS LINE                00000600
26PS                    CALL ALARM PROCESSING ROUTINE                            00000610
25PS                    IF MIS-SORT DESTINATION = 0 THEN                         00000620
                           SET STATUS CHANGE FLAG                                00000630
                        (ELSE)                                                   00000640
                        ENDIF                                                    00000650
                     ENDIF                                                       00000660
                     IF STATUS CHANGE FLAG CLEAR THEN                            00000670
09FM                    GET A FILE MANAGER CALL BUFFER                           00000680
                        IF ONE RECEIVED THEN                                     00000690
17FM                       SET UP BUFF. FOR A CHANGE STATUS CALL                 00000700
17FM                       SET STATUS = NEW STATUS                               00000710
17FM                       SET DESTINATION = DESTINATION RECEIVED FROM PSC       00000720
08FM                       CALL THE FILE MANAGER                                 00000730
17FM                       SAVE THE COMPLETION CODE                              00000740
09FM                       RETURN THE FILE MANAGER BUFFER TO THE FREE POOL       00000750
                           IF FREE POOL FULL THEN                                00000760
11PS                          PANIC!                                             00000770
                           ELSE                                                  00000780
                              IF COMPLETION CODE NOT SUCCESSFUL THEN             00000790
                                 IF COMP. CODE = TRANS. NOT ON FILE THEN         00000800
02AL                                GET ALARM MESSAGE BUFFER                     00000810
                                    IF ONE RECEIVED THEN                         00000820
03AL                                   SET UP BUFF. FOR "NOT ON FILE MSG"        00000830
01AL                                   PLACE BUFF. ON ALARM MSG. QUEUE           00000840
                                       IF QUEUE FULL THEN                        00000850
11PS                                      PANIC!                                 00000860
                                       (ELSE)                                    00000870
                                       ENDIF                                     00000880
                                    ELSE                                         00000890
11PS                                   PANIC!                                    00000900
                                    ENDIF                                        00000910
                                 ELSE                                            00000920
11PS                                PANIC!                                       00000930
                                 ENDIF                                           00000940
                              (ELSE)                                             00000950
                              ENDIF                                              00000960
                           ENDIF                                                 00000970
                        ELSE                                                     00000980
11PS                       PANIC!                                                00000990
                        ENDIF                                                    00001000
                     (ELSE)                                                      00001010
                     ENDIF                                                       00001020
                  ELSE                                                           00001030
                     IF TRANS. NO. = ZERO AND DESTINATION NOT EQUAL TO --        00001040
                     -- RECIRC OR ERROR LINE THEN                                00001050
26PS                    CALL ALARM PROCESSING ROUTINE                            00001060
                     (ELSE)                                                      00001070
                     ENDIF                                                       00001080
                  ENDIF                                                          00001090
               ELSE                                                              00001100
26PS              CALL ALARM PROCESSING ROUTINE                                  00001110
               ENDIF                                                             00001120
24PS           RETURN BUFFER TO FREE POOL                                        00001130
               IF BUFFER FULL THEN                                               00001140
11PS              PANIC!                                                         00001150
               (ELSE)                                                            00001160
               ENDIF                                                             00001170
            ENDDO                                                                00001180
            EXIT                                                                 00001190
                                                                                 00000010
                                                                                 00000020
```

```
                                                                    00000030
                                                                    00000040
                                                                    00000050
                                                                    00000060
PSC CHECKSUM CALCULATION SUBROUTINE                                 00000070
                                                                    00000080
                                                                    00000090
                                                                    00000100
                - MODULE HISTORY -                                  00000110
                                                                    00000120
                                                                    00000130
   PROJECT:      DAS                 76-01709                       00000140
                                                                    00000150
   SUB-SYSTEM:   PSC COMMUNICTIONS                                  00000160
                                                                    00000170
   MODULE:       PSCKSM              89                             00000180
                                                                    00000190
                                                                    00000200
                - MODULE ABSTRACT -                                 00000210
                                                                    00000220
   THIS SUBROUTINE TAKES EBCDIC CHARACTER INPUT, CONVERTS           00000230
IT TO ASCII AND CALCULATES AN ASCII CHECKSUM CHARACTER.             00000240
THIS CHARACTER IS THEN MADE PRINTABLE AND CONVERTED BACK            00000250
EBCDIC AND PASSED BACK TO THE CALLER.                               00000260
GET INPUTS                                                          00000270
CLEAR CHECKSUM RESULT LOCATION                                      00000280
CLEAR ERROR FLAG                                                    00000290
DO 13 TIMES                                                         00000300
   GET NEXT EBCDIC CHARACTER FROM BUFFER                            00000310
   DOWHILE EBCDIC CHAR NOT EQ TO TABLE VALUE AND NOT END OF TABL    00000320
      BUMP TO NEXT TABLE ENTRY                                      00000330
   ENDDO                                                            00000340
   IF NOT AT END OF TABLE, THEN                                     00000350
      ADD ASCII VALUE TO RESULT                                     00000360
   ELSE                                                             00000370
      SET ERROR FLAG                                                00000380
   ENDIF                                                            00000390
   BUMP TO NEXT BUFFER POSITION                                     00000400
ENDDO                                                               00000410
SET BIT 0 OFF IN RESULT                                             00000420
COMPLEMENT BIT 2 IN RESULT                                          00000430
SET BIT 1 TO RESULT OF COMPLEMENT                                   00000440
DOWHILE ASCII CHARACTER NOT EQ TO TABLE AND NOT END OF TABLE        00000450
   BUMP TO NEXT TABLE ENTRY                                         00000460
ENDDO                                                               00000470
IF NOT END OF TABLE AND ERROR FLAG NOT SET THEN                     00000480
   MOVE EBCDIC VALUE TO CHECK SUM CHARACTER                         00000490
ELSE                                                                00000500
   MOVE A -1 TO CHECK SUM CHARACTER                                 00000510
ENDIF                                                               00000520
RETURN                                                              00000530
```

```
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
PSC MESSAGE TRANSMIT/RECEIVE TASK                                   00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                - MODULE HISTORY -                                  00000100
                                                                    00000110
                                                                    00000120
   PROJECT:      DAS                 76-01709                       00000130
                                                                    00000140
   SUB-SYSTEM:   PSC COMMUNICATIONS                                 00000150
                                                                    00000160
   MODULE:       PSID                51                             00000170
                                                                    00000180
                                                                    00000190
                - MODULE ABSTRACT -                                 00000200
                                                                    00000210
```

```
      THIS TASK TAKES THE FORMATTED MESSAGE AND TRANSMITS IT          00000220
    TO THE PSC THEN STARTS THE PSC RESPONSE TIME-OUT TASK AND         00000230
    WAITS FOR THE RESPONSE FROM THE PSC. IF THE RESPONSE HAS          00000240
    TIMED OUT, THE COMPLETION CODE IS SET TO "RETRY". IF THE          00000250
    RESPONSE SEQUENCE BYTE OR MESSAGE CHECKSUM IS INVALID,            00000260
    THE COMPLETION CODE IS SET TO "RETRY". IF THE RESPONSE IS         00000270
    VALID AND DATA WAS RECEIVED FROM THE PSC, THE COMPLETION          00000280
    CODE IS SET TO "DATA RECEIVED" AND THE DATA IS PASSED TO THE      00000290
    PSC RESPONSE PROCESSING TASK. IF THE RESPONSE IS VALID            00000300
    AND AN IDLE WAS RECEIVED FROM THE PSC, THE COMPLETION CODE        00000310
    IS SET TO "IDLE RECEIVED".                                        00000320
    ENQUEUE THE PSC TERMINAL                                          00000330
    DOUNTIL CPU STOPPED                                               00000340
       RESET ERROR CODE                                               00000350
29PS   WAIT FOR PSC INPUT/OUTPUT EVENT                                00000360
21PS   SET COMPLETION CODE = UNUSED                                   00000370
       GET FORMATTED MESSAGE                                          00000380
       OUTPUT THE MESSAGE TO THE PSC                                  00000390
       READ THE RESPONSE FROM THE PSC                                 00000400
22PS   IF CORRECT NUMBER OF BYTES RECEIVED THEN                       00000410
12PS      IF REC. SEQ. BYTE = TOGGLE OF LAST RECEIVED SEQ. BYTE THEN  00000420
22PS                                                                  00000430
22PS         COMPUTE MESSAGE CHECKSUM                                 00000440
             IF INPUT CHECKSUM = COMPUTED CHECKSUM THEN               00000450
22PS            IF MESSAGE TYPE = IDLE THEN                           00000460
21PS               SET COMPLETION CODE = IDLE RECEIVED                00000470
30PS               POST RESPONSE PROCESSING EVENT                     00000480
                ELSE                                                  00000490
21PS               SET COMPLETION CODE = DATA RECEIVED                00000500
                ENDIF                                                 00000510
             ELSE                                                     00000520
15PS            ADD 1 TO INVALID CHECKSUM COUNTER                     00000530
21PS            SET COMPLETION CODE = RETRY                           00000540
                SET ERROR CODE = INVALID CHECKSUM                     00000550
             ENDIF                                                    00000560
          ELSE                                                        00000570
14PS         ADD 1 TO INVALID SEQUENCE COUNTER                        00000580
21PS         SET COMPLETION CODE = RETRY                              00000590
             SET ERROR CODE = INVALID SEQUENCE                        00000600
          ENDIF                                                       00000610
       ELSE                                                           00000620
PS13      ADD 1 TO TOTAL NUMBER OF TIME-OUTS                          00000630
21PS      SET COMPLETION CODE = RETRY                                 00000640
          SET ERROR CODE = TIME-OUT                                   00000650
       ENDIF                                                          00000660
21PS   IF COMPLETION CODE = DATA RECEIVED THEN                        00000670
23PS      GET PSC RESPONSE PROCESSING BUFFER                          00000680
          IF ONE RECEIVED THEN                                        00000690
22PS         PLACE DATA INPUT INTO RESPONSE PROC. BUFFER               00000700
23PS                                                                  00000710
24PS         PLACE BUFFER ADRS. ON PSC RESPONSE QUEUE                 00000720
             IF QUEUE FULL THEN                                       00000730
11PS            PANIC!                                                00000740
             ELSE                                                     00000750
30PS            POST RESPONSE PROCESSING EVENT                        00000760
          ENDIF                                                       00000770
          ELSE                                                        00000780
28PS         RESET PSC RESPONSE OVERFLOW EVENT                        00000790
          ENDIF                                                       00000800
       ELSE                                                           00000810
          IF ERROR CODE SET THEN                                      00000820
10PS         IF DUMP FLAG SET THEN                                    00000830
02AL            GET ALARM MESSAGE BUFFER                              00000840
                IF ONE RECEIVED THEN                                  00000850
                   PLACE ERROR CODE INTO BUFFER                       00000860
                   COPY LAST PSC MSG. RECEIVED INTO BUFFER            00000870
01AL               PLACE BUFFER ADRS. ON ALARM MSG. QUEUE             00000880
                   IF ALARM MSG. QUEUE FULL THEN                      00000890
11PS                  PANIC!                                          00000900
                   (ELSE)                                             00000910
                   ENDIF                                              00000920
                ELSE                                                  00000930
11PS               PANIC!                                             00000940
```

```
                ENDIF                                                   00000950
              (ELSE)                                                    00000960
              ENDIF                                                     00000970
            (ELSE)                                                      00000980
            ENDIF                                                       00000990
         ENDIF                                                          00001000
29PS     RESET PSC INPUT/OUTPUT EVENT                                   00001010
21PS     POST COMPLETION CODE TO TIME-OUT TASK                          00001020
      ENDDO                                                             00001030
      EXIT                                                              00001040
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
      PSC POLL TASK                                                     00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
         PROJECT:       DAS                 78-01709                  00000130
                                                                        00000140
         SUB-SYSTEM:    PSC COMMUNICATIONS                             00000150
                                                                        00000160
         MODULE:        PSPOL               54                         00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
         THIS TASK DELAYS FOR A PERIOD OF TIME THEN PLACES              00000220
      A PSC IDLE MESSAGE INTO THE PSC OUTPUT QUEUE FOR                  00000230
      TRANSMISSION TO THE PSC.                                          00000240
      DOUNTIL CPU STOPPED                                               00000250
08PS     DOWHILE IDLE MESSAGE COUNT LT. OR EQ. ZERO                     00000260
            DELAY FOR ?? MS.                                            00000270
         ENDDO                                                          00000280
08PS     DOWHILE IDLE MESSAGE COUNT > ZERO                              00000290
            DELAY FOR ?? MS.                                            00000300
08PS        SUBTRACT 1 FROM IDLE MESSAGE COUNT                          00000310
         ENDDO                                                          00000320
01PS     GET PSC OUTPUT BUFFER                                          00000330
         IF ONE RECEIVED THEN                                           00000340
06PS        SET BUFFER FOR IDLE MESSAGE                                 00000350
02PS        PLACE BUFFER ADRS. ON PSC OUTPUT QUEUE                      00000360
02PS        POST PSC RUN EVENT                                          00000370
            IF QUEUE FULL THEN                                          00000380
11PS           PANIC!                                                   00000390
            (ELSE)                                                      00000400
            ENDIF                                                       00000410
         ELSE                                                           00000420
11PS        PANIC!                                                      00000430
         ENDIF                                                          00000440
      ENDDO                                                             00000450
      EXIT                                                              00000460
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
      PSC PANIC PROCESSOR                                               00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
         PROJECT:       DAS                 78-01709                  00000130
                                                                        00000140
         SUB-SYSTEM:    PSC COMMUNICATIONS                             00000150
                                                                        00000160
         MODULE:        PSPAN               55                         00000170
```

```
                                                                    00000180
                                                                    00000190
                        - MODULE ABSTRACT -                         00000200
                                                                    00000210
      THIS SUBROUTINE IS CALLED FROM SEVERAL PSC                    00000220
    MODULES TO SEND A PANIC CODE TO THE ALARM MESSAGE SUB-SYSTEM    00000230
    TO PRINT A FATAL ERROR MESSAGE AND TO SUSPEND EXECUTION OF      00000240
    THE PSC SUB-SYSTEM.                                             00000250
11PS  GET INPUTS                                                    00000260
07PS  SET COMMUNICATIONS LINK = CLOSED                              00000270
01AL  PUT PANIC CODE INPUT INTO ALARM MSG. QUEUE                    00000280
07AL                                                                00000290
08AL                                                                00000300
      DOUNTIL NEW IPL                                               00000310
        WAIT FOR IMPOSSIBLE EVENT                                   00000320
      ENDDO                                                         00000330
      RETURN TO CALLER (NOT REALLY)                                 00000340
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
      PSC ALARM MESSAGE PROCESSING                                  00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                        - MODULE HISTORY -                          00000100
                                                                    00000110
                                                                    00000120
        PROJECT:      DAS                   73-01709                00000130
                                                                    00000140
        SUB-SYSTEM:   PSC COMMUNICATIONS                            00000150
                                                                    00000160
        MODULE:       PSA0                  56                      00000170
                                                                    00000180
                                                                    00000190
                        - MODULE ABSTRACT -                         00000200
                                                                    00000210
      THIS SUBROUTINE PROCESSES ALL PSC RESPONSES OTHER THAN        00000220
    A TRANSFER COMPLETE. IT UPDATES THE APPROPRIATE MESSAGE         00000230
    COUNTER AND PRINTS THE APPROPRIATE MESSAGE AT THE               00000240
    LOG DEVICES.                                                    00000250
26PS  GET INPUTS                                                    00000260
23PS  SET PSC RESPONSE CODE                                         00000270
09PS  GET RESPONSE COUNT TABLE                                      00000280
      DOUNTIL RESPONSE CODE FOUND IN TABLE OR END OF TABLE          00000290
09PS    INDEX TO FIRST/NEXT TABLE ENTRY                             00000300
23PS    IF RESPONSE CODE = TABLE ENTRY THEN                         00000310
09PS                                                                00000320
09PS      ADD 1 TO RESPONSE CODE COUNT                              00000330
        (ELSE)                                                      00000340
        ENDIF                                                       00000350
      ENDDO                                                         00000360
      IF END OF TABLE FOUND THEN                                    00000370
09PS    ADD 1 TO UNKNOWN RESPONSE COUNT                             00000380
      (ELSE)                                                        00000390
      ENDIF                                                         00000400
02AL  GET AN ALARM MESSAGE BUFFER                                   00000410
      IF ONE RECEIVED THEN                                          00000420
26SC    IF DATA RECEIVED WITHIN RANGE THEN                          00000430
25SC                                                                00000440
23PS      CASENTRY (RESPONSE CODE)                                  00000450
            CASE RESPONSE CODE = SORTATION LINE FULL                00000460
03AL          BUILD BUFFER FOR "SORTATION LINE FULL" MSG.           00000470
            CASE RESPONSE CODE = SORTATION TRANSFER FAILURE         00000480
03AL          BUILD BUFFER FOR "SORTATION TRANSFER FAILURE" MSG.     00000490
40FM          ADD 1 TO TRANSFER FAILURES AT THIS LINE               00000500
            CASE RESPONSE CODE = MISSORT                            00000510
26SC          IF SORT DESTINATION = ZERO THEN                       00000520
03AL            BUILD BUFFER FOR "TRANSACTION LOST" MSG.            00000530
              ELSE                                                  00000540
03AL            BUILD BUFFER FOR "MISSORT" MSG.                     00000550
              ENDIF                                                 00000560
```

```
                  CASE RESPONSE CODE = SUCCESSFUL TRANSFER                    00000570
03AL                 BUILD BUFFER FOR "OUT-OF-RANGE" MSG.                     00000580
                  CASE RESPONSE CODE = OTHER                                  00000590
10PS                 IF DUMP INPUT FLAG SET THEN                              00000600
03AL                    BUILD BUFFER FOR PRINTING THE DATA RECEIVED            00000610
                     ELSE                                                     00000620
02AL                    RETURN ALARM MESSAGE BUFFER TO FREE POOL              00000630
                        IF FREE POOL FULL THEN                               00000640
11PS                       PANIC!                                            00000650
                        (ELSE)                                               00000660
                        ENDIF                                                00000670
                     ENDIF                                                   00000680
                  ENDCASE                                                    00000690
               ELSE                                                          00000700
03AL              BUILD BUFFER FOR "OUT-OF-RANGE" MSG.                       00000710
               ENDIF                                                         00000720
               IF ALARM MESSAGE BUFFER NOT RETURNED TO FREE POOL THEN        00000730
01AL              PLACE BUFFER ADRS. ON ALARM MESSAGE QUEUE                  00000740
                  IF ALARM MESSAGE QUEUE FULL THEN                           00000750
11PS                 PANIC!                                                  00000760
                  (ELSE)                                                     00000770
                  ENDIF                                                      00000780
               (ELSE)                                                        00000790
               ENDIF                                                         00000800
            ELSE                                                             00000810
11PS           PANIC!                                                        00000820
            ENDIF                                                            00000830
25PS        RETURN TO CALLER                                                 00000840

                                                                             00000010
                                                                             00000020
                                                                             00000030
                                                                             00000040
                                                                             00000050
      PSC RESPONSE TIME-OUT TASK                                             00000060
                                                                             00000070
                                                                             00000080
                                                                             00000090
                           - MODULE HISTORY -                                00000100
                                                                             00000110
                                                                             00000120
         PROJECT:          DAS                    78-01709                  00000130
                                                                             00000140
         SUB-SYSTEM:       PSC COMMUNICATIONS                               00000150
                                                                             00000160
         MODULE:           PSTO                   52                        00000170
                                                                             00000180
                                                                             00000190
                           - MODULE ABSTRACT -                               00000200
                                                                             00000210
         THIS TASK IS THE PSC RESPONSE TIME-OUT TASK. IF THE PSC             00000220
      HAS NOT RESPONDED WHEN THIS TASK TIMES OUT, THE INPUT                  00000230
      REQUEST OF TASK "PSIO" IS TERMINATED. WHEN A COMPLETION                00000240
      CODE IS SET BY TASK "PSIO", THIS TASK WILL POST THAT CODE              00000250
      BACK TO THE PSC CONTROL TASK "PSOC".                                   00000260
      RESET TIME-OUT COUNT                                                   00000270
21PS  DOWHILE COMPLETION CODE NOT POSTED AND TIME-OUT COUNT NOT ZERO         00000280
         DELAY FOR ?? MS.                                                    00000290
         SUBTRACT 1 FROM THE TIME-OUT COUNT                                  00000300
      ENDDO                                                                  00000310
21PS  IF COMPLETION CODE NOT POSTED THEN                                     00000320
         KILL THE PSC READ FUNCTION                                          00000330
21PS     WAIT FOR COMPLETION CODE TO BE POSTED                               00000340
      (ELSE)                                                                 00000350
      ENDIF                                                                  00000360
21PS  POST COMPLETION CODE TO PSC CONTROL TASK "PSOC"                        00000370
17PS                                                                         00000380
      EXIT                                                                   00000390
```

```
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
PSC OUTPUT MESSAGE FORMATTING                                           00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                    78-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      PSC COMMUNICATIONS                                  00000150
                                                                        00000160
   MODULE:          PSMF                   50                          00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
   THIS SUBROUTINE FORMATES THE PSC OUTPUT MESSAGE INTO                 00000220
A FORM THAT CAN BE TRANSMITTED OVER THE TTY LINE TO THE                 00000230
PSC. IF THE INPUT MESSAGE TYPE IS NOT ONE OF THE VALID                  00000240
TYPES, THIS ROUTINE WILL RETURN TO THE CALLER WITH                      00000250
AN ERROR CODE.                                                          00000260
16PS  GET INPUTS                                                        00000270
16PS  SET RETURN CODE = ERROR                                           00000280
12PS  PLACE THE CURRENT OUTPUT SEQUENCE BYTE IN OUTPUT BUFFER           00000290
16PS  CASENTRY (MESSAGE CODE)                                           00000300
         CASE MESSAGE CODE = IDLE                                       00000310
18PS        BUILD IDLE MESSAGE IN OUTPUT BUFFER                         00000320
16PS        SET RETURN CODE = SUCCESSFUL                                00000330
         CASE MESSAGE CODE = OPEN LINK                                  00000340
19PS        BUILD OPEN LINK MESSAGE IN OUTPUT BUFFER                    00000350
16PS        SET RETURN CODE = SUCCESSFUL                                00000360
         CASE MESSAGE CODE = INDUCT PACKAGE                             00000370
20PS        BUILD INDUCT PACKAGE MESSAGE IN OUTPUT BUFFER               00000380
            IF NO CONVERSION ERRORS FOUND AND DATA IN RANGE THEN        00000390
16PS           SET RETURN CODE = SUCCESSFUL                             00000400
            (ELSE)                                                      00000410
            ENDIF                                                       00000420
      ENDCASE                                                           00000430
      IF RETURN CODE = SUCCESSFUL THEN                                  00000440
27PS     COMPUTE MESSAGE CHECKSUM                                       00000450
18PS     PLACE CHECKSUM IN OUTPUT BUFFER                                00000460
19PS                                                                    00000470
20PS                                                                    00000480
      (ELSE)                                                            00000490
      ENDIF                                                             00000500
16PS  RETURN TO CALLER                                                  00000510
```

```
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
INDUCTION SCANNER #2 TASK                                               00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                    78-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      SCANNER INPUT                                       00000150
                                                                        00000160
   MODULE:          SCSR2                  10                          00000170
                                                                        00000180
                                                                        00000190
```

```
                        - MODULE ABSTRACT -                                     00000200
                                                                                00000210
                                                                                00000220
   THIS TASK RUNS UNTIL THE SCANNER STOP FLAG IS SET (-1). IT                   00000230
FIRST READS A TRANSACTION NUMBER FROM INDUCTION SCANNER #2, THEN                00000240
CONVERTS THE NUMBER TO BINARY. AN ERROR FLAG IS SENT TO THE PACK-               00000250
AGE PROCESSING TASK (SCIND) TO INDICATE THE STATUS OF THE READ.                 00000260
THE SCAN INFORMATION IS PUT ON THE INDUCTION QUEUE AND THE PACK-                00000270
AGE PROCESSING TASK EVENT IS POSTED.                                            00000280
                                                                                00000290
      ENQUEUE THE SCANNER                                                       00000300
01SC  DO UNTIL THE SCANNER STOP FLAG IS SET                                     00000310
04SC     READ THE TRANSACTION NUMBER (READTEXT)                                 00000320
         IF THE BUFFER IS NOT COMPLETELY FULL, THEN                             00000330
20SC        GET AN INDUCTION QUEUE BUFFER                                       00000340
20SC        IF ONE RECEIVED, THEN                                               00000350
06SC           PUT IN SCANNER ID                                                00000360
               CONVERT TRANSACTION NUMBER FROM EBCDIC TO BINARY                 00000370
11SC           IF ANY ERRORS (CONVERSION, < 5-DIGITS), THEN                     00000380
                  SET TRANSACTION NUMBER TO ZERO AND PUT IN BUFFER              00000390
14SC              GET STOP ON NOREAD FLAG FROM SYSCOM                           00000400
11SC              MOVE NO READ FLAG TO ERROR FLAG                               00000410
               ELSE                                                             00000420
                  SET ERROR FLAG = -1                                           00000430
               ENDIF                                                            00000440
11SC           PUT ERROR FLAG IN BUFFER                                         00000450
21SC           PUT ADDRESS OF BUFFER ON INDUCTION QUEUE                         00000460
21SC           IF THERE WAS NO ROOM, THEN                                       00000470
01SC              SET SCANNER STOP FLAG                                         00000480
07AL              PANIC                                                         00000490
               ELSE                                                             00000500
18SC              POST PACKAGE PROCESSING TASK EVENT                            00000510
               ENDIF                                                            00000520
            ELSE                                                                00000530
01SC           SET SCANNER STOP FLAG                                            00000540
07AL           PANIC                                                            00000550
            ENDIF                                                               00000560
         (ELSE)                                                                 00000570
         ENDIF                                                                  00000580
      ENDDO                                                                     00000590
      DEQUEUE THE SCANNER                                                       00000600
      ENDTASK                                                                   00000610
                                                                                00000010
                                                                                00000020
                                                                                00000030
                                                                                00000040
                                                                                00000050
      INDUCTION SCANNER #1 TASK                                                 00000060
                                                                                00000070
                                                                                00000080
                                                                                00000090
                        - MODULE HISTORY -                                      00000100
                                                                                00000110
                                                                                00000120
         PROJECT:       DAS                    76-01709                        00000130
                                                                                00000140
         SUB-SYSTEM:    SCANNER INPUT                                          00000150
                                                                                00000160
                                                                                00000170
                        - MODULE ABSTRACT -                                     00000180
                                                                                00000190
                                                                                00000200
   THIS TASK RUNS UNTIL THE SCANNER STOP FLAG IS SET (-1). IT                   00000210
FIRST READS A TRANSACTION NUMBER FROM INDUCTION SCANNER #1, THEN                00000220
CONVERTS THE NUMBER TO BINARY. AN ERROR FLAG IS SENT TO THE PACK-               00000230
AGE PROCESSING TASK (SCIND) TO INDICATE THE STATUS OF THE READ.                 00000240
THE SCAN INFORMATION IS PUT ON THE INDUCTION QUEUE AND THE PACK-                00000250
AGE PROCESSING TASK EVENT IS POSTED.                                            00000260
```

```
                                                                        00000270
      ENQUEUE THE SCANNER                                               00000280
01SC  DO UNTIL THE SCANNER STOP FLAG IS SET                             00000290
04SC     READ THE TRANSACTION NUMBER (READTEXT)                         00000300
         IF THE BUFFER IS NOT COMPLETELY FULL, THEN                     00000310
20SC        GET AN INDUCTION QUEUE BUFFER                               00000320
20SC        IF ONE RECEIVED, THEN                                       00000330
06SC           PUT IN SCANNER ID                                        00000340
               CONVERT TRANSACTION NUMBER FROM EBCDIC TO BINARY         00000350
11SC           IF ANY ERRORS (CONVERSION, < 6-DIGITS), THEN             00000360
                  SET TRANSACTION NUMBER TO ZERO AND PUT IN BUFFER      00000370
14SC              GET STOP ON NOREAD FLAG FROM SYSCOM                   00000380
11SC              MOVE NO READ FLAG TO ERROR FLAG                       00000390
               ELSE                                                     00000400
                  SET ERROR FLAG = -1                                   00000410
               ENDIF                                                    00000420
11SC           PUT ERROR FLAG IN BUFFER                                 00000430
21SC           PUT ADDRESS OF BUFFER ON INDUCTION QUEUE                 00000440
21SC           IF THERE WAS NO ROOM, THEN                               00000450
01SC              SET SCANNER STOP FLAG                                 00000460
07AL              PANIC                                                 00000470
               ELSE                                                     00000480
18SC              POST PACKAGE PROCESSING TASK EVENT                    00000490
               ENDIF                                                    00000500
            ELSE                                                        00000510
01SC           SET SCANNER STOP FLAG                                    00000520
07AL           PANIC                                                    00000530
            ENDIF                                                       00000540
         (ELSE)                                                         00000550
         ENDIF                                                          00000560
      ENDDO                                                             00000570
      DEQUEUE THE SCANNER                                               00000580
      ENDTASK                                                           00000590


                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
      SCANNER INITIALIZATION PROGRAM                                    00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
         PROJECT:       DAS                 78-01709                 00000130
                                                                        00000140
         SUB-SYSTEM:    SCANNER INPUT                                 00000150
                                                                        00000160
         MODULE:        SCINIT              5                         00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
         THIS PROGRAM WILL BE LOADED INTO MEMORY BY THE INI-            00000230
      TIAL SYSTEM PROGRAM.  ONCE LOADED IT WILL SET THE SCAN-           00000240
      NER STOP FLAG TO NOSTOP.  AFTER ATTACHING ALL OF THE              00000250
      SCANNER ROUTINES, IT WILL GO INTO A LOOP AND POST THE             00000260
      PACKAGE PROCESSING TASK EVENT AND THE STATUS UPDATE TASK          00000270
      EVENT ONCE EVERY 5 SECONDS.  IF THE SCANNER STOP FLAG IS          00000280
      EVER SET, IT WILL DROP OUT OF THE LOOP AND WAIT FOR AN            00000290
      EVENT THAT WILL NEVER BE POSTED JUST TO KEEP IT IN MEMORY.        00000300
                                                                        00000310
02SC  INITIALIZATION                                                    00000320
01SC  INITIALIZE SCANNER STOP FLAG TO NOSTOP (0)                        00000330
39FM  GET NUMBER OF SCANNERS AND TYPES FROM SYSCOM                      00000340
      ATTACH THOSE TASKS                                                00000350
      ATTACH SCTSU                                                      00000360
      ATTACH SCINDT                                                     00000370
      DO UNTIL THE SCANNER STOP FLAG IS SET                             00000380
         DELAY 5 SECONDS                                                00000390
18SC     POST THE PACKAGE PROCESSING TASK EVENT                         00000400
```

```
19SC        POST THE STATUS UPDATE TASK EVENT                          00000410
          ENDDO                                                        00000420
10SC      WAIT FOR PROGSTOP EVENT                                      00000430
          PROGRAM STOP                                                 00000440


                                                                       00000010
-------------------------------------------------------------------    00000020
                                                                       00000030
                                                                       00000040
          TRANSACTION STATUS UPDATE TASK                               00000050
                                                                       00000060
                                                                       00000070
-------------------------------------------------------------------    00000080
                          - MODULE HISTORY -                           00000090
                                                                       00000100
-------------------------------------------------------------------    00000110
             PROJECT:     DAS                  73-01709               00000120
                                                                       00000130
             SUB-SYSTEM:  SCANNER INPUT                                00000140
                                                                       00000150
             MODULE:      SCTSU                12                     00000160
-------------------------------------------------------------------    00000170
                                                                       00000180
                          - MODULE ABSTRACT -                          00000190
-------------------------------------------------------------------    00000200
                                                                       00000210
          THIS TASK RUNS UNTIL THE SCANNER STOP FLAG IS SET.  IT       00000220
     FIRST GETS THE ADDRESS OF THE BUFFER OFF THE STATUS UPDATE        00000230
     QUEUE, THEN GETS THE STATUS AND TRANSACTION NUMBER OUT OF THE     00000240
     BUFFER.  IT THEN ENQUEUES AND CALLS THE FILE MANAGER TO CHNGE     00000250
     THE STATUS.  ON RETURN FROM THE FILE MANAGER IT IS DEQUEUED AND   00000260
     THE RETURN CODE IS CHECKED.  ANY CODE OTHER THAN SUCCESSFUL OR    00000270
     OR TRANSACTION NOT ON FILE IS CONSIDERED A FATAL ERROR.  IF THE   00000280
     TRANSACTION WAS NOT FOUND AN ALARM MESSAGE IS PRINTED.  IF THE    00000290
     ERROR IS CONSIDERED FATAL, A PANIC IS EXECUTED AND THE SCANNER    00000300
     STOP FLAG IS SET TO BRING DOWN THE SUB-SYSTEM.  IN ANY OF THE     00000310
     THREE CASES, THE STATUS UPDATE BUFFER IS RETRNED TO THE QUEUE.    00000320
     IF THE SCANNER STOP FLAG IS NOT YET SET, THE QUEUE IS CHECKED     00000330
     FOR ANOTHER ENTRY.  IF THERE IS ONE, IT IS PROCESSED.  IF NOT,    00000340
     A WAIT FOR STATUS UPDATE TASK EVENT IS PERFORMED.                 00000350
                                                                       00000360
02SC    INITIALIZATION                                                 00000370
01SC    DOUNTIL THE SCANNER STOP FLAG IS SET                           00000380
           DOUNTIL AN ENTRY RECIEVED FROM THE UPDATE QUEUE             00000390
19SC          WAIT FOR STATUS UPDATE TASK EVENT                        00000400
09SC          GET A BUFFER ADDRESS OFF THE STATUS UPDATE QUEUE         00000410
09SC          IF NONE RECIEVED, THEN                                   00000420
15SC             GET THE SLOW DOWN FLAG                                00000430
15SC             IF THE SLOW DOWN FLAG IS SET, THEN                    00000440
16SC                GET A BUFFER ADDRESS OFF THE RESERVE QUEUE         00000450
16SC                IF NONE RECEIVED, THEN                             00000460
15SC                   CLEAR THE SLOW DOWN FLAG                        00000470
                    (ELSE)                                             00000480
                 ENDIF                                                 00000490
              (ELSE)                                                   00000500
              ENDIF                                                    00000510
           (ELSE)                                                      00000520
           ENDIF                                                       00000530
05SC       IF A BUFFER ADDRESS WAS RECEIVED, THEN                      00000540
17SC                                                                   00000550
              GET THE DATA FROM THE BUFFER                             00000560
05SC          RETURN THE BUFFER TO THE BUFFER QUEUE                    00000570
17SC                                                                   00000580
              IF THERE IS NO ROOM, THEN                                00000590
08AL             PANIC                                                 00000600
              (ELSE)                                                   00000610
              ENDIF                                                    00000620
           ELSE                                                        00000630
19SC          RESET STATUS UPDATE TASK EVENT                           00000640
           ENDIF                                                       00000650
        ENDDO                                                          00000660
        SET ERROR CODE = GOOD                                          00000670
```

```
09FM        GET A FILE MANAGER INPUT BUFFER                                 00000680
09FM        IF ONE RECIEVED THEN                                            00000690
10FM           PUT IN CHANGE STATUS CALL CODE                               00000700
17FM           PUT IN DATA                                                  00000710
04FM                                                                        00000720
08FM        CALL THE FILE MANAGER                                           00000730
17FM           GET THE RETURN CODE                                          00000740
11FM           IF RETURN CODE NOT = SUCCESSFUL THEN                         00000750
11FM              IF RETURN CODE = RECORD NOT FOUND THEN                    00000760
02AL                 GET AN ALARM MESSAGE BUFFER                           00000770
02AL                 IF ONE RECIEVED THEN                                  00000780
03AL                    PUT IN MESSAGE #                                   00000790
                        PUT IN TRANSACTION #                               00000800
                        PUT IN SCANNER ID                                  00000810
01AL                    PUT ADDRESS OF BUFFER ON ALARM QUEUE               00000820
01AL                    IF ALREADY FULL THEN                               00000830
08AL                       SET ERROR CODE = FATAL,ALARM QUE FULL           00000840
                        (ELSE)                                             00000850
                        ENDIF                                              00000860
                    ELSE                                                   00000870
08AL                   SET ERROR CODE = FATAL, ALARM BUFQUE EMPTY          00000880
                    ENDIF                                                  00000890
                 ELSE                                                      00000900
08AL                SET ERROR CODE = FATAL, BAD RTURN FROM FM              00000910
                 ENDIF                                                     00000920
              (ELSE)                                                       00000930
              ENDIF                                                        00000940
08FM          RETURN FILE MANAGER BUFFER                                   00000950
08FM          IF NO ROOM THEN                                              00000960
08AL             SET ERROR CODE = FATAL,FM BUFFER QUE FULL                 00000970
              (ELSE)                                                       00000980
              ENDIF                                                        00000990
           ELSE                                                            00001000
08AL          SET ERROR CODE = FATAL, NO FILE MANAGER BUFFER               00001010
           ENDIF                                                           00001020
05SC       RETURN STATUS UPDATE BUFFER                                     00001030
05SC       IF THERE IS NO ROOM THEN                                        00001040
08AL          SET ERROR CODE = FATAL, STATUS UPDATE BUFQUE FULL            00001050
           (ELSE)                                                          00001060
           ENDIF                                                           00001070
           IF ERROR CODE = FATAL THEN                                      00001080
08AL          USE ERROR CODE FOR PANIC CODE                                00001090
01SC          SET SCANNER STOP FLAG                                        00001100
07AL          PANIC                                                        00001110
        (ELSE)                                                             00001120
        ENDIF                                                              00001130
     ENDDO                                                                 00001140
     ENDTASK                                                               00001150

                                                                           00000010
                                                                           00000020
                                                                           00000030
     INDUCTION SCANNER TASK                                                00000040
                                                                           00000050
                                                                           00000060
                                                                           00000070
                  - MODULE HISTORY -                                       00000080
                                                                           00000090
                                                                           00000100
     PROJECT:     DAS                    73-01709                       00000110
                                                                           00000120
     SUB-SYSTEM:  SCANNER INPUT                                            00000130
                                                                           00000140
     MODULE:      SCIND                  13                             00000150
                                                                           00000160
                                                                           00000170
                  - MODULE ABSTRACT -                                      00000180
                                                                           00000190
                                                                           00000200
   THIS TASK IS ATTACHED BY THE SCANNER INITIALIZATION PROGRAM.            00000210
 IT THEN WAITS ON THE PACKAGE PROCESSING EVENT TO BE POSTED.               00000220
 WHEN POSTED IT WILL GET AN ENTRY FROM THE INDUCTION QUEUE AND             00000230
 GET THE DATA FROM THE BUFFER. THE DATA WILL THEN BE PROCESSED.            00000240
```

```
                                                                    00000250
02SC    INITIALIZATION                                              00000260
01SC    DO UNTIL THE SCANNER STOP FLAG IS SET                       00000270
           RESET FLAGS (STATJS,PSC)                                 00000280
           SET MSG# = 3                                             00000290
           SET PANIC CODE = 0                                       00000300
09SC       SET DESTINATION = RECIRCULATION                          00000310
21SC       DO UNTIL ELEMENT RECEIVED FROM INDUCTION QUEUE           00000320
18SC          WAIT FOR PACKAGE PROCESSING TASK EVENT                00000330
21SC          GET AN ELEMENT FROM THE QUEUE                         00000340
21SC          IF NONE RECEIVED, THEN                                00000350
18SC             RESET EVENT                                        00000360
              (ELSE)                                                00000370
              ENDIF                                                 00000380
           ENDDO                                                    00000390
20SC       GET THE DATA FROM THE BUFFER                             00000400
20SC       PUT THE BUFFER ON THE BUFFER QUEUE                       00000410
20SC       IF THERE WAS ROOM, THEN                                  00000420
07PS          GET THE PSC COMMUNICATIONS FLAG                       00000430
07PS          IF THE LINK IS OPEN, THEN                             00000440
15SC             IF THE SLOW DOWN FLAG IS SET, THEN                 00000450
16SC                SET UP TO USE THE ALTERNATE QUEUES              00000460
17SC                                                                00000470
                 ELSE                                               00000480
05SC                SET UP TO USE THE PRIMARY QUEUES                00000490
08SC                                                                00000500
                 ENDIF                                              00000510
12SC             GET THE ERROR FLAG                                 00000520
12SC             IF THE ERROR FLAG = -1 THEN                        00000530
13SC                IF TRANSACTION# NOT = 0 THEN                    00000540
23SC                                                                00000550
03FM                   COMPUTE RELATIVE RECORD NUMBER               00000560
03FM                   READ RECORD FROM TRANSACTION DIRECTORY       00000570
                       IF THERE WERE NO READ ERRORS THEN            00000580
                          SET PSC FLAG                              00000590
03FM                      COMPUTE INDEX TO BUFFER                   00000600
03FM                      GET RELATIVE TRANSACTION NUMBER           00000610
23SC                      IF ON FILE THEN                           00000620
03FM                         GET RELATIVE BATCH AND DESTINATION     00000630
02FM                         GET BATCH STATUS FROM RESIDENT BATCH TABLE 00000640
23SC                                                                00000650
02FM                         IF STATUS = PENDING, ACTIVE, OR CMPLTE THEN 00000660
04FM                            COMPUTE RELTIVE RCRD FOR THE TRANS FILE 00000670
04FM                            READ RECORD FROM TRANSACTION FILE   00000680
                                IF THERE WERE NO ERRORS, THEN       00000690
04FM                               COMPUTE INDEX TO BUFFER          00000700
04FM                               GET THE TRANSACTION NUMBER       00000710
23SC                               IF THIS TRANS# EQ OUR TRANS#, THEN 00000720
                                      SET STATUS FLAG               00000730
23SC                                  IF BATCH STATUS = ACTIVE THEN 00000740
04FM                                     PICK UP DESTINATION        00000750
                                      ELSE                          00000760
03AL                                     SET MSG# = TRANS OUT OF BATCH 00000770
23SC                                     IF THE BATCH IS COMPLETE, THEN 00000780
13SC                                        SET DESTINATION = ERROR 00000790
                                         (ELSE)                     00000800
                                         ENDIF                      00000810
                                      ENDIF                         00000820
                                   ELSE                             00000830
03AL                                  SET MSG# = TRANS NOT ON FILE  00000840
13SC                                  SET DESTINATION = ERROR       00000850
                                   ENDIF                            00000860
                                ELSE                                00000870
                                   SET PANIC CODE = READ ERROR      00000880
                                ENDIF                               00000890
                             ELSE                                   00000900
03AL                            SET MSG# = TRANS NOT ON FILE        00000910
13SC                            SET DESTINATION = ERROR             00000920
                             ENDIF                                  00000930
                          ELSE                                      00000940
03AL                         SET MSG# = TRANS NOT ON FILE           00000950
13SC                         SET DESTINATION = ERROR                00000960
                          ENDIF                                     00000970
                       ELSE                                         00000980
```

```
08AL                    SET PANIC CODE = READ ERROR                    00000990
                        ENDIF                                          00001000
                     ELSE                                              00001010
                        SET PSC FLAG                                   00001020
13SC                    GET ERROR DESTINATION                          00001030
13SC                    SET DESTINATION = ERROR                        00001040
                     ENDIF                                             00001050
                  ELSE                                                 00001060
40FM                 INCREMENT NOREAD COUNT FOR THIS SCANNER IN SYSCOM  00001070
03AL                 SET MSG# = SCANNER READ ERROR                     00001080
12SC                 IF ERROR FLAG = 0 THEN                            00001090
                        SET PSC FLAG                                   00001100
                     (ELSE)                                            00001110
                     ENDIF                                             00001120
                  ENDIF                                                00001130
               (ELSE)                                                  00001140
               ENDIF                                                   00001150
            ELSE                                                       00001160
08AL           SET PANIC CODE = INDUCTION BUFFER QUEUE FULL             00001170
            ENDIF                                                      00001180
            IF PSC FLAG SET THEN                                       00001190
01PS           GET A BUFFER FROM THE PSC BUFFER QUEUE                  00001200
01PS           IF ONE RECIEVED THEN                                    00001210
03PS              PUT IN CODE FOR INDUCT PACKAGE                       00001220
04PS              PUT IN TRANSACTION NUMBER                            00001230
04PS              PUT IN SCANNER ID                                    00001240
                  IF THE DESTINATION IS NOT REGIRC OR THE ERR QUEUE, THEN00001250
06FM                 GET DEST. FROM SRT TRNSLTN TBLE                   00001260
06FM                 IF TRNSLTN TBLE DEST = 0, THEN                    00001270
06FM                    DEST = DIRECTORY DESTINATION                   00001280
                     ELSE                                              00001290
06FM                    DEST = TRNSLTION TBLE DEST                     00001300
                     ENDIF                                             00001310
                  ELSE                                                 00001320
                     IF THE DESTINATION IS REGIRC, THEN                00001330
40FM                    INCREMENT THE REGIRC CNT IN SYSCOM             00001340
                     ENDIF                                             00001350
                  ENDIF                                                00001360
04PS              PUT IN DESTINATION                                   00001370
02PS              PUT ADDRESS OF BUFFER ON PSC OUTPUT QUEUE            00001380
02PS              IF THE PSC OUTPUT QUEUE IS FULL THEN                 00001390
08AL                 SET PANIC CODE = PSC OUTPUT QUEUE FULL            00001400
                  ELSE                                                 00001410
02PS                 GET ADDRESS OF PSC EVENT                          00001420
02PS                 POST PSC EVENT                                    00001430
                  ENDIF                                                00001440
               ELSE                                                    00001450
08AL              SET PANIC CODE = PSC BUFFER QUEUE EMPTY               00001460
               ENDIF                                                   00001470
            (ELSE)                                                     00001480
            ENDIF                                                      00001490
            IF MESSAGE NUMBER NOT EQUAL TO ZERO THEN                   00001500
02AL           GET AN ALARM MESSAGE OUTPUT BUFFER                      00001510
02AL           IF ONE RECIEVED THEN                                    00001520
03AL              PUT IN MESSAGE#, SCANNER ID, AND TRANS #             00001530
01AL              PUT ADDRESS OF BUFFER ON ALARM MESSAGE QUEUE         00001540
01AL              IF THE ALARM QUEUE IS FULL THEN                      00001550
08AL                 SET PANIC CODE = ALARM QUEUE FULL                 00001560
                  (ELSE)                                               00001570
                  ENDIF                                                00001580
               ELSE                                                    00001590
08AL              SET PANIC CODE = ALARM BUFFER QUEUE EMPTY             00001600
               ENDIF                                                   00001610
            (ELSE)                                                     00001620
            ENDIF                                                      00001630
            IF STATUS FLAG SET THEN                                    00001640
               ENQUEUE THE QUEUE RESOURCE                              00001650
               DOUNTIL A BUFFER IS RECIEVED FROM THE BUFFER QUEUE      00001660
05SC              GET A BUFFER                                         00001670
17SC                                                                   00001680
05SC              IF ONE RECIEVED THEN                                 00001690
04FM                 PUT IN STATUS, SCANNER ID, TRANS#, AND DIRCTRY ENTRY00001700
08SC                 PUT ADDRESS OF BUFFER ON STATUS UPDATE QUEUE      00001710
08SC                 IF THE QUEUE IS ALREADY FULL THEN                 00001720
```

```
08AL                    SET PANIC CODE = STATUS UPDATE QUEUE FULL        00001730
                      ELSE                                               00001740
19SC                     POST STATUS UPDATE TASK EVENT                   00001750
                      ENDIF                                              00001760
                    ELSE                                                 00001770
15SC                   SET THE SLOW DOWN FLAG                            00001780
15SC                   IF IT IS NOT SET, THEN                            00001790
16SC                     SET UP TO USE ALTERNATE QUEUES                  00001800
17SC                                                                     00001810
15SC                   SET THE SLOW DOWN FLAG                            00001820
                      ELSE                                               00001830
08AL                     PANIC                                           00001840
                      ENDIF                                              00001850
                    ENDIF                                                00001860
                  ENDDO                                                  00001870
                DEQUEUE THE QUEUE RESOURCE                               00001880
              (ELSE)                                                     00001890
              ENDIF                                                      00001900
              IF PANIC CODE NOT EQUAL TO ZERO THEN                       00001910
08AL            GET PANIC CODE                                           00001920
01SC            SET SCANNER STOP FLAG                                    00001930
07AL            PANIC                                                    00001940
              (ELSE)                                                     00001950
              ENDIF                                                      00001960
            ENDDO                                                        00001970
          ENDTASK                                                        00001980


                                                                         00000010
                                                                         00000020
                                                                         00000030
                                                                         00000040
                                                                         00000050
          STOCK OUT SCANNER TASK                                         00000060
                                                                         00000070
                                                                         00000080
                                                                         00000090
                          - MODULE HISTORY -                             00000100
                                                                         00000110
                                                                         00000120
             PROJECT:       DAS                    79-01709             00000130
                                                                         00000140
             SUB-SYSTEM:    SCANNER INPUT                               00000150
                                                                         00000160
             MODULE:        SCSTO                  6                    00000170
                                                                         00000180
                                                                         00000190
                          - MODULE ABSTRACT -                            00000200
                                                                         00000210
                                                                         00000220
          THIS TASK RUNS UNTIL THE SCANNER STOP FLAG IS SET (-1).  IT    00000230
        FIRST READS A TRANSACTION NUMBER FROM THE STOCK OUT SCANNER,     00000240
        THEN CONVERTS THE NUMBER TO BINARY.  IF THERE WERE ANY ERRORS AN 00000250
        ALARM MESSAGE IS PRINTED TO TELL THE OPERATOR.  OTHERWISE A      00000260
        STATUS UPDATE BUFFER IS OBTAINED.  IN IT ARE PUT THE NEW STATUS  00000270
        AND TRANSACTION NUMBER.  THE ADDRESS OF THE BUFFER IS THEN PUT   00000280
        ON THE STATUS UPDATE QUEUE AND THE STATUS UPDATE TASK EVENT IS   00000290
        POSTED.  THEN THE PROCESS STARTS ALL OVER WAITING FOR ANOTHER READ. 00000300
                                                                         00000310
          ENQUEUE THE STOCK OUT SCANNER                                  00000320
02SC      INITIALIZATION                                                 00000330
01SC      DOUNTIL SCANNER STOP FLAG SET                                  00000340
            RESET THE FATAL FLAG                                         00000350
04SC        READ THE TRANSACTION NUMBER (READTEXT)                       00000360
            IF WE RECEIVED 7 BYTES (6-CHAR AND LINEFEED)                 00000370
              CONVERT FROM EBCDIC TO BINARY                              00000380
            (ELSE)                                                       00000390
            ENDIF                                                        00000400
04SC        IF NO ERRORS (CONVERSION,=0,<6 DIGITS) THEN                  00000410
05SC          GET A BUFFER FROM THE STATUS UPDATE BUFFER QUEUE            00000420
05SC          IF ONE RECIEVED THEN                                       00000430
06SC            PUT IN SCANNER ID                                        00000440
                PUT IN TRANSACTION NUMBER                                00000450
04FM            PUT IN STATUS                                            00000460
```

```
08SC            PUT ADDRESS OF BUFFER ON STATUS UPDATE QUEUE                00000470
08SC            IF STATUS UPDATE QUEUE ALREADY FULL THEN                   00000480
                   SET FATAL FLAG                                          00000490
08AL               SET PANIC CODE = STATUS UPDATE QUEUE FULL                00000500
                ELSE                                                       00000510
19SC               POST THE STATUS UPDATE TASK EVENT                       00000520
                ENDIF                                                      00000530
             ELSE                                                          00000540
                SET FATAL FLAG                                             00000550
08AL            SET PANIC CODE = STATUS UPDT BUFQUE EMPTY                   00000560
             ENDIF                                                         00000570
          ELSE                                                             00000580
             IF THE READ BUFFER WAS NOT FULL, THEN                         00000590
02AL            GET A BUFFER FROM THE ALARM MESSAGE BUFFER QUEUE            00000600
02AL            IF ONE RECIEVED THEN                                       00000610
03AL               PUT IN MESSAGE NUMBER = NOREAD                          00000620
05SC               PUT IN SCANNER ID                                       00000630
01AL               PUT ADDRESS OF BUFFER ON ALARM QUEUE                    00000640
01AL               IF THERE IS NO ROOM THEN                                00000650
                      SET FATAL FLAG                                       00000660
08AL                  SET PANIC CODE = ALRM QUE FULL                       00000670
                   (ELSE)                                                  00000680
                   ENDIF                                                   00000690
                ELSE                                                       00000700
                   SET FATAL FLAG                                          00000710
08AL               SET PANIC CODE = ALRM BUFQUE EMPTY                       00000720
                ENDIF                                                      00000730
             (ELSE)                                                        00000740
             ENDIF                                                         00000750
          ENDIF                                                            00000760
          IF THE FATAL FLAG IS SET THEN                                    00000770
01SC         SET SCANNER STOP FLAG                                         00000780
08AL         GET PANIC CODE                                                00000790
07AL         PANIC                                                         00000800
          (ELSE)                                                           00000810
          ENDIF                                                            00000820
       ENDDO                                                               00000830
       DEQUEUE THE SCANNER                                                 00000840
       ENDTASK                                                             00000850

                                                                           00000010
                                                                           00000020
                                                                           00000030
                                                                           00000040
                                                                           00000050
       EXCESS REPACK SCANNER TASK                                          00000060
                                                                           00000070
                                                                           00000080
                                                                           00000090
                          - MODULE HISTORY -                               00000100
                                                                           00000110
                                                                           00000120
          PROJECT:        DAS                  73-01709                  00000130
                                                                           00000140
          SUB-SYSTEM:     SCANNER INPUT                                    00000150
                                                                           00000160
          MODULE:         SCXRP                8                           00000170
                                                                           00000180
                                                                           00000190
                          - MODULE ABSTRACT -                              00000200
                                                                           00000210
                                                                           00000220
       THIS TASK RUNS UNTIL THE SCANNER STOP FLAG IS SET (-1). IT          00000230
    FIRST READS A TRANSACTION NUMBER FROM THE EXCESS REPACK SCANNER,       00000240
    THEN CONVERTS THE NUMBER TO BINARY. IF THERE WERE ANY ERRORS AN        00000250
    ALARM MESSAGE IS PRINTED TO TELL THE OPERATOR. OTHERWISE A             00000260
    STATUS UPDATE BUFFER IS OBTAINED. IN IT ARE PUT THE NEW STATUS         00000270
    AND TRANSACTION NUMBER. THE ADDRESS OF THE BUFFER IS THEN PUT          00000280
    ON THE STATUS UPDATE QUEUE AND THE STATUS UPDATE TASK EVENT IS         00000290
    POSTED. THEN THE PROCESS STARTS ALL OVER WAITING FOR ANOTHER READ.     00000300
```

```
                                                                    00000310
      ENQUEUE THE EXCESS REPACK SCANNER                             00000320
02SC  INITIALIZATION                                                00000330
01SC  DOUNTIL SCANNER STOP FLAG SET                                 00000340
        RESET THE FATAL FLAG                                        00000350
04SC    READ THE TRANSACTION NUMBER (READTEXT)                      00000360
        IF WE RECEIVED 7 BYTES (6-CHAR AND LINEFEED)                00000370
          CONVERT FROM EBCDIC TO BINARY                             00000380
        (ELSE)                                                      00000390
        ENDIF                                                       00000400
04SC    IF NO ERRORS (CONVERSION,=0,=6 DIGITS) THEN                 00000410
05SC      GET A BUFFER FROM THE STATUS UPDATE BUFFER QUEUE          00000420
05SC      IF ONE RECIEVED THEN                                      00000430
06SC        PUT IN SCANNER ID                                       00000440
            PUT IN TRANSACTION NUMBER                               00000450
04FM        PUT IN STATUS                                           00000460
08SC        PUT ADDRESS OF BUFFER ON STATUS UPDATE QUEUE            00000470
08SC        IF STATUS UPDATE QUEUE ALREADY FULL THEN                00000480
              SET FATAL FLAG                                        00000490
08AL          SET PANIC CODE = STATUS UPDATE QUEUE FULL             00000500
            ELSE                                                    00000510
19SC          POST THE STATUS UPDATE TASK EVENT                     00000520
            ENDIF                                                   00000530
          ELSE                                                      00000540
            SET FATAL FLAG                                          00000550
08AL        SET PANIC CODE = STATUS UPDT BUFQUE EMPTY               00000560
        ENDIF                                                       00000570
      ELSE                                                          00000580
        IF THE READ BUFFER WAS NOT FULL, THEN                       00000590
02AL      GET A BUFFER FROM THE ALARM MESSAGE BUFFER QUEUE          00000600
02AL      IF ONE RECIEVED THEN                                      00000610
03AL        PUT IN MESSAGE NUMBER = NOREAD                          00000620
06SC        PUT IN SCANNER ID                                       00000630
01AL        PUT ADDRESS OF BUFFER ON ALARM QUEUE                    00000640
01AL        IF THERE IS NO ROOM THEN                                00000650
              SET FATAL FLAG                                        00000660
08AL          SET PANIC CODE = ALRM QUE FULL                        00000670
            (ELSE)                                                  00000680
            ENDIF                                                   00000690
          ELSE                                                      00000700
            SET FATAL FLAG                                          00000710
08AL        SET PANIC CODE = ALRM BUFQUE EMPTY                      00000720
          ENDIF                                                     00000730
        (ELSE)                                                      00000740
        ENDIF                                                       00000750
      ENDIF                                                         00000760
      IF THE FATAL FLAG IS SET THEN                                 00000770
01SC    SET SCANNER STOP FLAG                                       00000780
08AL    GET PANIC CODE                                              00000790
07AL    PANIC                                                       00000800
      (ELSE)                                                        00000810
      ENDIF                                                         00000820
    ENDDO                                                           00000830
    DEQUEUE THE SCANNER                                             00000840
    ENDTASK                                                         00000850


                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
    SHIPMENT SCANNER TASK                                           00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
    PROJECT:        DAS                 76-01709                    00000130
                                                                    00000140
    SUB-SYSTEM:     SCANNER INPUT                                   00000150
                                                                    00000160
    MODULE:         SCSHP               7                           00000170
                                                                    00000180
                                                                    00000190
```

```
                        - MODULE ABSTRACT -                                  00000200
                                                                             00000210
                                                                             00000220
   THIS TASK RUNS UNTIL THE SCANNER STOP FLAG IS SET (-1).  IT               00000230
FIRST READS A TRANSACTION NUMBER FROM THE SHIPMENT SCANNER,                  00000240
THEN CONVERTS THE NUMBER TO BINARY.  IF THERE WERE ANY ERRORS AN             00000250
ALARM MESSAGE IS PRINTED TO TELL THE OPERATOR.  OTHERWISE A                  00000260
STATUS UPDATE BUFFER IS OBTAINED.  IN IT ARE PUT THE NEW STATUS              00000270
AND TRANSACTION NUMBER.  THE ADDRESS OF THE BUFFER IS THEN PUT               00000280
ON THE STATUS UPDATE QUEUE AND THE STATUS UPDATE TASK EVENT IS               00000290
POSTED.  THEN THE PROCESS STARTS ALL OVER WAITING FOR ANOTHER READ.          00000300
                                                                             00000310
      ENQUEUE THE SHIPMENT SCANNER                                           00000320
02SC  INITIALIZATION                                                         00000330
01SC  DOUNTIL SCANNER STOP FLAG SET                                          00000340
        RESET THE FATAL FLAG                                                 00000350
04SC    READ THE TRANSACTION NUMBER (READTEXT)                               00000360
        IF WE RECEIVED 7 BYTES (6-CHAR AND LINEFEED)                         00000370
          CONVERT FROM EBCDIC TO BINARY                                      00000380
        (ELSE)                                                               00000390
        ENDIF                                                                00000400
04SC    IF NO ERRORS (CONVERSION,=0,<6 DIGITS) THEN                          00000410
05SC      GET A BUFFER FROM THE STATUS UPDATE BUFFER QUEUE                    00000420
05SC      IF ONE RECIEVED THEN                                               00000430
06SC        PUT IN SCANNER ID                                                00000440
            PUT IN TRANSACTION NUMBER                                        00000450
04FM        PUT IN STATUS                                                    00000460
08SC        PUT ADDRESS OF BUFFER ON STATUS UPDATE QUEUE                     00000470
08SC        IF STATUS UPDATE QUEUE ALREADY FULL THEN                         00000480
              SET FATAL FLAG                                                 00000490
08AL          SET PANIC CODE = STATUS UPDATE QUEUE FULL                      00000500
            ELSE                                                             00000510
19SC          POST THE STATUS UPDATE TASK EVENT                              00000520
            ENDIF                                                            00000530
          ELSE                                                               00000540
            SET FATAL FLAG                                                   00000550
08AL        SET PANIC CODE = STATUS UPDT BUFQUE EMPTY                        00000560
          ENDIF                                                              00000570
        ELSE                                                                 00000580
          IF THE READ BUFFER WAS NOT FULL, THEN                              00000590
02AL        GET A BUFFER FROM THE ALARM MESSAGE BUFFER QUEUE                 00000600
02AL        IF ONE RECIEVED THEN                                             00000610
03AL          PUT IN MESSAGE NUMBER = NOREAD                                 00000620
06SC          PUT IN SCANNER ID                                              00000630
01AL          PUT ADDRESS OF BUFFER ON ALARM QUEUE                           00000640
01AL          IF THERE IS NO ROOM THEN                                       00000650
                SET FATAL FLAG                                               00000660
08AL            SET PANIC CODE = ALRM QUE FULL                               00000670
              (ELSE)                                                         00000680
              ENDIF                                                          00000690
            ELSE                                                             00000700
              SET FATAL FLAG                                                 00000710
08AL          SET PANIC CODE = ALRM BUFQUE EMPTY                             00000720
            ENDIF                                                            00000730
          (ELSE)                                                             00000740
          ENDIF                                                              00000750
        ENDIF                                                                00000760
        IF THE FATAL FLAG IS SET THEN                                        00000770
01SC      SET SCANNER STOP FLAG                                              00000780
08AL      GET PANIC CODE                                                     00000790
07AL      PANIC                                                              00000800
        (ELSE)                                                               00000810
        ENDIF                                                                00000820
      ENDDO                                                                  00000830
      DEQUEUE THE SCANNER                                                    00000840
      ENDTASK                                                                00000850

                                                                             00000010
                                                                             00000020
                                                                             00000030
                                                                             00000040
                                                                             00000050
```

```
INDUCTION SCANNER #3 TASK                                                00000060
                                                                         00000070
                                                                         00000080
                                                                         00000090
                        - MODULE HISTORY -                               00000100
                                                                         00000110
                                                                         00000120
    PROJECT:        DAS                     79-01709                     00000130
                                                                         00000140
    SUB-SYSTEM:     SCANNER INPUT                                        00000150
                                                                         00000160
                                                                         00000170
                        - MODULE ABSTRACT -                              00000180
                                                                         00000190
                                                                         00000200
    THIS TASK RUNS UNTIL THE SCANNER STOP FLAG IS SET (-1). IT           00000210
FIRST READS A TRANSACTION NUMBER FROM INDUCTION SCANNER #3, THEN         00000220
CONVERTS THE NUMBER TO BINARY. AN ERROR FLAG IS SENT TO THE PACK-        00000230
AGE PROCESSING TASK (SCIND) TO INDICATE THE STATUS OF THE READ.          00000240
THE SCAN INFORMATION IS PUT ON THE INDUCTION QUEUE AND THE PACK-         00000250
AGE PROCESSING TASK EVENT IS POSTED.                                     00000260
                                                                         00000270
    ENQUEUE THE SCANNER                                                  00000280
01SC    DO UNTIL THE SCANNER STOP FLAG IS SET                            00000290
04SC        READ THE TRANSACTION NUMBER (READTEXT)                       00000300
            IF THE BUFFER IS NOT COMPLETELY FULL, THEN                   00000310
20SC            GET AN INDUCTION QUEUE BUFFER                            00000320
20SC            IF ONE RECEIVED, THEN                                    00000330
06SC                PUT IN SCANNER ID                                    00000340
                    CONVERT TRANSACTION NUMBER FROM EBCDIC TO BINARY     00000350
11SC                IF ANY ERRORS (CONVERSION, < 6-DIGITS), THEN         00000360
                        SET TRANSACTION NUMBER TO ZERO AND PUT IN BUFFER 00000370
14SC                    GET STOP ON NOREAD FLAG FROM SYSCOM              00000380
11SC                    MOVE NO READ FLAG TO ERROR FLAG                  00000390
                    ELSE                                                 00000400
                        SET ERROR FLAG = -1                              00000410
                    ENDIF                                                00000420
11SC                PUT ERROR FLAG IN BUFFER                             00000430
21SC                PUT ADDRESS OF BUFFER ON INDUCTION QUEUE             00000440
21SC                IF THERE WAS NO ROOM, THEN                           00000450
01SC                    SET SCANNER STOP FLAG                            00000460
07AL                    PANIC                                            00000470
                    ELSE                                                 00000480
18SC                    POST PACKAGE PROCESSING TASK EVENT               00000490
                    ENDIF                                                00000500
                ELSE                                                     00000510
01SC                SET SCANNER STOP FLAG                                00000520
07AL                PANIC                                                00000530
            ENDIF                                                        00000540
        (ELSE)                                                           00000550
        ENDIF                                                            00000560
    ENDDO                                                                00000570
    DEQUEUE THE SCANNER                                                  00000580
    ENDTASK                                                              00000590

                                                                         00000010
                                                                         00000020
                                                                         00000030
                                                                         00000040
                                                                         00000050
MIS-SORT REPORT                                                          00000060
                                                                         00000070
                                                                         00000080
                                                                         00000090
                        - MODULE HISTORY -                               00000100
                                                                         00000110
                                                                         00000120
    PROJECT:        DAS                     79-01709                     00000130
                                                                         00000140
    SUB-SYSTEM:     OPERATOR INTERFACE                                   00000150
                                                                         00000160
    MODULE:         DIMSR                   94                           00000170
                                                                         00000180
                                                                         00000190
```

```
                        - MODULE ABSTRACT -                                     00000200
                                                                                00000210
                                                                                00000220
   THIS MODULE REQUESTS A BATCH ID. THE FILE MANAGER IS                         00000230
CALLED TO READ THE BATCH FILE INFORMATION.                                      00000240
   IF THE BATCH INFORMATION IS FOUND, THE USER IS ASKED IF                      00000250
ALL ORDERS ARE TO BE LISTED. IF ALL ORDERS ARE NOT TO BE                        00000260
LISTED, A SPECIFIC ORDER ID IS REQUESTED, AND THE FILE                          00000270
MANAGER IS CALLED TO READ THE ORDER FILE INFORMATION.                           00000280
   IF HARDCOPY IS REQUIRED, THEN THE TERMINAL IS DEQUEUED                       00000290
DEQUEUED AND THE SYSTEM PRINTER IS ENQUEUED. THE BATCH ID,                      00000300
HEADER AND STATUS ARE THEN OUTPUT TO THE ENQUEUED DEVICE.                       00000310
   IF ALL ORDERS TO BE LISTED, THE FILE MANAGER IS CALLED TO                    00000320
SEQUENTIALLY READ EACH ORDER RECORD. FOR EACH ORDER RECORD,                     00000330
OR THE SPECIFICLY REQUESTED ORDER RECORD, THE ORDER ID, ALL                     00000340
QUANTITIES AND ORDER PERCENT COMPLETE ARE OUTPUT.                               00000350
   THE FILE MANAGER IS CALLED TO SEQUENTIALLY READ EACH TRAN-                   00000360
SACTION BLOCK. EACH TRANSACTION RECORD ASSOCIATED WITH THE                      00000370
ORDER IS EXAMINED. IF THE TRANSACTION STATUS IS A MIS-SORT,                     00000380
THE TRANSACTION INFORMATION IS OUTPUT TO THE ENQUEUED DEVICE.                   00000390
THE TERMINAL/PRINTER IS DEQUEUED BEFORE THE PROGRAM ENDS.                       00000400


                                                                                00000410
                                                                                00000420
      ENQUEUE USER TERMINAL                                                     00000430
040I  DISPLAY ON USER TERMINAL "MIS-SORT REPORT"                                00000440
030I  ASK USER FOR BATCH ID                                                     00000450
120I                                                                            00000460
480I  DEQUEUE AN IPM BUFFER (GET.IPM)                                           00000470
490I  DEQUEUE RECORD BUFFER (GET.SRB)                                           00000480
600I  INITIALIZE PANIC CODE                                                     00000490
160I  ACCESS BATCH FILE (GET.BATCH)                                             00000500
100I  IF BATCH RECORD FOUND (OK SET), THEN                                      00000510
030I  !  ASK USER IF ALL ORDERS TO BE LISTED                                    00000520
600I  !  IF NOT ALL ORDERS, THEN                                                00000530
030I  !     ASK USER FOR ORDER ID                                               00000540
130I  !                                                                         00000550
600I  !     INITIALIZE PANIC CODE                                               00000560
180I  !     ACCESS ORDER FILE (GET.ORDER)                                       00000570
100I  !     IF ORDER RECORD FOUND (OK SET), THEN                                00000580
100I  !        SET SPECIFIC ORDER                                               00000590
      !     (ELSE)                                                            00000600
      !     ENDIF                                                             00000610
      !  ELSE                                                                 00000620
100I  !     SET OK                                                              00000630
100I  !     CLEAR SPECIFIC ORDER                                                00000640
      !  ENDIF                                                                00000650
100I  !  IF ORDER FOUND (OK SET), THEN                                          00000660
100I  !  !  SET ORDER SEQUENCE                                                  00000670
100I  !  !  SET EXCEPTIONS                                                      00000680
030I  !  !  ASK USER IF HARDCOPY OUTPUT IS REQUIRED                             00000690
      !  !  IF HARDCOPY REQUIRED, THEN                                          00000700
100I  !  !     SET HARDCOPY                                                     00000710
      !  !     DEQUEUE USER TERMINAL                                            00000720
      !  !     ENQUEUE PRINTER                                                  00000730
      !  !  ELSE                                                                00000740
100I  !  !     CLEAR HARDCOPY                                                   00000750
      !  !  ENDIF                                                               00000760
600I  !  !  INITIALIZE DATE, TIME, PAGE COUNT AND LINE COUNT                     00000770
610I  !  !  PRINT/DISPLAY BATCH HEADINGS                                        00000780
610I  !  !  PRINT/DISPLAY BATCH INFORMATION                                     00000790
610I  !  !  PRINT/DISPLAY ORDER HEADINGS                                        00000800
600I  !  !  INITIALIZE PANIC CODE                                               00000810
130I  !  !  INITIALIZE ORDER COUNT                                              00000820
130I  !  !  DOUNTIL ORDER COUNT GREATER THAN MAXIMUM                            00000830
100I  !  !  !  IF SPECIFIC ORDER SET, THEN                                     00000840
130I  !  !  !     UPDATE ORDER COUNT BEYOND MAXIMUM                            00000850
120I  !  !  !                                                                  00000860
      !  !  !  ELSE                                                            00000870
170I  !  !  !     FIND THE FIRST/NEXT ORDER IN THIS BATCH (FIND.ORDER)         00000880
      !  !  !  ENDIF                                                           00000890
```

```
10JI    !  !  !  IF ORDER RECORD FOUND (OK SET), THEN                    00000900
13JI    !  !  !  !  COMPUTE ORDER PERCENT COMPLETE                       00000910
        !  !  !  !  SET LIST ORDER INFORMATION                           00000920
61JI    !  !  !  !  PRINT/DISPLAY ORDER INFORMATION AND PERCENT COMPLETE 00000930
        !  !  !  !  --- (FORMS.CTL) ---                                  00000940
10JI    !  !  !  !  IF ABORT SET THEN                                    00000950
13JI    !  !  !  !     SET ORDER COUNT GREATER THAN MAX.                 00000960
        !  !  !  !  ELSE                                                 00000970
        !  !  !  !  !  IF MIS-SORTS IN THIS ORDER THEN                  00000980
        !  !  !  !  !  !  CLEAR LIST ORDER INFORMATION                  00000990


61JI    !  !  !  !  !  !  PRINT/DISPLAY TRANSACTION HEADINGS.           00001000
60JI    !  !  !  !  !  !                                                 00001010
60AL    !  !  !  !  !  !  INITIALIZE PANIC CODE                         00001020
60JI    !  !  !  !  !  !  INITIALIZE REPORT TRANSACTION COUNT           00001030
14JI    !  !  !  !  !  !  INITIALIZE TRANS. CURRENT BLOCK NUMBER        00001040
14JI    !  !  !  !  !  !  INITIALIZE TRANSACTION COUNT                  00001050
14JI    !  !  !  !  !  !  DOUNTIL TRANSACTION COUNT > MAXIMUM --        00001060
        !  !  !  !  !  !  -- OR BATCH NO LONGER ON FILE                 00001070
19JI    !  !  !  !  !  !  !  FIND THE FIRST/NEXT MIS-SORT RECORD        00001080
        !  !  !  !  !  !  !  --- (FIND.TRANS) ---                       00001090
        !  !  !  !  !  !  !  IF BATCH STILL ON FILE THEN                00001100
10JI    !  !  !  !  !  !  !  !  IF EXCEPTION RECORD FOUND THEN          00001110
        !  !  !  !  !  !  !  !  !  IF TRANS. STATUS = MIS-SORT THEN     00001120
20FM    !  !  !  !  !  !  !  !  !  !  SET UP TO READ THE TRANS. DIR.    00001130
08FM    !  !  !  !  !  !  !  !  !  !  CALL FILE MANAGER FOR DIR. REC.   00001140
20FM    !  !  !  !  !  !  !  !  !  !  IF RETURN CODE = ERROR THEN       00001150
20FM    !  !  !  !  !  !  !  !  !  !  !  IF RETURN CODE = NOT ON FILE  00001160
        !  !  !  !  !  !  !  !  !  !  !  !  PRINT ABORT MSG.           00001170
        !  !  !  !  !  !  !  !  !  !  !  ELSE                          00001180
54JI    !  !  !  !  !  !  !  !  !  !  !  !  PANIC                     00001190
        !  !  !  !  !  !  !  !  !  !  !  ENDIF                        00001200
        !  !  !  !  !  !  !  !  !  !  ELSE                           00001210
03FM    !  !  !  !  !  !  !  !  !  !  !  GET MIS-SORT DEST. FROM REC. 00001220
        !  !  !  !  !  !  !  !  !  !  !  IF MIS-SORT DEST. = -1 THEN  00001230
        !  !  !  !  !  !  !  !  !  !  !  !  SET OUTPUT = '?'          00001240
        !  !  !  !  !  !  !  !  !  !  !  ELSE                         00001250
        !  !  !  !  !  !  !  !  !  !  !  !  PLACE DEST. IN OUTPUT     00001260
        !  !  !  !  !  !  !  !  !  !  !  ENDIF                        00001270
        !  !  !  !  !  !  !  !  !  !  ENDIF                           00001280
        !  !  !  !  !  !  !  !  !  ELSE                              00001290
        !  !  !  !  !  !  !  !  !  !  SET OUTPUT TO BLANKS            00001300
        !  !  !  !  !  !  !  !  !  ENDIF                             00001310
        !  !  !  !  !  !  !  !  !  IF BATCH STILL ON FILE THEN        00001320
        !  !  !  !  !  !  !  !  !  !  SET LIST TRANSACTION INFO.      00001330
61JI    !  !  !  !  !  !  !  !  !  !  PRINT/DISPLAY TRANS INFORMATION 00001340
        !  !  !  !  !  !  !  !  !  !  --- (FORMS.CTL) ---             00001350
        !  !  !  !  !  !  !  !  !  (ELSE)                            00001360
        !  !  !  !  !  !  !  !  !  ENDIF                             00001370
10JI    !  !  !  !  !  !  !  !  !  IF ABORT SET THEN                  00001380
14JI    !  !  !  !  !  !  !  !  !  !  SET TRANS. COUNT > MAX         00001390
13JI    !  !  !  !  !  !  !  !  !  !  SET ORDER COUNT > MAX.         00001400
        !  !  !  !  !  !  !  !  !  (ELSE)                            00001410
        !  !  !  !  !  !  !  !  !  ENDIF                             00001420
        !  !  !  !  !  !  !  !  (ELSE)                               00001430
        !  !  !  !  !  !  !  !  ENDIF                                00001440
        !  !  !  !  !  !  !  ELSE                                    00001450
04JI    !  !  !  !  !  !  !  !  OUTPUT "BATCH NO LONGER ON FILE      00001460
        !  !  !  !  !  !  !  !  --- REPORT TERMINATED" ---           00001470
10JI    !  !  !  !  !  !  !  !  SET ABORT                            00001480
        !  !  !  !  !  !  !  ENDIF                                   00001490
        !  !  !  !  !  !  ENDDO                                      00001500
        !  !  !  !  !  !  CLEAR LIST TRANSACTION INFORMATION         00001510
        !  !  !  !  !  ELSE                                         00001520
02JI    !  !  !  !  !     PRINT/DISPLAY "NO MIS-SORTS IN THIS ORDER" 00001530
        !  !  !  !  !  ENDIF                                        00001540
        !  !  !  !  !  IF HARD COPY SET THEN                        00001550
        !  !  !  !  !     UPDATE LINE COUNT > MAX.                  00001560
        !  !  !  !  !  ELSE                                         00001570
        !  !  !  !  !     MOVE REPORT TO TOP OF SCREEN              00001580
        !  !  !  !  !  ENDIF                                        00001590
```

```
        !  !  !  !  ENDIF                                                    00001600
        !  !  !  ELSE                                                       00001610
04DI    !  !  !  !  PRINT/DISPLAY "BATCH NO LONGER ON FILE - RPT TERM"     00001620
10DI    !  !  !  !  SET ABORT                                              00001630
        !  !  !  !  IF HARD COPY NOT SET THEN                              00001640
        !  !  !  !     MOVE REPORT TO TOP OF SCREEN                        00001650
        !  !  !  !  (ELSE)                                                 00001660
        !  !  !  !  ENDIF                                                  00001670
        !  !  !  ENDIF                                                     00001680
        !  !  ENDDO                                                        00001690
        !  !  IF HARD COPY AND EXCEPTIONS NOT SET THEN                     00001700
        !  !     MOVE REPORT TO TOP OF SCREEN                              00001710
        !  !  ELSE                                                         00001720
        !  !  ENDIF                                                        00001730
        !  (ELSE)                                                          00001740
        !  ENDIF                                                           00001750
        ELSE                                                              00001760
02DI       PRINT/DISPLAY BATCH ID, "BATCH NOT ON FILE"                    00001770
        ENDIF                                                             00001780
51DI    RELEASE RECORD BUFFER (FREE.SR3)                                  00001790
50DI    RELEASE THE IPM BUFFER (FREE.IPM)                                 00001800
        DEQUEUE TERMINAL/PRINTER                                          00001810
        ENDPROG                                                           00001820
                                                                          00001830
                                                                          00001840
              FORMS.CTL SUBROUTINE                                        00001850
                                                                          00001860
10DI    IF HARDCOPY SET, THEN                                             00001870
60DI       IF LINE COUNT GREATER THAN PRINTER PAGE, THEN                  00001880
10DI          SET OK                                                      00001890
           ELSE                                                           00001900
10DI          CLEAR OK                                                    00001910
           ENDIF                                                          00001920
        ELSE                                                              00001930
60DI       IF LINE COUNT GREATER THAN TERMINAL PAGE, THEN                 00001940
10DI          SET OK                                                      00001950
           ELSE                                                           00001960
10DI          CLEAR OK                                                    00001970
           ENDIF                                                          00001980
        ENDIF                                                             00001990
10DI    IF PAGE OVERFLOW (OK SET), THEN                                   00002000
60DI       INCREMENT PAGE COUNT                                           00002010
61DI       PRINT/DISPLAY BATCH HEADINGS                                   00002020
61DI       PRINT/DISPLAY BATCH INFORMATION                                00002030
61DI       PRINT/DISPLAY ORDER HEADINGS                                   00002040
60DI       INITIALIZE LINE COUNT                                          00002050
10DI       IF LIST TRANSACTION INFORMATION SET, THEN                      00002060
61DI          PRINT/DISPLAY ORDER INFORMATION                             00002070
61DI          PRINT/DISPLAY TRANSACTION HEADINGS                          00002080
60DI          INCREMENT LINE COUNT                                        00002090
           (ELSE)                                                         00002100
           ENDIF                                                          00002110
        (ELSE)                                                            00002120
        ENDIF                                                             00002130
10DI    IF LIST ORDER INFORMATION SET, THEN                               00002140
61DI       PRINT/DISPLAY ORDER INFORMATION                                00002150
60DI       INCREMENT LINE COUNT                                           00002160
        (ELSE)                                                            00002170
        ENDIF                                                             00002180
10DI    IF LIST TRANSACTION INFORMATION SET, THEN                         00002190

61DI       PRINT/DISPLAY TRANSACTION INFORMATION                          00002200
60DI       INCREMENT REPORT TRANSACTION COUNT                             00002210
60DI       INCREMENT LINE COUNT                                           00002220
        (ELSE)                                                            00002230
        ENDIF                                                             00002240
        IF REPORT CANCELLED AND HARD COPY SET THEN                        00002250
           PRINT REPORT CANCELLED MESSAGE                                 00002260
        (ELSE)                                                            00002270
        ENDIF                                                             00002280
        RETURN                                                            00002290
```

```
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
      SYSTEM ERROR COUNTS REPORT                                        00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
         PROJECT:       DAS                     78-01709             00000130
                                                                        00000140
         SUB-SYSTEM:    OPERATOR INTERFACE                            00000150
                                                                        00000160
         MODULE:        DISER                   93                   00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
         THIS MODULE DISPLAYS ALL OF THE SYSTEM ERROR COUNTS            00000230
      EITHER ON THE TERMINAL OR ON THE SYSTEM PRINTER.                  00000240
                                                                        00000250
                                                                        00000260
                                                                        00000270
      INITIALIZATION                                                    00000280
04JI  DISPLAY 'SYSTEM ERROR REPORT'                                     00000290
03JI  ASK USER IF HARDCOPY IS REQUIRED                                  00000300
      IF YES, THEN                                                      00000310
59JI     SET A HARDCOPY FLAG                                            00000320
         ENQUEUE THE SYSTEM PRINTER                                     00000330
         EJECT TO TOP OF FORM                                           00000340
      ELSE                                                              00000350
59JI     CLEAR THE HARDCOPY FLAG                                        00000360
         ENQUEUE THE TERMINAL                                           00000370
         CLEAR THE SCREEN                                               00000380
      ENDIF                                                             00000390
58JI  PUT TIME AND DATE IN THE HEADER                                   00000400
58JI  INITIALIZE PAGE COUNT AND PUT IN THE HEADER                       00000410
58JI  PRINT/DISPLAY HEADER #1                                           00000420
58JI  PRINT/DISPLAY HEADER #2 AND HEADER #3                             00000430
58JI  PRINT/DISPLAY LOST, REJECTED, AND RECIRC COUNTS                   00000440
39FM  GET NUMBER OF INDUCTS FROM SYSCOM                                 00000450
58JI  CALCULATE CENTERING FOR SCANNER HEADERS AND COUNTS                00000460
58JI  PRINT/DISPLAY HEADERS AND COUNTS                                  00000470
59JI  IF HARDCOPY FLAG IS NOT SET, THEN                                 00000480
         MOVE THE REPORT TO THE TOP OF THE SCREEN                       00000490
03JI     ASK USER 'LIST MISSORT COUNTS ?'                               00000500
         IF THE ANSWER IS YES, THEN                                     00000510
            CLEAR THE SCREEN                                            00000520
            INITIALIZE THE LINE COUNT                                   00000530
58JI        INCREMENT THE PAGE NUMBER AND PUT IN THE HEADER             00000540
58JI        PRINT THE HEADER #1                                         00000550
59JI        CLEAR ABORT FLAG                                            00000560
         ELSE                                                           00000570
59JI        SET ABORT FLAG                                              00000580
         ENDIF                                                          00000590
      ELSE                                                              00000600
59JI     CLEAR THE ABORT FLAG                                           00000610
         SKIP A FEW LINES                                               00000620
      ENDIF                                                             00000630
59JI  IF ABORT FLAG IS NOT SET, THEN                                    00000640
04JI     PRINT/DISPLAY 'M I S - S O R T S'                              00000650
39FM     GET NUMBER OF SORT LINES FROM SYSCOM                           00000660
         CALL ROUTINE TO PRINT MIS-SORT COUNTS                          00000670
      (ELSE)                                                            00000680
      ENDIF                                                             00000690
59JI  IF HARDCOPY FLAG NOT SET, THEN                                    00000700
         MOVE THE REPORT TO THE TOP OF SCREEN                           00000710
03JI     ASK USER 'LIST TRANSFER FAILURE COUNTS ?'                      00000720
         IF THE ANSWER IF YES, THEN                                     00000730
            CLEAR THE SCREEN                                            00000740
```

```
                INITIALIZE THE LINE COUNT                                00000750
583I            INCREMENT THE PAGE NUMBER AND PUT IN HEADER              00000760
583I            PRINT THE HEADER #1                                      00000770
593I            CLEAR THE ABORT FLAG                                     00000780
             ELSE                                                        00000790
593I            SET THE ABORT FLAG                                       00000800
             ENDIF                                                       00000810
          ELSE                                                           00000820
593I         CLEAR THE ABORT FLAG                                        00000830
             SKIP A FEW LINES                                            00000840
          ENDIF                                                          00000850
593I      IF THE ABORT FLAG IS NOT SET, THEN                             00000860
043I         PRINT/DISPLAY 'T R A N F E R  F A I L U R E S'              00000870
39FM         GET NUMBER OF SORT LINES FROM SYSCOM                        00000880
             CALL THE SUBROUTINE TO OUTPUT THE TRANS. FAILURE REPORT     00000890
          (ELSE)                                                         00000900
          ENDIF                                                          00000910
593I      IF HARDCOPY FLAG IS SET, THEN                                  00000920
             EJECT TO TOP OF FORM                                        00000930
          (ELSE)                                                         00000940
          ENDIF                                                          00000950
          DEQUEUE WHATEVER WAS ENQUEUED                                  00000960
          EXIT                                                           00000970
                                                                         00000980
          SUBROUTINE TO OUTPUT MIS-SORT OR TRANSFER FAILURE REPORT       00000990
                                                                         00001000
583I      CALCULATE THE NUMBER OF COLUMNS NEEDED                         00001010
583I      PRINT/DISPLAY THAT MANY 'LINE  COUNT'                          00001020
40FM      PRINT THE COUNTS                                               00001030
          ADJUST THE LINE COUNT                                          00001040
593I      IF THE HARDCOPY FLAG IS NOT SET, THEN                          00001050
             MOVE TO TOP OF SCREEN                                       00001060
          (ELSE)                                                         00001070
          ENDIF                                                          00001080
          RETURN TO CALLER                                               00001090
```

```
                                                                         00000010
                                                                         00000020
                                                                         00000030
                                                                         00000040
                                                                         00000050
          RESET ERROR COUNTS                                             00000060
                                                                         00000070
                                                                         00000080
                                                                         00000090
                        - MODULE HISTORY -                               00000100
                                                                         00000110
                                                                         00000120
             PROJECT:       DAS                  73-01709              00000130
                                                                         00000140
             SUB-SYSTEM:    OPERATOR INTERFACE                          00000150
                                                                         00000160
             MODULE:        DIRSE                92                     00000170
                                                                         00000180
                                                                         00000190
                        - MODULE ABSTRACT -                              00000200
                                                                         00000210
                                                                         00000220
             THIS MODULE RESETS THE SYSTEM ERROR COUNTS TO ZERO.         00000230
          ALL ERROR COUNTS CAN BE RESET OR ANY ONE OF THE FIVE           00000240
          CAN BE RESET INDIVIDUALLY.                                     00000250
                                                                         00000260
                                                                         00000270
                                                                         00000280
          INITIALIZATION                                                 00000290
043I      DISPLAY 'RESET ERROR COUNTS COMMAND'                           00000300
033I      ASK USER IF ALL COUNTS SHOULD BE CLEARED                       00000310
          IF NOT, THEN                                                   00000320
033I         ASK USER IF MISSORTS SHOULD BE CLEARED                      00000330
             IF YES, THEN                                                00000340
40FM            CLEAR MISSORT COUNTS                                     00000350
             (ELSE)                                                      00000360
             ENDIF                                                       00000370
```

```
030I        ASK USER IF THE NO-READ COUNTS SHOULD BE CLEARED                00000380
            IF YES, THEN                                                    00000390
40FM           CLEAR NO-READ COUNTS                                         00000400
            (ELSE)                                                          00000410
            ENDIF                                                           00000420
030I        ASK USER IF THE REJECT COUNT SHOULD BE CLEARED                  00000430
            IF YES, THEN                                                    00000440
40FM           CLEAR REJECT COUNT                                           00000450
            (ELSE)                                                          00000460
            ENDIF                                                           00000470
030I        ASK USER IF THE RECIRC COUNT SHOULD BE CLEARED                  00000480
            IF YES, THEN                                                    00000490
40FM           CLEAR RECIRC COUNT                                           00000500
            (ELSE)                                                          00000510
            ENDIF                                                           00000520
030I        ASK USER IF LOST FROM TRACKING COUNT SHOULD BE CLEARED          00000530
            IF YES, THEN                                                    00000540
40FM           CLEAR LOST FROM TRACKING COUNT                               00000550
            (ELSE)                                                          00000560
            ENDIF                                                           00000570
030I        ASK USER IF TRANSFER FAILURE COUNTS SHOULD BE CLEARED           00000580
            IF YES, THEN                                                    00000590
40FM           CLEAR TRANSFER FAILURE COUNTS                                00000600
            (ELSE)                                                          00000610
            ENDIF                                                           00000620
         ELSE                                                               00000630
40FM        CLEAR ALL COUNTS                                                00000640
         ENDIF                                                              00000650
040I     DISPLAY 'COUNTS CLEARED'                                           00000660
         EXIT                                                               00000670


                                                                            00000010
                                                                            00000020
                                                                            00000030
                                                                            00000040
                                                                            00000050
         WRITE BATCH (TO DISKETTE) COMMAND                                  00000060
                                                                            00000070
                                                                            00000080
                                                                            00000090
                        - MODULE HISTORY -                                  00000100
                                                                            00000110
                                                                            00000120
            PROJECT:        DAS                 76-01709                    00000130
                                                                            00000140
            SUB-SYSTEM:     OPERATOR INTERFACE                              00000150
                                                                            00000160
            MODULE:         DIWRB               21                          00000170
                                                                            00000180
                                                                            00000190
                        - MODULE ABSTRACT -                                 00000200
                                                                            00000210
                                                                            00000220
            THIS MODULE REQUESTS A BATCH ID. THE FILE MANAGER IS CALLED     00000230
         TO READ THE BATCH FILE RECORD. IF FOUND, THE BATCH INFORMATION     00000240
         IS DISPLAYED FOR USER VERIFICATION. THE USER IS ASKED IF OK TO     00000250
         WRITE TO DISKETTE. IF YES, THE USER IS ASKED IF DISKETTE IS        00000260
         MOUNTED. IF YES, THE FILE MANAGER IS CALLED TO SEQUENTIALLY READ   00000270
         ALL ORDER FILE RECORDS TO BUILD A TABLE OF ORDER ID'S.             00000280
            NEXT THE FILE MANAGER IS CALLED TO SEQUENTIALLY READ ALL        00000290
         TRANSACTION RECORDS, THEN EACH IS WRITTEN TO THE DISKETTE.         00000300
         ENQUEUE USER TERMINAL                                              00000310
         CLEAR THE SCREEN                                                   00000320
040I     DISPLAY ON USER TERMINAL "WRITE BATCH COMMAND"                     00000330
480I     DEQUEUE AN IPM BUFFER (GET.IPM)                                    00000340
         DEQUEUE A RECORD BUFFER (GET.SRB)                                  00000350
030I     ASK USER FOR BATCH ID                                              00000360
08AL     INITIALIZE PANIC CODE                                              00000370
120I                                                                        00000380
100I     CLEAR BATCH ABORT                                                  00000390
160I     ACCESS BATCH FILE RECORD (GET.BATCH)                               00000400
100I     IF BATCH RECORD FOUND (OK SET), THEN                               00000410
430I     !  DISPLAY BATCH HEADINGS                                          00000420
```

```
430I    !  DISPLAY BATCH INFORMATION                                        00000430
030I    !  ASK IF OK TO WRITE ?                                             00000440
        !  IF YES, THEN                                                     00000450
030I    !  !  ASK IF DISKETTE MOUNTED ?                                     00000460
        !  !  IF YES, THEN                                                  00000470
        !  !     CLEAR DISKETTE NOT MOUNTED FLAG                           00000480
08AL    !  !  !  INITIALIZE PANIC CODE                                     00000490
120I    !  !  !                                                            00000500
570I    !  !  !  SET DISKETTE COMMAND WORD                                 00000510
330I    !  !  !                                                            00000520
230I    !  !  !  WRITE DISKETTE HEADER RECORD (WRITE.BATCH)                00000530
100I    !  !  !  IF HEADER RECORD WRITTEN (OK SET), THEN                   00000540
        !  !  !  !  DEQUEUE TERMINAL                                       00000550
08AL    !  !  !  !  INITIALIZE PANIC CODE                                  00000560
130I    !  !  !  !                                                         00000570
130I    !  !  !  !  INITIALIZE ORDER COUNT                                 00000580
130I    !  !  !  !  DOUNTIL ORDER COUNT GREATER THAN MAXIMUM               00000590
170I    !  !  !  !     FIND FIRST/NEXT ORDER IN THIS BATCH (FIND.ORDER)    00000600
100I    !  !  !  !     IF ORDER RECORD FOUND (OK SET), THEN                00000610
330I    !  !  !  !        STORE ORDER ID IN TABLE                          00000620
        !  !  !  !     ELSE                                                00000630
100I    !  !  !  !        SET BATCH ABORT                                  00000640
        !  !  !  !     ENDIF                                               00000650
        !  !  !  !  ENDDO                                                  00000660
100I    !  !  !  !  IF BATCH ABORT NOT SET, THEN                          00000670
08AL    !  !  !  !  !  INITIALIZE PANIC CODE                              00000680
120I    !  !  !  !  !                                                      00000690
140I    !  !  !  !  !                                                      00000700
100I    !  !  !  !  !  CLEAR ORDER SEQUENCE, AND EXCEPTIONS               00000710
570I    !  !  !  !  !  SET DISKETTE COMMAND WORD                          00000720
330I    !  !  !  !  !                                                      00000730
140I    !  !  !  !  !  INITIALIZE CURRENT TRANSACTION BLOCK COUNT         00000740
140I    !  !  !  !  !  INITIALIZE TRANSACTION COUNT                       00000750
100I    !  !  !  !  !  DOUNTIL TRANSACTION COUNT GREATER THAN MAXIMUM     00000760
190I    !  !  !  !  !  !  FIND FIRST/NEXT TRANSACTION RECORD IN BATCH     00000770
        !  !  !  !  !  !  --- (FIND.TRANS) ---                            00000780
100I    !  !  !  !  !  !  IF TRANSACTION RECORD FOUND (OK SET), THEN      00000790
140I    !  !  !  !  !  !     IF TRANSACTION COUNT GREATER THAN MAX, THEN   00000800
570I    !  !  !  !  !  !        SET DISKETTE COMMAND WORD                  00000810
        !  !  !  !  !  !     (ELSE)                                        00000820
        !  !  !  !  !  !     ENDIF                                         00000830
330I    !  !  !  !  !  !     UPDATE ORDER ID FROM TABLE                    00000840
230I    !  !  !  !  !  !     WRITE NEXT DISKETTE RECORD (WRITE.BATCH)      00000850
100I    !  !  !  !  !  !     IF DISKETTE ERROR OCCURRED (OK CLEAR), THEN   00000860
140I    !  !  !  !  !  !        UPDATE TRANSACTION COUNT BEYOND MAXIMUM    00000870
        !  !  !  !  !  !     ELSE                                          00000880
        !  !  !  !  !  !     ENDIF                                         00000890
        !  !  !  !  !  !  ELSE                                            00000900
100I    !  !  !  !  !  !     SET BATCH ABORT                               00000910
        !  !  !  !  !  !  ENDIF                                           00000920
        !  !  !  !  !  ENDDO                                             00000930
        !  !  !  !  (ELSE)                                               00000940
        !  !  !  !  ENDIF                                                00000950
        !  !  !  (ELSE)                                                  00000960
        !  !  !  ENDIF                                                   00000970
        !  !  ELSE                                                       00000980
100I    !  !  !  SET BATCH ABORT                                         00000990
        !  !  ENDIF                                                      00001000
        !  ELSE                                                          00001010
100I    !  !  SET BATCH ABORT                                            00001020
        !  ENDIF                                                         00001030
        ELSE                                                            00001040
020I       DISPLAY BATCH ID, "BATCH NOT ON FILE"                        00001050
100I       SET BATCH ABORT                                              00001060
        ENDIF                                                           00001070
500I    RELEASE IPM BUFFER (FREE.IPM)                                   00001080
        ENQUEUE TERMINAL                                                00001090
100I    IF BATCH ABORT SET, THEN                                        00001100
040I       DISPLAY "WRITE BATCH COMMAND ABORTED"                        00001110
        (ELSE)                                                          00001120
        ENDIF                                                           00001130
        IF DISKETTE MOUNTED THEN                                        00001140
```

```
0430I       DISPLAY "REMOVE DISKETTE"                                    00001150
           (ELSE)                                                        00001160
           ENDIF                                                         00001170
1130I      CLEAR DISKETTE IN USE                                         00001180
           DEQUEUE TERMINAL                                              00001190
           ENDPROG                                                       00001200


                                                                         00000010
                                                                         00000020
                                                                         00000030
                                                                         00000040
                                                                         00000050
           WRITE DISKETTE DATA BLOCK SUBROUTINE                          00000060
                                                                         00000070
                                                                         00000080
                                                                         00000090
                        - MODULE HISTORY -                               00000100
                                                                         00000110
                                                                         00000120
              PROJECT:        DAS                  76-01709             00000130
                                                                         00000140
              SUB-SYSTEM:     DISKETTE MANAGER                           00000150
                                                                         00000160
              MODULE:         DMWDB                59                  00000170
                                                                         00000180
                                                                         00000190
                        - MODULE ABSTRACT -                              00000200
                                                                         00000210
                                                                         00000220
                THIS SUBROUTINE WHEN CALLED WILL PUT A HEADER OR         00000230
             DETAIL RECORD ON   A DATA INTERCHANGE FORMAT DISKETTE.      00000240
             WHEN THE HEADER RECORD IS PROCESSED THE DISKETTE WILL       00000250
             BE CHECKED AND BROUGHT ON-LINE TO EDX WITHOUT A $VARYON     00000260
             COMMAND. WHEN ALL RECORDS HAVE BEEN PROCESSED THE END-      00000270
             OF-DATA (EOD) POINTER ON THE DISKETTE HDR1 LABEL WILL BE    00000280
             UPDATED AND THE HEADER RECORD WILL BE UPDATED WITH THE      00000290
             RECORD COUNT.                                               00000300
                                                                         00000310
             NOTE:  THE CALLING PROGRAM MUST HAVE DS1=$$ AND AN ENTRY    00000320
             STATEMENT FOR DS1                                           00000330
                                                                         00000340
           IF COMMAND EQ TO 'PUT HEADER', THEN                           00000350
              BRING DISKETTE ON-LINE                                     00000360
              IF NO ERRORS, THEN                                         00000370
                 READ THE HDR1 RECORD FROM DISKETTE                      00000380
                 IF DISKETTE EXTENTS OK, THEN                            00000390
                    POINT TO FIRST RECORD                                00000400
                    MOVE 3 LOGICAL RECORDS TO BUFFER                     00000410
                    SET INITILIZE SWITCH                                 00000420
                    MOVE 'OK' TO RETURN CODE                             00000430
                 ELSE                                                    00000440
                    MOVE 'I/O ERROR TO RETURN CODE                       00000450
                 ENDIF                                                   00000460
              ELSE                                                       00000470
                 MOVE 'I/O ERROR' TO RETURN CODE                         00000480
              ENDIF                                                      00000490
           ELSE                                                          00000500
              IF COMMAND EQ TO 'PUT DETAIL RECORD' OR 'LAST DETAIL', THEN 00000510
                 IF INITILIZE SWITCH SET, THEN                           00000520
                    MOVE NEXT 24 BYTE LOGICAL RECORD TO BUFFER           00000530
                    MOVE 'OK' TO RETURN CODE                             00000540
                    IF END OF PHYSICAL BUFFER OR COMMAND EQ 'LAST DETAIL' 00000550
                       WRITE RECORD                                      00000560
                       IF NOT END OF FILE, THEN                          00000570
                          IF DISK WRITE ERROR, THEN                      00000580
                             MOVE 'I/O ERROR' TO RETURN CODE             00000590
                          ELSE                                           00000600
                             IF COMMAND EQ LAST RECORD THEN              00000610
                                PUT NUMBER OF RECORDS IN HEADER          00000620
                                WRITE HEADER RECORD                      00000630
                                IF DISK WRITE ERROR, THEN                00000640
                                   MOVE 'I/O ERROR' TO RETURN CODE       00000650
```

```
                        (ELSE)                                          00000660
                        ENDIF                                           00000670
                        CALCULATE EOD PNTR FROM NUMBER OF RECORDS       00000680
                        SET EOD POINTER IN HDR1 RECORD                  00000690
                        WRITE HDR1 RECORD TO DISK                       00000700
                        IF DISK WRITE ERROR, THEN                       00000710
                           MOVE 'I/O ERROR' TO RETURN CODE              00000720
                        (ELSE)                                          00000730
                        ENDIF                                           00000740
                     (ELSE)                                             00000750
                     ENDIF                                              00000760
                  ENDIF                                                 00000770
               ELSE                                                     00000780
                  MOVE 'I/O ERROR' TO RETURN CODE                       00000790
               ENDIF                                                    00000800
            ELSE                                                        00000810
               IF END OF FIRST 128 BYTES, THEN                          00000820
                  BUMP BUFFER POINTER TO NEXT 128 BYTES                 00000830
               (ELSE)                                                   00000840
               ENDIF                                                    00000850
            ENDIF                                                       00000860
         ELSE                                                           00000870
            MOVE 'INVALID CALL CODE' TO RETURN CODE                     00000880
         ENDIF                                                          00000890
      ELSE                                                              00000900
         MOVE 'INVALID CALL CODE' TO RETURN CODE                        00000910
      ENDIF                                                             00000920
   ENDIF                                                                00000930
   RETURN                                                               00000940


                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
   BATCH STATUS REPORT                                                  00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                     - MODULE HISTORY -                                 00000100
                                                                        00000110
                                                                        00000120
      PROJECT:       DAS                  73-01709                    00000130
                                                                        00000140
      SUB-SYSTEM:    OPERATOR INTERFACE                               00000150
                                                                        00000160
      MODULE:        DIBSR                15                          00000170
                                                                        00000180
                                                                        00000190
                     - MODULE ABSTRACT -                                00000200
                                                                        00000210
                                                                        00000220
      THIS MODULE REQUESTS THE BATCH ID. THE FILE MANAGER IS            00000230
   CALLED TO READ THE BATCH FILE INFORMATION.                           00000240
      IF THE BATCH INFORMATION IS FOUND, THE USER IS ASKED              00000250
   WHETHER OR NOT HARDCOPY OUTPUT IS REQUIRED. IF HARDCOPY IS           00000260
   REQUIRED, THEN THE TERMINAL IS DEQUEUED AND THE SYSTEM PRINTER       00000270
   ENQUEUED.                                                            00000280
      THE FILE MANAGER IS CALLED TO SEQUENTIALLY READ EACH ORDER        00000290
   RECORD. FOR EACH ORDER RECORD, THE ORDER ID AND ALL QUANTITIES       00000300
   AND ORDER PERCENT COMPLETE ARE OUTPUT. THE COMPLETED QUANTITY        00000310
   IS ACCUMULATED.                                                      00000320
      THE BATCH PERCENT COMPLETE IS OUTPUT TO THE ENQUEUED DEVICE.      00000330
   THE TERMINAL/PRINTER IS DEQUEUED BEFORE THE PROGRAM ENDS.            00000340
   ENQUEUE USER TERMINAL                                                00000350
04DI  DISPLAY ON USER TERMINAL "BATCH STATUS REPORT"                    00000360
48DI  DEQUEUE AN IPM BUFFER (GET.IPM)                                   00000370
49DI  DEQUEUE RECORD BUFFER (GET.SRB)                                   00000380
03DI  ASK USER FOR BATCH ID                                             00000390
12DI                                                                    00000400
08AL  INITIALIZE PANIC CODE                                             00000410
28DI                                                                    00000420
16DI  ACCESS BATCH FILE (GET.BATCH)                                     00000430
01FM                                                                    00000440
```

```
10JI    IF BATCH RECORD FOUND (OK SET), THEN                              00000450
10JI       SET BATCH FOUND                                                00000460
03JI       ASK USER IF HARDCOPY OUTPUT IS REQUIRED                        00000470
           IF HARDCOPY REQUIRED, THEN                                     00000480
10JI          SET HARDCOPY                                                00000490
              DEQUEUE USER TERMINAL                                       00000500
              ENQUEUE PRINTER                                             00000510
           ELSE                                                           00000520
10JI          CLEAR HARDCOPY                                              00000530
           ENDIF                                                          00000540
28JI       INITIALIZE DATE, TIME, PAGE COUNT AND LINE COUNT               00000550
06JI       PRINT/DISPLAY BATCH HEADINGS                                   00000560
06JI       PRINT/DISPLAY BATCH INFORMATION                                00000570
06JI       PRINT/DISPLAY ORDER HEADINGS                                   00000580
08AL       INITIALIZE PANIC CODE                                          00000590
28JI                                                                      00000600
13JI       INITIALIZE QUANTITY COMPLETE                                   00000610
13JI       INITIALIZE ORDER COUNT                                         00000620
13JI       DOUNTIL ORDER COUNT GREATER THAN MAXIMUM                       00000630
17JI          FIND THE FIRST/NEXT ORDER IN THIS BATCH (FIND.ORDER)        00000640
10JI          IF ORDER RECORD FOUND (OK SET), THEN                        00000650
13JI             COMPUTE ORDER PERCENT COMPLETE                           00000660
06JI             PRINT/DISPLAY ORDER INFORMATION AND PERCENT COMPLETE     00000670
                 --- (FORMS.CTL) ---                                      00000680
10JI             IF ABORT SET THEN                                        00000690
13JI                SET ORDER COUNT GREATER THAN MAX.                     00000700
10JI                CLEAR BATCH FOUND                                     00000710
                 (ELSE)                                                   00000720
                 ENDIF                                                    00000730
13JI             ACCUMULATE QUANTITY COMPLETE                             00000740
              ELSE                                                        00000750
10JI             CLEAR BATCH FOUND                                        00000760
04JI             PRINT/DISPLAY "BATCH NO LONGER ON FILE - RPT TERM."      00000770
              ENDIF                                                       00000780
           ENDDO                                                          00000790
           IF BATCH FOUND SET, THEN                                       00000800
28JI          COMPUTE BATCH PERCENT COMPLETE                              00000810
06JI          PRINT/DISPLAY BATCH PERCENT, "BATCH PERCENT COMPLETE"       00000820
           (ELSE)                                                         00000830
           ENDIF                                                          00000840
           IF HARD COPY NOT SET THEN                                      00000850
              MOVE REPORT TO TOP OF SCREEN                                00000860
           (ELSE)                                                         00000870
           ENDIF                                                          00000880
        ELSE                                                              00000890
02JI       DISPLAY BATCH ID, "BATCH NOT ON FILE"                          00000900
        ENDIF                                                             00000910
51JI    RELEASE RECORD BUFFER (FREE.SRB)                                  00000920
28JI    RELEASE THE IPM BUFFER (FREE.IPM)                                 00000930
        DEQUEUE TERMINAL/PRINTER                                          00000940
        ENDPROG                                                           00000950
              FORMS.CTL SUBROUTINE                                        00000960
                                                                          00000970
10JI    IF HARDCOPY SET, THEN                                             00000980
28JI       IF LINE COUNT GREATER THAN PRINTER PAGE, THEN                  00000990
10JI          SET OK                                                      00001000
           ELSE                                                           00001010
10JI          CLEAR OK                                                    00001020
           ENDIF                                                          00001030
        ELSE                                                              00001040
28JI       IF LINE COUNT GREATER THAN TERMINAL PAGE, THEN                 00001050
10JI          SET OK                                                      00001060
           ELSE                                                           00001070
10JI          CLEAR OK                                                    00001080
           ENDIF                                                          00001090
        ENDIF                                                             00001100
10JI    IF PAGE OVERFLOW (OK SET), THEN                                   00001110
28JI       INCREMENT PAGE COUNT                                           00001120
06JI       PRINT/DISPLAY BATCH HEADINGS                                   00001130
06JI       PRINT/DISPLAY BATCH INFORMATION                                00001140
06JI       PRINT/DISPLAY ORDER HEADINGS                                   00001150
28JI       INITIALIZE LINE COUNT                                          00001160
        (ELSE)                                                            00001170
```

```
        ENDIF                                                                   00001180
06JI    PRINT/DISPLAY ORDER INFORMATION                                         00001190
28JI    INCREMENT LINE COUNT                                                    00001200
        IF REPORT CANCELLED AND HARD COPY SET THEN                              00001210
           PRINT REPORT CANCELLED MESSAGE                                       00001220
        (ELSE)                                                                  00001230
        ENDIF                                                                   00001240
        RETURN                                                                  00001250

                                                                                00000010
                                                                                00000020
                                                                                00000030
                                                                                00000040
                                                                                00000050
        READ BATCH (FROM DISKETTE) COMMAND                                      00000060
                                                                                00000070
                                                                                00000080
                                                                                00000090
                          - MODULE HISTORY -                                    00000100
                                                                                00000110
                                                                                00000120
           PROJECT:       DAS                    79-01709                      00000130
                                                                                00000140
           SUB-SYSTEM:    OPERATOR INTERFACE                                   00000150
                                                                                00000160
           MODULE:        DIRDR                  20                            00000170
                                                                                00000180
                                                                                00000190
                          - MODULE ABSTRACT -                                   00000200
                                                                                00000210
                                                                                00000220
           THIS MODULE REQUESTS THE USER TO MOUNT A DISKETTE. THE               00000230
        DISKETTE HEADER IS READ AND DISPLAYED BACK TO THE USER. THE             00000240
        USER IS ASKED IF THIS DISKETTE IS OK TO READ. IF YES, THEN              00000250
        THE FILE MANAGER IS CALLED TO ADD THE BATCH. THE DISKETTE IS            00000260
        SEQUENTIALLY READ WHILE THE FILE MANAGER IS CALLED FOR EACH             00000270
        DISKETTE RECORD, TO ADD EACH TRANSACTION RECORD.                        00000280
                                                                                00000290
        ENQUEUE USER TERMINAL                                                   00000300
39FM    GET NUMBER OF SORT LINES                                                00000310
14JI    DISPLAY ON USER TERMINAL "READ BATCH COMMAND"                           00000320
03JI    ASK USER IF DISKETTE MOUNTED ?                                          00000330
        IF YES, THEN                                                            00000340
49JI    !  DEQUEUE RECORD BUFFER (GET.SRB)                                      00000350
08JI    !  INITIALIZE PANIC CODE                                                00000360
32JI    !                                                                       00000370
57JI    !  SET DISKETTE COMMAND WORD                                            00000380
32JI    !                                                                       00000390
22JI    !  GET DISKETTE HEADER RECORD (READ.BATCH)                              00000400
10JI    !  IF HEADER RECORD FOUND (OK SET), THEN                                00000410
43JI    !  !  DISPLAY BATCH HEADINGS                                            00000420
43JI    !  !  DISPLAY BATCH INFORMATION                                         00000430
10JI    !  !  CLEAR BATCH ABORT                                                 00000440
03JI    !  !  ASK IF OK TO READ ?                                               00000450
        !  !  IF YES, THEN                                                      00000460
48JI    !  !  !  DEQUEUE AN IPM BUFFER (GET.IPM)                                00000470
08AL    !  !  !  INITIALIZE PANIC CODE                                          00000480
12JI    !  !  !                                                                 00000490
24JI    !  !  !  ADD BATCH INFORMATION (ADD.BATCH)                              00000500
10JI    !  !  !  IF BATCH ADDED (OK SET), THEN                                  00000510
        !  !  !  !  DEQUEUE TERMINAL                                            00000520
08AL    !  !  !  !  INITIALIZE PANIC CODE                                       00000530
14JI    !  !  !  !                                                              00000540
57JI    !  !  !  !  SET DISKETTE COMMAND WORD                                   00000550
32JI    !  !  !  !                                                              00000560
10JI    !  !  !  !  CLEAR LAST RECORD                                           00000570
10JI    !  !  !  !  CLEAR DELETION                                              00000580
10JI    !  !  !  !  DOUNTIL LAST RECORD SET                                     00000590
22JI    !  !  !  !  !  GET DISKETTE TRANSACTION RECORD (READ.BATCH)             00000600
10JI    !  !  !  !  !  IF TRANSACTION RECORD FOUND (OK SET), THEN               00000610
        !  !  !  !  !     VALIDATE TRANSACTION RECORD INFORMATION               00000620
        !  !  !  !  !     --- (VALIDATE.TRANSACTION) ---                       00000630
10JI    !  !  !  !  !     IF TRANSACTION DATA VALID (OK SET), THEN              00000640
25JI    !  !  !  !  !        ADD TRANSACTION INFORMATIONN (ADD.TRANS)           00000650
```

```
10JI    !  !  !  !  !        IF ERROR (OK CLEAR), THEN                    00000650
10JI    !  !  !  !  !           SET LAST RECORD                           00000660
10JI    !  !  !  !  !           SET BATCH ABORT                           00000670
        !  !  !  !  !        (ELSE)                                       00000680
        !  !  !  !  !        ENDIF                                        00000690
        !  !  !  !  !     ELSE                                            00000700
10JI    !  !  !  !  !        SET LAST RECORD                           00000710
10JI    !  !  !  !  !        SET BATCH ABORT                           00000720
10JI    !  !  !  !  !        SET DELETION                              00000730
        !  !  !  !  !     ENDIF                                           00000740
        !  !  !  !  !  ELSE                                             00000750
10JI    !  !  !  !  !     SET LAST RECORD                              00000760
10JI    !  !  !  !  !     SET BATCH ABORT                              00000770
10JI    !  !  !  !  !     SET DELETION                                 00000780
        !  !  !  !  !  ENDIF                                           00000790
        !  !  !  !  ENDDO                                              00000800
        !  !  !. ELSE                                                   00000810
10JI    !  !  !     SET BATCH ABORT                                    00000820
        !  !  !  ENDIF                                                 00000830
25JI    !  !  !  RELEASE IPM BUFFER (FREE.IPM)                        00000840
        !  !  ELSE                                                     00000850
10JI    !  !     SET BATCH ABORT                                       00000860
        !  !  ENDIF                                                    00000870
        !  ELSE                                                        00000880
10JI    !     SET BATCH ABORT                                          00000890
        !  ENDIF                                                       00000900
51JI    !  RELEASE RECORD BUFFER (FREE.SRB)                           00000910
        ELSE                                                        00000920
10I        SET BATCH ABORT                                          00000930
        ENDIF                                                       00000940
10JI    IF DELETION SET, THEN                                       00000950
12JI       INITIALIZE BATCH DIRECTORY ADDRESS, ENTRY SIZE, BATCH COUNT 00000960
12JI       DOWHILE RESIDENT BATCH STATUS NOT ADDING                 00000970
              ADD 1 TO BATCH COUNT                                  00000980
              INDEX TO NEXT REC. IN RESIDENT BATCH FILE             00000990
           ENDDO                                                    00001000
02FM       CHANGE BATCH STATUS TO DELETING                          00001010
53JI       USE FILE MANAGER TO DELETE THIS BATCH (DELETE.BATCH)     00001020
        (ELSE)                                                      00001030
        ENDIF                                                       00001040
10JI    IF BATCH ABORT SET, THEN                                    00001050
           ENQUEUE TERMINAL                                         00001060
04JI       DISPLAY "READ BATCH COMMAND ABORTED"                     00001070
        (ELSE)                                                      00001080
        ENDIF                                                       00001090
04JI    DISPLAY "REMOVE DISKETTE"                                   00001100
11JO    CLEAR DISKETTE IN USE                                       00001110
        DEQUEUE USER TERMINAL                                       00001120
        ENDPROG                                                     00001130
                                                                    00001140
                                                                    00001150
              VALIDATE TRANSACTION SUBROUTINE                       00001160
                                                                    00001170
10JI    SET OK                                                      00001180
07FM    IF TRANSACTION ID ZERO OR NOT NUMERIC, THEN                 00001190
10JI       CLEAR OK                                                 00001200
        (ELSE)                                                      00001210
        ENDIF                                                       00001220
07FM    IF STATUS NOT VALID OR NOT NUMERIC THEN                     00001230
10JI       CLEAR OK                                                 00001240
        (ELSE)                                                      00001250
        ENDIF                                                       00001260
07FM    IF DESTINATION GREATER THAN MAXIMUM OR NOT NUMERIC, THEN    00001270
10JI       CLEAR OK                                                 00001280
        (ELSE)                                                      00001290
        ENDIF                                                       00001300
10JI    IF VALIDATION FAILED (OK CLEAR), THEN                       00001310
02JI       DISPLAY "TRANSACTION VALIDATION ERROR"                   00001320
44JI       DISPLAY DISKETTE TRANSACTION INFORMATION                 00001330
        (ELSE)                                                      00001340
        ENDIF                                                       00001350
        RETURN                                                      00001360
```

```
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
DISKETTE MANAGER READ SUBROUTINE                                    00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                - MODULE HISTORY -                                  00000100
                                                                    00000110
                                                                    00000120
   PROJECT:       DAS                  76-01709                    00000130
                                                                    00000140
   SUB-SYSTEM:    DISKETTE MANAGER                                 00000150
                                                                    00000160
   MODULE:        DMRDB                58                          00000170
                                                                    00000180
                                                                    00000190
                - MODULE ABSTRACT -                                 00000200
                                                                    00000210
                                                                    00000220
     THIS SUBROUTINE WHEN CALLED WILL GET A HEADER OR               00000230
   DETAIL RECORD FROM A DATA INTERCHANGE FORMAT DISKETTE.           00000240
   WHEN THE HEADER RECORD IS REQUESTED THE DISKETTE WILL            00000250
   BE CHECKED AND BROUGHT ON-LINE TO EDX WITHOUT A $VARYON          00000260
   COMMAND. WHEN ALL RECORDS HAVE BEEN PROCESSED FROM THE           00000270
   DISKETTE, THE RETURN CODE WILL BE SET TO INDICATE THE            00000280
   LAST RECORD HAS BEEN PROCESSED.                                  00000290
                                                                    00000300
IF COMMAND EQ TO 'GET HEADER', THEN                                 00000310
   BRING DISKETTE ON-LINE                                           00000320
   IF NO ERRORS, THEN                                               00000330
      READ THE HDR1 RECORD FROM DISKETTE                            00000340
      IF DISKETTE EXTENTS OK, THEN                                  00000350
         READ FIRST DATA RECORD                                     00000360
         IF DISK READ OK, THEN                                      00000370
            SET NUMBER OF RECORDS FROM HEADER                       00000380
            IF NUMBER OF RECORDS BETWEEN 4 AND 9490, THEN           00000390
               SET INITILIZE SWITCH                                 00000400
               MOVE 3 LOGICAL RECORDS TO BUFFER                     00000410
               MOVE 'OK' TO RETURN CODE                             00000420
               DECREMENT RECORD COUNT                               00000430
            ELSE                                                    00000440
               MOVE 'DISKETTE FORMAT ERROR TO RTN CODE              00000450
            ENDIF                                                   00000460
         ELSE                                                       00000470
            MOVE 'I/O ERROR' TO RETURN CODE                         00000480
         ENDIF                                                      00000490
      ELSE                                                          00000500
         MOVE 'I/O ERROR' TO RETURN CODE                            00000510
      ENDIF                                                         00000520
   ELSE                                                             00000530
      MOVE 'I/O ERROR' TO RETURN CODE                               00000540
   ENDIF                                                            00000550
ELSE                                                                00000560
   IF COMMAND EQ TO 'GET DETAIL RECORD', THEN                       00000570
      IF INITILIZE SWITCH SET, THEN                                 00000580
         MOVE NEXT 24 BYTE LOGICAL RECORD TO BUFFER                 00000590
         MOVE 'OK' TO RETURN CODE                                   00000600
         IF ALL RECORDS HAVE BEEN PROCESSED, THEN                   00000610
            MOVE 'LAST RECORD' TO RETURN CODE                       00000620
         ELSE                                                       00000630
            IF END OF FIRST 128 BYTE BUFFER, THEN                   00000640
               BUMP BUFFER POINTER TO NEXT 128 BYTES                00000650
            ELSE                                                    00000660
               IF END OF PHYSICAL BUFFER, THEN                      00000670
                  READ NEXT DISK RECORD                             00000680
                  IF DISK READ ERRORS, THEN                         00000690
                     MOVE 'I/O ERROR' TO RETURN CODE                00000700
                  (ELSE)                                            00000710
                  ENDIF                                             00000720
               (ELSE)                                               00000730
               ENDIF                                                00000740
```

```
                ENDIF                                                   00000750
              ENDIF                                                     00000760
            ELSE                                                        00000770
              MOVE 'INVALID CALL CODE' TO RETURN CODE                   00000780
              PANIC                                                     00000790
          ! ENDIF                                                       00000800
          ELSE                                                          00000810
            MOVE 'INVALID CALL CODE' TO RETURN CODE                     00000820
          ENDIF                                                         00000830
        ENDIF                                                           00000840
        RETURN                                                          00000850

                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
        TRANSACTION STATUS REPORT                                       00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
          PROJECT:      DAS                 78-01709                  00000130
                                                                        00000140
          SUB-SYSTEM:   OPERATOR INTERFACE                              00000150
                                                                        00000160
          MODULE:       DITSR               18                        00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
          THIS MODULE REQUESTS A TRANSACTION NUMBER, THEN CALLS THE     00000230
        FILE MANAGER TO READ THE TRANSACTION FILE.                      00000240
          IF THE TRANSACTION RECORD IS FOUND, THE FILE MANAGER IS AGAIN 00000250
        CALLED, THIS TIME TO READ THE DISK BATCH FILE.                  00000260
          IF THE DISK BATCH FILE INFORMATION IS FOUND, THE USER IS      00000270
        ASKED WHETHER OR NOT HARDCOPY OUTPUT IS REQUIRED. IF HARDCOPY IS00000280
        REQUIRED, THEN THE TERMINAL IS DEQUEUED AND THE SYSTEM PRINTER  00000290
        IS ENQUEUED.                                                    00000300
          THE COMBINED BATCH AND TRANSACTION FILE INFORMATION IS THEN   00000310
        OUTPUT TO THE ENQUEUED DEVICE.                                  00000320
          THE TERMINAL/PRINTER IS DEQUEUED BEFORE THE PROGRAM ENDS.     00000330
        ENQUEUE USER TERMINAL                                           00000340
040I    DISPLAY ON USER TERMINAL "TRANSACTION STATUS REPORT"            00000350
        CLEAR THE SCREEN                                                00000360
140I                                                                    00000370
480I    DEQUEUE AN IPM BUFFER (GET.IPM)                                 00000380
490I    DEQUEUE RECORD BUFFER (GET.SRB)                                 00000390
030I    ASK USER FOR TRANSACTION ID                                     00000400
        IF VALID (0<TRANSACTION ID<=999999), THEN                       00000410
08AL    ! INITIALIZE PANIC CODE                                         00000420
140I    !                                                               00000430
20FM    ! UPDATE IPM FOR CALL                                           00000440
08FM    ! CALL FILE MANGR TO TRANSACTION DIRECTORY RECORD BY TRANS. ID  00000450
20FM    ! CASENTRY (FOK, FTN, FCC, ? )                                  00000460
        !    CASE FOK ( SUCCESSFUL )                                    00000470
100I    !       SET OK                                                  00000480
03FM    !       SAVE THE MIS-SORT DESTINATION                           00000490
        !    CASE FTN ( TRANSACTION NOT FOUND )                         00000500
020I    !       DISPLAY TRANSACTION ID, "TRANSACTION NOT ON FILE"        00000510
100I    !       CLEAR OK                                                00000520
        !    CASE FCC ( INVALID CALL CODE )                             00000530
540I    !       PANIC                                                   00000540
        !    CASE  ?  ( INVALID COMPLETION CODE )                       00000550
540I    !       PANIC                                                   00000560
        ! ENDCASE                                                       00000570
100I    ! IF TRANSACTION DIRECTORY RECORD FOUND (OK SET), THEN          00000580
120I    ! ! GET BATCH ID FROM SYSCOM BATCH DIRECTORY                    00000590
02FM    ! !                                                            00000600
```

```
16DI    !  !  ACCESS BATCH FILE (GET.BATCH)                                  00000610
10DI    !  !  IF BATCH RECORD FOUND (OK SET), THEN                           00000620
08AL    !  !  !  INITIALIZE PANIC CODE                                       00000630
14DI    !  !  !                                                              00000640
21DI    !  !  !  ACCESS TRANSACTION FILE (GET.TRANS)                         00000650
10DI    !  !  !  IF TRANSACTION RECORD FOUND (OK SET), THEN                  00000660
23FM    !  !  !     CALL FILE MANAGER FOR ORDER ID                           00000670
23FM    !  !  !     IF ORDER RECORD FOUND, THEN                              00000680
05FM    !  !  !        SAVE ORDER ID                                         00000690
03DI    !  !  !        ASK USER IF HARDCOPY OUTPUT IS REQUIRED               00000700
        !  !  !        IF HARDCOPY REQUIRED, THEN                            00000710
10DI    !  !  !           SET HARD COPY FLAG                                 00000720
        !  !  !           DEQUEUE USER TERMINAL                              00000730
        !  !  !           ENQUEUE PRINTER                                    00000740
        !  !  !        ELSE                                                  00000750
10DI    !  !  !           CLEAR HARD COPY FLAG                               00000760
        !  !  !        ENDIF                                                 00000770
30DI    !  !  !        INITIALIZE DATE, TIME AND PAGE COUNT                  00000780
08DI    !  !  !        PRINT/DISPLAY BATCH HEADINGS                          00000790
08DI    !  !  !        PRINT/DISPLAY BATCH INFORMATION                       00000800
08DI    !  !  !        PRINT/DISPLAY TRANSACTION HEADINGS                    00000810
        !  !  !        IF TRANS. STATUS = MIS-SORT THEN                      00000820
        !  !  !           IF MIS-SORT DESTINATION = -1 THEN                  00000830
        !  !  !              SET MIS-SORT DEST. = '?'                        00000840
        !  !  !           ELSE                                               00000850
        !  !  !              SET MIS-SORT DEST. IN OUTPUT LINE               00000860
        !  !  !           ENDIF                                              00000870
        !  !  !        (ELSE)                                                00000880
        !  !  !        ENDIF                                                 00000890
08DI    !  !  !        PRINT/DISPLAY TRANSACTION INFORMATION                 00000900
10DI    !  !  !        IF HARD COPY FLAG NOT SET, THEN                       00000910
        !  !  !           MOVE DISPLAY TO TOP OF SCREEN                      00000920
        !  !  !        (ELSE)                                                00000930
        !  !  !        ENDIF                                                 00000940
        !  !  !     ELSE                                                     00000950
23FM    !  !  !        IF BATCH WAS NOT FOUND, THEN                          00000960
02DI    !  !  !           DISPLAY BATCH ID "BATCH NOT ON FILE"               00000970
        !  !  !        ELSE                                                  00000980
23FM    !  !  !           IF THE RELATIVE ORDER NUMBER WAS INVALID,THEN      00000990
08AL    !  !  !              SET UP PANIC CODE FOR INVLD REL ORD             00001000
        !  !  !           ELSE                                               00001010
08AL    !  !  !              SET UP PANIC CODE FOR INVLD CALL CODE           00001020
        !  !  !           ENDIF                                              00001030
07AL    !  !  !           PANIC                                              00001040
        !  !  !        ENDIF                                                 00001050
        !  !  !     ENDIF                                                    00001060
        !  !  !  ELSE                                                        00001070
02DI    !  !  !     DISPLAY TRANSACTION ID, "TRANSACTION NOT ON FILE"        00001080
        !  !  !  ENDIF                                                       00001090
        !  !  ELSE                                                          00001100
02DI    !  !  !  DISPLAY BATCH ID "BATCH NOT ON FILE"                       00001110
        !  !  ENDIF                                                         00001120
        !  (ELSE)                                                           00001130
        !  ENDIF                                                            00001140
        ELSE                                                               00001150
02DI    !  DISPLAY INVALID TRANSATCION NO                                  00001160
        ENDIF                                                              00001170
51DI    RELEASE RECORD BUFFER (FREE.SRB)                                   00001180
50DI    RELEASE THE IPM BUFFER (FREE.IPM)                                  00001190
        DEQUEUE TERMINAL/PRINTER                                           00001200
        ENDPROG                                                            00001210
```

```
                                                                           00000010
                                                                           00000020
                                                                           00000030
                                                                           00000040
                                                                           00000050
FIND DISK TRANSACTION FILE RECORD BY CRITERIA ( FIND.TRANS )               00000060
                                                                           00000070
                                                                           00000080
                                                                           00000090
```

```
                        - MODULE HISTORY -                                  00000100
                                                                            00000110
                                                                            00000120
   PROJECT:         DAS                    78-01709                        00000130
                                                                            00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                                      00000150
                                                                            00000160
   MODULE:          DICCFT                 78                              00000170
                                                                            00000180
                                                                            00000190
                        - MODULE ABSTRACT -                                 00000200
                                                                            00000210
                                                                            00000220
      EACH RELATIVE TRANSACTION RECORD IS SEQUENTIALLY                      00000230
   ACCESSED ONE AT A TIME BY CALLING THE SEQ.TRANS                          00000240
   SUBROUTINE FOR EACH RELATIVE ORDER NUMBER.                               00000250
      IF SUCCESSFUL, THE ORDER SEQUENCE INDICATOR IS                        00000260
   EXAMINED. IF SET, THE TRANSACTION RELATIVE ORDER                         00000270
   NUMBER IS COMPARED TO THE CURRENT RELATIVE ORDER                         00000280
   NUMBER. IF MATCHED, THE EXCEPTIONS INDICATOR IS                          00000290
   EXAMINED. IF SET, THE TRANSACTION STATUS IS EX-                          00000300
   AMINED FOR EXCEPTION STATUS.                                             00000310
      THIS SUBROUTINE WILL EXIT WHEN EITHER:                                00000320
   1) THE DISK TRANSACTION FILE HAS BEEN ACCESSED                           00000330
      WITHOUT ERROR, AND THE (OPTIONAL) SELECTION                           00000340
      CRITERIA (ORDER SEQUENCE, EXCEPTIONS) HAS BEEN                        00000350
      MET, OR                                                               00000360
   2) THE DISK TRANSACTION FILE HAS BEEN COMPLETELY                         00000370
      ACCESSED.                                                             00000380
14DI  DOUNTIL TRANSACTION COUNT GREATER THAN MAXIMUM OR OK SET--             00000390
12DI  -- OR TRANSACTION COUNT = ZERO                                        00000400
10DI                                                                        00000410
20DI     FIND FIRST/NEXT TRANSACTION IN THIS BATCH (SEQ.TRANS)              00000420
10DI     IF TRANSACTION RECORD FOUND (OK SET), THEN                         00000430
14DI        COMPUTE TRANSACTION RECORD ENTRY ADDRESS                        00000440
04FM        GET TRANSACTION INFORMATION                                     00000450
14DI                                                                        00000460
10DI        IF ORDER SEQUENCE SET, THEN                                     00000470
14DI           IF RELATIVE ORDER NUMBER MATCH, THEN                         00000480
13DI                                                                        00000490
10DI              IF EXCEPTIONS SET, THEN                                   00000500
14DI                 IF TRANSACTION STATUS NOT AN EXCEPTION, THEN           00000510
10DI                    CLEAR OK                                            00000520
                     (ELSE)                                                 00000530
                     ENDIF                                                  00000540
                  (ELSE)                                                    00000550
                  ENDIF                                                     00000560
               ELSE                                                         00000570
10DI              CLEAR OK                                                  00000580
               ENDIF                                                        00000590
            (ELSE)                                                          00000600
         (ELSE)                                                             00000610
         ENDIF                                                              00000620
         IF TRANSACTION COUNT > ZERO THEN                                   00000630
            INCREMENT TRANSACTION COUNT                                     00000640
         (ELSE)                                                             00000650
         ENDIF                                                              00000660
      ENDDO                                                                 00000670
      RETURN                                                                00000680

                                                                            00000010
                                                                            00000020
                                                                            00000030
                                                                            00000040
                                                                            00000050
      END BATCH COMMAND                                                     00000060
                                                                            00000070
                                                                            00000080
                                                                            00000090
                        - MODULE HISTORY -                                  00000100
                                                                            00000110
                                                                            00000120
   PROJECT:         DAS                    78-01709                        00000130
                                                                            00000140
```

```
      SUB-SYSTEM:     OPERATOR INTERFACE                                00000150
                                                                        00000160
      MODULE:         DIEN8                 23                        00000170
                                                                        00000180
                                                                        00000190
                      - MODULE ABSTRACT -                               00000200
                                                                        00000210
                                                                        00000220
         THIS MODULE REQUESTS A BATCH ID. THE FILE MANAGER IS CALLED    00000230
      TO READ THE CORRESPONDING BATCH INFORMATION. IF THE BATCH RECORD  00000240
      IS FOUND, THEN THE BATCH ID, HEADER AND STATUS ARE DISPLAYED.     00000250
         THE USER IS ASKED WHETHER OR NOT BATCH IS TO BE ENDED. IF      00000260
      YES, THEN THE FILE MANAGER IS CALLED TO CHANGE THE BATCH STATUS   00000270
      ENQUEUE USER TERMINAL                                             00000280
040I  DISPLAY ON USER TERMINAL "END BATCH COMMAND"                      00000290
480I  DEQUEUE AN IPM BUFFER (GET.IPM)                                   00000300
490I  DEQUEUE RECORD BUFFER (GET.SRB)                                   00000310
030I  ASK USER FOR BATCH ID                                             00000320
120I                                                                    00000330
08AL  INITIALIZE PANIC CODE                                             00000340
350I                                                                    00000350
160I  ACCESS BATCH FILE (GET.BATCH)                                     00000360
100I  IF BATCH RECORD FOUND (OK SET), THEN                              00000370
430I     DISPLAY BATCH HEADER                                           00000380
430I     DISPLAY BATCH INFORMATION                                      00000390
030I     ASK USER OK TO END BATCH                                       00000400
         IF YES, THEN                                                   00000410
08AL        INITIALIZE PANIC CODE                                       00000420
350I                                                                    00000430
01FM        INITIALIZE NEW BATCH STATUS                                 00000440
120I                                                                    00000450
450I        CHANGE BATCH STATUS (CHANGE.BATCH)                          00000460
            IF OK SET THEN                                              00000470
               DISPLAY "BATCH ENDED"                                    00000480
520I           ALARM LOG BATCH STATUS (SEND.ALARM)                      00000490
            (ELSE)                                                      00000500
            ENDIF                                                       00000510
         ELSE                                                           00000520
            DISPLAY "COMMAND ABORTED"                                   00000530
         ENDIF                                                          00000540
      ELSE                                                              00000550
320I     DISPLAY BATCH ID, "BATCH NOT ON FILE"                          00000560
         DISPLAY "COMMAND ABORTED"                                      00000570
      ENDIF                                                             00000580
510I  RELEASE RECORD BUFFER (FREE.SRB)                                  00000590
500I  RELEASE THE IPM BUFFER (FREE.IPM)                                 00000600
      ENDPROG                                                           00000610

                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
      SORT TRANSLATION REPORT                                           00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                      - MODULE HISTORY -                                00000100
                                                                        00000110
                                                                        00000120
      PROJECT:        DAS                   78-01709                  00000130
                                                                        00000140
      SUB-SYSTEM:     OPERATOR INTERFACE                                00000150
                                                                        00000160
      MODULE:         DISTR                 19                        00000170
                                                                        00000180
                                                                        00000190
                      - MODULE ABSTRACT -                               00000200
                                                                        00000210
                                                                        00000220
         THIS MODULE REQUESTS WHETHER OR NOT HARDCOPY IS REQUIRED. IF   00000230
      HARDCOPY IS REQUIRED, THEN THE TERMINAL IS DEQUEUED AND THE       00000240
      SYSTEM PRINTER IS ENQUEUED. THE RESIDENT SORT TRANSLATION TABLE   00000250
      IS FORMATTED AND OUTPUT TO THE ENQUEUED DEVICE.                   00000260
         THE TERMINAL/PRINTER IS DEQUEUED BEFORE THE PROGRAM ENDS.      00000270
```

```
                                                                        00000280
                                                                        00000290
                                                                        00000300
        ENQUEUE USER TERMINAL                                           00000310
39FM    GET NUMBER OF SORT LINES FROM SYSCOM                            00000320
04DI    DISPLAY ON USER TERMINAL "SORT LINE ROUTING REPORT"             00000330
03DI    ASK USER IF HARDCOPY OUTPUT IS REQUIRED                         00000340
        IF HARDCOPY REQUIRED, THEN                                      00000350
           DEQUEUE USER TERMINAL                                        00000360
           ENQUEUE PRINTER                                              00000370
        (ELSE)                                                          00000380
        ENDIF                                                           00000390
31DI    INITIALIZE DATE, TIME AND PAGE COUNT                            00000400
31DI    INITIALIZE SORT LINE COUNT                                      00000410
31DI    INITIALIZE POINTER TO SYSCOM SORT TRANSLATION TABLE             00000420
09DI    PRINT/DISPLAY REPORT HEADINGS                                   00000430
31DI    DOUNTIL END OF SYSCOM SORT TRANSLATION TABLE                    00000440
09DI       PRINT/DISPLAY TRANSLATION INFORMATION                        00000450
31DI       INCREMENT POINTER TO SORT TRANSLATION TABLE                  00000460
        ENDDO                                                           00000470
        DEQUEUE TERMINAL/PRINTER                                        00000480
        ENDPROG                                                         00000490


                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
        WORK SCHEDULE REPORT                                            00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
           PROJECT:     DAS                 79-01709                    00000130
                                                                        00000140
           SUB-SYSTEM:  OPERATOR INTERFACE                              00000150
                                                                        00000160
           MODULE:      DIWSR               15                          00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
           THIS MODULE REQUESTS WHETHER OR NOT HARDCOPY OUTPUT IS       00000230
        REQUIRED. IF HARDCOPY REQUIRED, THEN THE TERMINAL IS DEQUEUED   00000240
        AND THE SYSTEM PRINTER IS ENQUEUED. THE RESIDENT BATCH DIR-     00000250
        ECTORY IS CONSULTED FOR BATCH ENTRIES IN USE.                   00000260
           FOR EACH BATCH ENTRY IN USE, THE FILE MANAGER IS CALLED      00000270
        TO READ THE BATCH FILE RECORD. THE BATCH INFORMATION IS THEN    00000280
        OUTPUT TO THE ENQUEUED DEVICE.                                  00000290
           THE TERMINAL/PRINTER IS DEQUEUED BEFORE THE PROGRAM ENDS.    00000300
        ENQUEUE USER TERMINAL                                           00000310
        CLEAR THE SCREEN                                                00000320
04DI    DISPLAY ON USER TERMINAL "WORK SCHEDULE REPORT"                  00000330
03DI    ASK USER IF HARDCOPY OUTPUT IS REQUIRED                         00000340
        IF HARDCOPY REQUIRED, THEN                                      00000350
           SET HARD COPY FLAG                                           00000360
           DEQUEUE USER TERMINAL                                        00000370
           ENQUEUE PRINTER                                              00000380
        ELSE                                                            00000390
           CLEAR HARD COPY FLAG                                         00000400
        ENDIF                                                           00000410
48DI    DEQUEUE AN IPM BUFFER (GET.IPM)                                 00000420
49DI    DEQUEUE RECORD BUFFER (GET.SRB)                                 00000430
27DI    INITIALIZE DATE, TIME AND PAGE COUNT                            00000440
05DI    PRINT/DISPLAY BATCH HEADINGS                                    00000450
10DI    CLEAR BATCH FOUND                                               00000460
02FM    INITIALIZE POINTER TO SYSCOM BATCH DIRECTORY                    00000470
12DI                                                                    00000480
02FM    INITIALIZE SYSCOM BATCH DIRECTORY ENTRY SIZE                    00000490
12DI                                                                    00000500
```

```
084L    INITIALIZE PANIC CODE                                                   00000510
273I                                                                            00000520
120I    INITIALIZE BATCH COUNT                                                  00000530
120I    DOUNTIL BATCH COUNT GREATER THAN MAXIMUM                                00000540
150I       FIND AN ACTIVE, PENDING OR COMPLETE BATCH RECORD (FIND.BATCH)        00000550
100I       IF BATCH RECORD FOUND (OK SET), THEN                                 00000560
050I          PRINT/DISPLAY BATCH INFORMATION                                   00000570
273I                                                                            00000580
100I          SET BATCH FOUND                                                   00000590
           (ELSE)                                                               00000600
           ENDIF                                                                00000610
        ENDDO                                                                   00000620
510I    RELEASE RECORD BUFFER (FREE.SRB)                                        00000630
500I    RELEASE THE IPM BUFFER (FREE.IPM)                                       00000640
100I    IF BATCH FOUND NOT SET, THEN                                            00000650
040I       PRINT/DISPLAY "NO CURRENT BATCH INFORMATION"                         00000660
        (ELSE)                                                                  00000670
        ENDIF                                                                   00000680
        IF HARDCOPY FLAG NOT SET, THEN                                          00000690
           MOVE REPORT TO TOP OF SCREEN                                         00000700
        (ELSE)                                                                  00000710
        ENDIF                                                                   00000720
        DEQUEUE TERMINAL/PRINTER                                                00000730
        ENDPROG                                                                 00000740


                                                                                00000010
                                                                                00000020
                                                                                00000030
                                                                                00000040
                                                                                00000050
'GET A BLOCK BUFFER                                                             00000060
                                                                                00000070
                                                                                00000080
                                                                                00000090
                    - MODULE HISTORY -                                          00000100
                                                                                00000110
                                                                                00000120
   PROJECT:         DAS                    78-01709                            00000130
                                                                                00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                                         00000150
                                                                                00000160
   MODULE:          DICCGS                 63                                  00000170
                                                                                00000180
                                                                                00000190
                    - MODULE ABSTRACT -                                         00000200
                                                                                00000210
                                                                                00000220
        THIS SUBROUTINE DEQUEUES A 256 BYTE BLOCK BUFFER                        00000230
      FROM THE BUFFER QUEUE IN SYSCOM. IF THE QUEUE IS EMPTY                    00000240
      A DELAY IS PERFORMED BEFORE TRYING AGAIN.                                 00000250
                                                                                00000260
                                                                                00000270
                                                                                00000280
37FM    GET ADDRESS OF SBB (256-BYTE) BUFFER QUEUE IN SYSCOM                    00000290
        DOUNTIL BUFFER RECEIVED                                                 00000300
37FM       DEQUEUE A BUFFER FROM THE QUEUE                                      00000310
37FM       IF THE QUEUE WAS EMPTY THEN                                          00000320
              DELAY                                                             00000330
           (ELSE)                                                               00000340
           ENDIF                                                                00000350
        ENDDO                                                                   00000360
        RETURN                                                                  00000370


                                                                                00000010
                                                                                00000020
                                                                                00000030
                                                                                00000040
                                                                                00000050
START BATCH COMMAND                                                             00000060
                                                                                00000070
                                                                                00000080
                                                                                00000090
                    - MODULE HISTORY -                                          00000100
```

```
                                                                        00000110
                                                                        00000120
   PROJECT:        DAS                    73-01709                      00000130
                                                                        00000140
   SUB-SYSTEM:     OPERATOR INTERFACE                                   00000150
                                                                        00000160
   MODULE:         DISTR                  22                            00000170
                                                                        00000180
                                                                        00000190
                   - MODULE ABSTRACT -                                  00000200
                                                                        00000210
                                                                        00000220
     THIS MODULE REQUESTS A BATCH ID. THE FILE MANAGER IS CALLED        00000230
   TO READ THE CORRESPONDING BATCH INFORMATION. IF THE BATCH RECORD     00000240
   IS FOUND, THEN THE BATCH ID, HEADER AND STATUS ARE DISPLAYED.        00000250
     THE USER IS ASKED WHETHER OR NOT BATCH IS TO BE STARTED. IF        00000260
   YES, THEN THE FILE MANAGER IS CALLED TO CHANGE THE BATCH STATUS      00000270
   ENQUEUE USER TERMINAL                                                00000280
   CLEAR THE SCREEN                                                     00000290
04DI   DISPLAY ON USER TERMINAL "START BATCH COMMAND"                   00000300
48DI   DEQUEUE AN IPM BUFFER (GET.IPM)                                  00000310
49DI   DEQUEUE RECORD BUFFER (GET.SRB)                                  00000320
03DI   ASK USER FOR BATCH ID                                            00000330
12DI                                                                    00000340
08AL   INITIALIZE PANIC CODE                                            00000350
34DI                                                                    00000360
16DI   ACCESS BATCH FILE (GET.BATCH)                                    00000370
10DI   IF BATCH RECORD FOUND (OK SET), THEN                             00000380
43DI      DISPLAY BATCH HEADER                                          00000390
43DI      DISPLAY BATCH INFORMATION                                     00000400
03DI      ASK USER OK TO START BATCH                                    00000410
          IF YES, THEN                                                  00000420
08AL         INITIALIZE PANIC CODE                                      00000430
34DI                                                                    00000440
01FM         INITIALIZE NEW BATCH STATUS                                00000450
12DI                                                                    00000460
45DI         CHANGE BATCH STATUS (CHANGE.BATCH)                         00000470
             IF OK SET THEN                                             00000480
                DISPLAY "BATCH STARTED"                                 00000490
52DI            ALARM LOG BATCH STATUS (SEND.ALARM)                     00000500
             (ELSE)                                                     00000510
             ENDIF                                                      00000520
          ELSE                                                          00000530
             DISPLAY "COMMAND ABORTED"                                  00000540
          ENDIF                                                         00000550
       ELSE                                                             00000560
02DI      DISPLAY BATCH ID, "BATCH NOT ON FILE"                         00000570
          DISPLAY "COMMAND ABORTED"                                     00000580
       ENDIF                                                            00000590
51DI   RELEASE RECORD BUFFER (FREE.SRB)                                 00000600
50DI   RELEASE THE IPM BUFFER (FREE.IPM)                                00000610
       ENDPROG                                                          00000620
```

```
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
   READ RELATIVE DISK TRANSACTION FILE RECORD ( SEQ.TRANS )             00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                   - MODULE HISTORY -                                   00000100
                                                                        00000110
                                                                        00000120
   PROJECT:        DAS                    73-01709                      00000130
                                                                        00000140
   SUB-SYSTEM:     OPERATOR INTERFACE                                   00000150
                                                                        00000160
   MODULE:         DIOCST                 79                            00000170
                                                                        00000180
                                                                        00000190
```

```
                        - MODULE ABSTRACT -                                     00000200
                                                                                00000210
                                                                                00000220
              EACH RELATIVE TRANSACTION NUMBER IS USED TO                       00000230
           CALCULATE THE RELATIVE BLOCK NUMBER AND TRANSACTION                  00000240
           RECORD ENTRY OFFSET. IF THE TRANSACTION INFORMATION                  00000250
           REQUESTED RESIDES IN A BLOCK OTHER THAN THE CURRENT                  00000260
           BLOCK IN THE SYSCOM RECORD BUFFER, THE FILE MANAGER IS               00000270
           CALLED TO READ THE NEW BLOCK INTO THE SYSCOM RECORD                  00000280
           BUFFER.                                                              00000290
              IF SUCCESSFUL, OK IS SET. IF AN ERROR OCCURRED,                   00000300
           TELL THE USER, CLEAR OK AND TAKE ANY FUTHER APPRO-                   00000310
           PRIATE ACTION REQUIRED.                                              00000320
              THIS SUBROUTINE WILL EXIT WHEN EITHER:                            00000330
           1) THE TRANSACTION RECORD REQUESTED IS IN THE CUR-                   00000340
              RENT SYSCOM RECORD BUFFER, OR                                     00000350
           2) THE DISK TRANSACTION FILE HAS BEEN ACCESSED WITH-                 00000360
              OUT ERROR, OR                                                     00000370
           3) THE DISK TRANSACTION FILE HAS BEEN COMPLETELY                     00000380
              ACCESSED.                                                         00000390
                                                                                00000400
                                                                                00000410
                                                                                00000420
                                                                                00000430
14DI    COMPUTE TRANSACTION BLOCK NUMBER/ENTRY OFFSET                           00000440
14DI    IF NEW BLOCK NUMBER, THEN                                               00000450
55DI       DEQUEUE BLOCK BUFFER (GET.SBB)                                       00000460
14DI       SAVE TRANSACTION CURRENT BLOCK NUMBER                                00000470
21FM       UPDATE IPM FOR CALL                                                  00000480
08FM       CALL FILE MANAGER TO READ TRANSACTION FILE BY BLOCK NUMBER           00000490
           SAVE FM RETURN DATA                                                  00000500
56DI       RELEASE BLOCK BUFFER (FREE.SBB)                                      00000510
21FM       CASENTRY (FOK, FBN, FIR, FCC, ? )                                    00000520
              CASE FOK ( SUCCESSFUL )                                           00000530
10DI             SET OK                                                       00000540
              CASE FBN ( BATCH NOT FOUND )                                      00000550
10DI             CLEAR OK                                                     00000560
13DI             UPDATE ORDER COUNT BEYOND MAXIMUM                            00000570
14DI             SET TRANSACTION COUNT = ZERO                                 00000580
12DI                                                                          00000590
              CASE FCC ( INVALID CALL CODE )                                    00000600
54DI             PANIC                                                        00000610
              CASE FIR ( INVALID RELATIVE BLOCK )                               00000620
54DI             PANIC                                                        00000630
              CASE  ?  ( INVALID COMPLETION CODE )                              00000640
54DI             PANIC                                                        00000650
           ENDCASE                                                            00000660
        (ELSE)                                                                00000670
        ENDIF                                                                 00000680
        RETURN                                                                00000690

                                                                              00000010
                                                                              00000020
                                                                              00000030
                                                                              00000040
                                                                              00000050
        DELETE BATCH COMMAND                                                  00000060
                                                                              00000070
                                                                              00000080
                                                                              00000090
                        - MODULE HISTORY -                                    00000100
                                                                              00000110
                                                                              00000120
           PROJECT:         DAS                  78-01709                   00000130
                                                                              00000140
           SUB-SYSTEM:      OPERATOR INTERFACE                              00000150
                                                                              00000160
           MODULE:          DIDLB                24                         00000170
                                                                              00000180
                                                                              00000190
```

```
                              - MODULE ABSTRACT -                                00000200
                                                                                 00000210
                                                                                 00000220
        THIS MODULE ASKS THE USER IF ALL BATCHES TO BE DELETED. IF               00000230
      ALL BATCHES NOT TO BE DELETED, A SPECIFIC BATCH ID IS REQUESTED,           00000240
      AND THE FILE MANAGER IS CALLED TO READ THE BATCH FILE INFORM-              00000250
      ATION.                                                                     00000260
         IF ALL BATCHES TO BE DELETED, THE FILE MANAGER IS CALLED TO             00000270
      SEQUENTIALLY READ EACH BATCH RECORD. FOR EACH BATCH RECORD, OR            00000280
      THE SPECIFICLY REQUESTED BATCH RECORD, THE BATCH ID, HEADER AND           00000290
      BATCH STATUS ARE LISTED FOR USER VERIFICATION.                             00000300
         THE USER IS ASKED WHETHER OR NOT BATCH IS TO BE DELETED. IF             00000310
      YES, THEN THE FILE MANAGER IS CALLED TO DELETE THE BATCH.                  00000320
      ENQUEUE USER TERMINAL                                                      00000330
      CLEAR THE USERS SCREEN                                                     00000340
04DI  DISPLAY ON USER TERMINAL "DELETE BATCH COMMAND"                            00000350
48DI  DEQUEUE AN IPM BUFFER (#1) (GET.IPM)                                       00000360
48DI  DEQUEUE AN IPM BUFFER (#2) (GET.IPM)                                       00000370
49DI  DEQUEUE RECORD BUFFER (GET.SRB)                                            00000380
08AL  INITIALIZE PANIC CODE                                                      00000390
36DI                                                                             00000400
12DI  INITIALIZE POINTER TO SYSCOM BATCH DIRECTORY                               00000410
12DI  INITIALIZE SYSCOM BATCH DIRECTORY ENTRY SIZE                               00000420
08AL  INITIALIZE PANIC CODE                                                      00000430
03DI  ASK USER IF ALL BATCHES TO BE DELETED                                      00000440
36DI  IF NOT ALL BATCHES, THEN                                                   00000450
03DI     ASK USER FOR BATCH ID                                                   00000460
12DI                                                                             00000470
16DI     ACCESS BATCH FILE (GET.BATCH)                                           00000480
         IF THE BATCH IS ACTIVE, THEN                                            00000490
            SET THE ACTIVE FLAG                                                  00000500
         ELSE                                                                    00000510
            IF THE BATCH IS ACTIVE OR DELETING, THEN                             00000520
               CLEAR OK FLAG                                                     00000530
            (ELSE)                                                               00000540
            ENDIF                                                                00000550
         ENDIF                                                                   00000560
10DI     IF BATCH RECORD FOUND (OK SET), THEN                                    00000570
10DI        SET SPECIFIC BATCH                                                   00000580
         ELSE                                                                    00000590
02DI        DISPLAY BATCH ID, "BATCH NOT ON FILE"                                00000600
         ENDIF                                                                   00000610
      ELSE                                                                       00000620
         CLEAR THE USERS SCREEN                                                  00000630
10DI     SET OK                                                                  00000640
10DI     CLEAR SPECIFIC BATCH                                                    00000650
      ENDIF                                                                      00000660
10DI  IF BATCH FOUND (OK SET), THEN                                              00000670
10DI  ! CLEAR BATCH FOUND                                                        00000680
12DI  ! INITIALIZE BATCH COUNT                                                   00000690
12DI  ! COUNTIL BATCH COUNT GREATER THAN MAXIMUM                                 00000700
10DI  ! ! IF SPECIFIC BATCH SET, THEN                                            00000710
02FM  ! !    SAVE RELATIVE BATCH ID                                              00000720
10DI  ! !    SET BATCH FOUND                                                     00000730
      ! ! ELSE                                                                   00000740
15DI  ! !    FIND AN ACTIVE, PENDING OR COMPLETE BATCH RECORD                    00000750
      ! !    --- (FIND.BATCH) ---                                                00000760
      ! !    SAVE RELATIVE BATCH                                                 00000770
      ! !    IF THE BATCH IS ACTIVE, THEN                                        00000780
      ! !       SET THE ACTIVE FLAG                                              00000790
      ! !       CLEAR OK FLAG                                                    00000800
      ! !       SET BATCH FOUND FLAG                                             00000810
      ! !    (ELSE)                                                              00000820
      ! !    ENDIF                                                               00000830
      ! ! ENDIF                                                                  00000840
10DI  ! ! IF BATCH RECORD FOUND (OK SET), THEN                                   00000850
10DI  ! !    SET BATCH FOUND                                                     00000860
      ! !    IF THIS IS THE FIRST BATCH, THEN                                    00000870
      ! !       DISPLAY BATCH HEADINGS                                           00000880
      ! !    (ELSE)                                                              00000890
      ! !    ENDIF                                                               00000900
43DI  ! !    DISPLAY BATCH INFORMATION                                           00000910
```

```
        !  !     IF NOT ACTIVE OR ALL BATCHES, THEN                          00000920
        !  !        IF NOT ACTIVE, THEN                                      00000930
030I    !  !           ASK USER OK TO DELETE BATCH                           00000940
        !  !           IF YES, THEN                                          00000950
100I    !  !              SET DELETE FOUND                                   00000960
14FM    !  !              STORE RELATIVE BATCH NUMBER IN IPM BUFFER (#2)     00000970
        !  !           (ELSE)                                                00000980
        !  !           ENDIF                                                 00000990
        !  !        (ELSE)                                                   00001000
        !  !        ENDIF                                                    00001010
        !  !     ELSE                                                        00001020
020I    !  !        DISPLAY 'BATCH MUST BE PENDING OR COMPLETE'              00001030
        !  !     ENDIF                                                       00001040
        !  !  (ELSE)                                                         00001050
        !  !  ENDIF                                                          00001060
100I    !  !  IF SPECIFIC BATCH, THEN                                        00001070
120I    !  !     UPDATE BATCH COUNT BEYOND MAXIMUM                           00001080
        !  !  (ELSE)                                                         00001090
        !  !  ENDIF                                                          00001100
        !  !  CLEAR THE ACTIVE BATCH FLAG                                    00001110
        !  ENDDO                                                            00001120
100I    !  IF DELETE FOUND, THEN                                            00001130
530I    !  !  USE FILE MANAGER TO DELETE BATCH(ES) (DELETE.BATCH)            00001140
        !  ELSE                                                             00001150
100I    !  !  IF BATCH FOUND NOT SET, THEN                                   00001160
040I    !  !     DISPLAY 'NO CURRENT BATCH INFORMATION'                      00001170
        !  !  ELSE                                                           00001180
        !  !     IF ALL OF THE BATCHES WERE ACTIVE, THEN                     00001190
020I    !  !        DISPLAY 'ALL BATCHES ARE ACTIVE'                         00001200
        !  !     (ELSE)                                                      00001210
        !  !     ENDIF                                                       00001220
        !  !  ENDIF                                                          00001230
        !  ENDIF                                                            00001240
        (ELSE)                                                              00001250
        ENDIF                                                               00001260
510I    RELEASE RECORD BUFFER (FREE.SRB)                                    00001270
500I    RELEASE THE IPM BUFFER (#1) (FREE.IPM)                              00001280
500I    RELEASE THE IPM BUFFER (#2) (FREE.IPM)                              00001290
        DISPLAY 'BATCH DELETION ENDED'                                      00001300
        DEQUEUE USER TERMINAL                                               00001310
        ENDPROG                                                             00001320


                                                                            00000010
                                                                            00000020
                                                                            00000030
                                                                            00000040
                                                                            00000050
    READ DISKETTE BATCH/TRANSACTION RECORD ( READ.BATCH )                   00000060
                                                                            00000070
                                                                            00000080
                                                                            00000090
                        - MODULE HISTORY -                                  00000100
                                                                            00000110
                                                                            00000120
       PROJECT:         DAS                    76-01709                    00000130
                                                                            00000140
       SUB-SYSTEM:      OPERATOR INTERFACE                                  00000150
                                                                            00000160
       MODULE:          DICORB                 73                          00000170
                                                                            00000180
                                                                            00000190
                        - MODULE ABSTRACT -                                 00000200
                                                                            00000210
                                                                            00000220
            CALL THE DISKETTE SUBROUTINE TO READ A BATCH OR                 00000230
         TRANSACTION RECORD.                                                00000240
            THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL                   00000250
         WHETHER DISKETTE ACCESS WAS SUCCESSFUL OR NOT.                     00000260
    CALL READ DISKETTE DATA BLOCK                                           00000270
    CASENTRY (DOK, DIO, DBD, DLR, DCC, ? )                                  00000280
       CASE DOK ( SUCCESSFUL )                                              00000290
100I      SET OK                                                            00000300
       CASE DIO ( I/O ERROR OCCURRED )                                      00000310
          ENQUEUE USER TERMINAL                                             00000320
```

```
020I          DISPLAY "DISKETTE I/O ERROR OCCURRED"                    00000330
100I          SET LAST RECORD                                          00000340
100I          SET BATCH ABORT                                          00000350
100I          CLEAR OK                                                 00000360
           CASE DBD (INVALID RECORD COUNT)                             00000370
              ENQUEUE USER TERMINAL                                    00000380
              DISPLAY "DISKETTE FORMAT ERROR"                          00000390
              CLEAR OK FLAG                                            00000400
           CASE DLR ( LAST RECORD = SUCCESSFUL )                       00000410
100I          SET LAST RECORD                                          00000420
100I          SET OK                                                   00000430
           CASE DCC ( INVALID CALL CODE )                              00000440
540I          PANIC                                                    00000450
           CASE  ?  ( INVALID COMPLETION CODE )                        00000460
540I          PANIC                                                    00000470
        ENDCASE                                                       00000480
        RETURN                                                        00000490
```

```
                                                                      00000010
                                                                      00000020
                                                                      00000030
                                                                      00000040
                                                                      00000050
        PANIC                                                         00000060
                                                                      00000070
                                                                      00000080
                                                                      00000090
                        - MODULE HISTORY -                            00000100
                                                                      00000110
                                                                      00000120
           PROJECT:        DAS                    76-01709           00000130
                                                                      00000140
           SUB-SYSTEM:     OPERATOR INTERFACE                         00000150
                                                                      00000160
           MODULE:         DICCPA                 84                 00000170
                                                                      00000180
                                                                      00000190
                        - MODULE ABSTRACT -                           00000200
                                                                      00000210
                                                                      00000220
              THIS SUBROUTINE PERFORMS THE PANIC.  THE PANIC          00000230
           CODE IS RETRIEVED AND PUT ON THE ALARM QUEUE.  A           00000240
           DEQUEUE TERMINAL IS PERFORMED TO GET RID OF ANY LEFT       00000250
           OVER ENQUEUES.  THEN A WAIT FOR AN EVENT THAT WILL         00000260
           NEVER BE POSTED IS PERFORMED.                              00000270
                                                                      00000280
                                                                      00000290
                                                                      00000300
08AL    GET PANIC CODE                                                00000310
01AL    GET ADDRESS OF ALARM QUEUE IN SYSCOM                          00000320
        DEQUEUE TERMINAL / PRINTER                                    00000330
07AL    PUT PANIC CODE ON THE ALARM QUEUE                             00000340
        WAIT FOR NEVER EVENT                                          00000350
        RETURN                                                        00000360
```

```
                                                                      00000010
                                                                      00000020
                                                                      00000030
                                                                      00000040
                                                                      00000050
        WRITE SORT TRANSLATION TABLE TO DISK                          00000060
                                                                      00000070
                                                                      00000080
                                                                      00000090
                        - MODULE HISTORY -                            00000100
                                                                      00000110
                                                                      00000120
           PROJECT:        DAS                    76-01709           00000130
                                                                      00000140
           SUB-SYSTEM:     OPERATOR INTERFACE                         00000150
                                                                      00000160
           MODULE:         DICCWS                 82                 00000170
                                                                      00000180
```

```
        CHECKED:        S.R.MARSDEN             16-JAN-79               00000190
                                                                        00000200
                                                                        00000210
                        - MODULE ABSTRACT -                             00000220
                                                                        00000230
                                                                        00000240
           THIS SUBROUTINE WRITES THE SORT TRANSLATION TABLE            00000250
        TO THE DISK.  IT FIRST CLEARS THE 256 BYTE BUFFER AND           00000260
        THEN COPIES IN THE SORT TRANSLATION TABLE.  THE BUF-            00000270
        FER IS THEN WRITTEN TO THE DISK.  IF ANY ERRORS ARE             00000280
        ENCOUNTERED A PANIC IS PERFORMED.                               00000290
                                                                        00000300
                                                                        00000310
                                                                        00000320
06FM    GET ADDRESS OF SORT TRANSLATION TABLE IN SYSCOM                  00000330
        CLEAR WRITE BUFFER                                              00000340
06FM    COPY TRANSLATION TABLE IN SYSCOM TO THE BUFFER                  00000350
06FM    WRITE THE BUFFER TO DISK                                        00000360
        IF ANY DISK ERRORS WERE ENCOUNTERED THEN                        00000370
54DI       PANIC                                                        00000380
        (ELSE)                                                          00000390
        ENDIF                                                           00000400
        RETURN                                                          00000410


                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
        ALARM OUTPUT                                                    00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
           PROJECT:     DAS                     78-01709                00000130
                                                                        00000140
           SUB-SYSTEM:  OPERATOR INTERFACE                              00000150
                                                                        00000160
           MODULE:      DICCSA                  83                      00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
           THIS SUBROUTINE SENDS AN ALARM MESSAGE TO THE ALARM          00000230
        MESSAGE HANDLING SUB-SYSTEM.  IT GETS A BUFFER, PUTS            00000240
        IN THE MESSAGE NUMBER AND DATA, AND PUTS THE ADDRESS            00000250
        OF THE BUFFER ON THE ALARM QUEUE.  IF ANY ERRORS ARE            00000260
        ENCOUNTERED, PANICS ARE PERFORMED.                              00000270
                                                                        00000280
                                                                        00000290
                                                                        00000300
02AL    GET ADDRESS ALARM BUFFER QUEUE IN SYSCOM                        00000310
02AL    DEQUEUE A BUFFER                                                00000320
02AL    IF ONE RECIEVED THEN                                            00000330
03AL       PUT IN MESSAGE NUMBER                                        00000340
03AL       PUT IN MESSAGE DATA                                          00000350
01AL       GET ADDRESS OF ALARM QUEUE IN SYSCOM                         00000360
01AL       PUT ADDRESS OF BUFFER ON THE ALARM QUEUE                     00000370
           IF THE ALARM QUEUE WAS ALREADY FULL THEN                     00000380
54DI          PANIC                                                     00000390
           (ELSE)                                                       00000400
           ENDIF                                                        00000410
        ELSE                                                            00000420
           PANIC                                                        00000430
        ENDIF                                                           00000440
        RETURN                                                          00000450
```

```
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
FREE A RECORD BUFFER                                                00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                 73-01709                    00000130
                                                                    00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                              00000150
                                                                    00000160
   MODULE:          DICCFR              67                          00000170
                                                                    00000180
                                                                    00000190
                    - MODULE ABSTRACT -                             00000200
                                                                    00000210
                                                                    00000220
      THIS SUBROUTINE PUTS THE 64 BYTE RECORD BUFFER BACK           00000230
   ON THE BUFFER QUEUE IN SYSCOM. IF THE QUEUE IS FULL A            00000240
   PANIC IS PERFORMED.                                              00000250

                                                                    00000260
                                                                    00000270
                                                                    00000280
36FM  GET ADDRESS OF SRB (64-BYTE) BUFFER QUEUE IN SYSCOM           00000290
36FM  PUT ADDRESS OF BUFFER ON THE QUEUE                            00000300
36FM  IF THE QUEUE WAS ALREADY FULL THEN                            00000310
54JI     PANIC                                                      00000320
      (ELSE)                                                        00000330
      ENDIF                                                         00000340
      RETURN                                                        00000350

                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
GET A RECORD BUFFER                                                 00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                 73-01709                    00000130
                                                                    00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                              00000150
                                                                    00000160
   MODULE:          DICCGR              66                          00000170
                                                                    00000180
                                                                    00000190
                    - MODULE ABSTRACT -                             00000200
                                                                    00000210
                                                                    00000220
      THIS SUBROUTINE GETS A 64 BYTE RECORD BUFFER FROM             00000230
   THE RECORD BUFFER QUEUE IN SYSCOM. IF THE QUEUE IS EMPTY         00000240
   A PANIC IS PERFORMED.                                            00000250

                                                                    00000260
                                                                    00000270
                                                                    00000280
36FM  GET ADDRESS OF SRB (64-BYTE) BUFFER QUEUE IN SYSCOM           00000290
36FM  DEQUEUE A BUFFER FROM THE QUEUE                               00000300
36FM  IF THE QUEUE WAS EMPTY THEN                                   00000310
54JI     PANIC                                                      00000320
      (ELSE)                                                        00000330
      ENDIF                                                         00000340
      RETURN                                                        00000350
```

```
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
FREE AN INTER-PROGRAM MESSAGE BUFFER                                00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                 73-01709                    00000130
                                                                    00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                              00000150
                                                                    00000160
   MODULE:          DICCFI              65                          00000170
                                                                    00000180
                                                                    00000190
                    - MODULE ABSTRACT -                             00000200
                                                                    00000210
                                                                    00000220
      THIS SUBROUTINE PUTS AN INTER-PROGRAM MESSAGE BUFFER          00000230
   BACK ON THE IPM QUEUE.  IF THE QUEUE IS FULL A PANIC IS          00000240
   PERFORMED.                                                       00000250

                                                                    00000260
                                                                    00000270
                                                                    00000280
09FM   GET ADDRESS OF IPM BUFFER QUEUE IN SYSCOM                    00000290
09FM   PUT ADDRESS OF BUFFER ON THE QUEUE                           00000300
09FM   IF THE QUEUE WAS ALREADY FULL THEN                           00000310
543I      PANIC                                                     00000320
       (ELSE)                                                       00000330
       ENDIF                                                        00000340
       RETURN                                                       00000350


                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
GET AN INTER-PROGRAM MESSAGE BUFFER                                 00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                 73-01709                    00000130
                                                                    00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                              00000150
                                                                    00000160
   MODULE:          DICCGI              64                          00000170
                                                                    00000180
                                                                    00000190
                    - MODULE ABSTRACT -                             00000200
                                                                    00000210
                                                                    00000220
      THIS SUBROUTINE DEQUEUES AN INTER-PROGRAM MESSAGE             00000230
   BUFFER FROM THE IPM QUEUE.  IF THE QUEUE IS EMPTY A              00000240
   PANIC IS PERFORMED.                                              00000250

                                                                    00000260
                                                                    00000270
                                                                    00000280
09FM   GET ADDRESS OF IPM BUFFER QUEUE IN SYSCOM                    00000290
09FM   DEQUEUE A BUFFER FROM THE QUEUE                             00000300
09FM   IF THE QUEUE WAS EMPTY THEN                                 00000310
543I      PANIC                                                     00000320
       (ELSE)                                                       00000330
       ENDIF                                                        00000340
       RETURN                                                       00000350
```

```
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
WRITE DISKETTE BATCH/TRANSACTION RECORD ( WRITE.BATCH )                 00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                    76-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                                  00000150
                                                                        00000160
   MODULE:          DICCWB                 75                          00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
                                                                        00000220
         CALL THE DISKETTE SUBROUTINE TO WRITE A BATCH OR               00000230
      TRANSACTION RECORD.                                               00000240
         THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL                  00000250
      WHETHER DISKETTE ACCESS WAS SUCCESSFUL OR NOT.                    00000260

   CALL WRITE DISKETTE DATA BLOCK                                       00000270
   CASENTRY (DOK, DIO, DCC, ? )                                         00000280
      CASE DOK ( SUCCESSFUL )                                           00000290
100I     SET OK                                                        00000300
      CASE DIO ( I/O ERROR OCCURRED )                                   00000310
         ENQUEUE USER TERMINAL                                          00000320
023I     DISPLAY "DISKETTE I/O ERROR OCCURRED"                          00000330
140I     UPDATE TRANSACTION COUNT BEYOND MAXIMUM                        00000340
100I     SET BATCH ABORT                                               00000350
100I     CLEAR OK                                                      00000360
      CASE DCC ( INVALID CALL CODE )                                    00000370
543I     PANIC                                                         00000380
      CASE  ?  ( INVALID COMPLETION CODE )                              00000390
543I     PANIC                                                         00000400
   ENDCASE                                                              00000410
   RETURN                                                               00000420


                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
ASSIGN TRANSACTION STATUS COMMAND                                       00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                    76-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                                  00000150
                                                                        00000160
   MODULE:          DIATS                  28                          00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
                                                                        00000220
   THIS MODULE REQUESTS A BATCH ID. THE FILE MANAGER IS CALLED          00000230
TO READ THE BATCH FILE INFORMATION. IF THE BATCH INFORMATION IS         00000240
FOUND, THE USER IS ASKED FOR START AND END RANGE OF TRANSACTION         00000250
ID'S. EACH IS CHECKED FOR NON-ZERO VALUE,                               00000260
AND THE END NUMBER GREATER THAN THE START NUMBER. IF TRANS-             00000270
ACTION VALUES ARE ACCEPTABLE, THE USER IS ASKED FOR THE NEW             00000280
TRANSACTION STATUS.                                                     00000290
THE FILE MANAGER IS CALLED TO CHANGE TRANSACTION RECORD STATUS.         00000300
```

```
        ENQJEJE USER TERMINAL                                                   00000310
04JI    DISPLAY ON USER TERMINAL "ASSIGN TRANSACTION STATUS COMMAND"             00000320
48JI    DEQJEUE AN IPM BUFFER (GET.IPM)                                         00000330
49JI    DEQJEUE RECORD BUFFER (GET.SRB)                                         00000340
03JI    ASK USER FOR BATCH ID                                                   00000350
12JI                                                                            00000360
08AL    INITIALIZE PANIC CODE                                                   00000370
40JI                                                                            00000380
16JI    ACCESS BATCH FILE (GET.BATCH)                                           00000390
01FM                                                                            00000400
10JI    IF BATCH RECORD FOUND (OK SET), THEN                                    00000410
43JI    !  DISPLAY BATCH HEADER                                                 00000420
43JI    !  DISPLAY BATCH INFORMATION                                            00000430
03JI    !  ASK USER FOR START TRANSACTION NUMBER                                00000440
40JI    !                                                                       00000450
40JI    !  IF INPUT TRANS ID WITHIN RANGE THEN                                  00000460
03JI    !  !  ASK USER FOR END TRANSACTION NUMBER                               00000470
40JI    !  !  IF INPJT TRANS ID WITHIN RANGE AND GT. OR EQ. START ID. --        00000480
        !  !  -- THEN                                                          00000490
03JI    !  !  !  DO UNTIL VALID STATJS RECEIVED                                00000500
40JI    !  !  !                                                                00000510
16FM    !  !  !    VERIFY STATUS CODE BY RANGE CHECK                           00000520
40JI    !  !  !  ENDDO                                                         00000530
        !  !  !  ASK USER IF ALL PARAMETERS CORRECT                            00000540
        !  !  !  IF YES THEN                                                   00000550
08AL    !  !  !  !  INITIALIZE PANIC CODE                                      00000560
        !  !  !  !  DISPLAY "PROCESSING STATUS CHANGES"                        00000570
40JI    !  !  !  !                                                             00000580
40JI    !  !  !  !  DOUNTIL START TRANS ID GREATER THAN END TRANS ID --       00000590
        !  !  !  !  -- OR ERROR FOUND                                          00000600
        !  !  !  !     DELAY                                                   00000610
16JI    !  !  !  !     JPDATE IPM FOR CALL                                     00000620
08FM    !  !  !  !     CALL FILE MANAGER TO CHANGE TRANSACTION STATUS          00000630
16FM    !  !  !  !     CASENTRY (FOK, FNB, FTN, FCC, FIS, ? )                 00000640
        !  !  !  !       - CASE FOK ( SUCCESSFUL )                             00000650
        !  !  !  !         CASE FNB ( TRANSACTION NOT IN BATCH )               00000660
02JI    !  !  !  !            DISPLAY "TRANS. NOT IN BATCH"                    00000670
        !  !  !  !            SET ERROR FOUND                                  00000680
14JI    !  !  !  !            SUB 1 FROM TRANS. ID                             00000690
        !  !  !  !         CASE FTN ( TRANSACTION NOT ON FILE )                00000700
        !  !  !  !            DISPLAY "TRANS. NOT ON FILE"                     00000710
        !  !  !  !            SET ERROR FOUND                                  00000720
14JI    !  !  !  !            SUB 1 FROM TRANS. ID                             00000730
        !  !  !  !         CASE FCC ( INVALID CALL CODE )                      00000740
54JI    !  !  !  !            PANIC                                            00000750
        !  !  !  !         CASE FIS ( INVALID STATJS)                          00000760
54JI    !  !  !  !            PANIC                                            00000770
        !  !  !  !         CASE  ?  ( INVALID COMPLETION CODE )                00000780
54JI    !  !  !  !            PANIC                                            00000790
        !  !  !  !     ENDCASE                                                 00000800
14JI    !  !  !  !     INCREMENT TRANSACTION ID                                00000810
        !  !  !  !  ENDDO                                                      00000820
        !  !  !  !  IF ANY TRANSACTIONS CHANGED THEN                           00000830
        !  !  !  !     DISPLAY TRANS. THAT WHERE CHANGED                       00000840
        !  !  !  !  ELSE                                                       00000850
        !  !  !  !     DISPLAY "NO CHANGE MADE"                                00000860
        !  !  !  !  ENDIF                                                      00000870
        !  !  !  (ELSE)                                                        00000880
        !  !  !  ENDIF                                                         00000890
        !  !  ELSE                                                           00000900
        !  !  !  IF END ID LESS THAN START ID THEN                           00000910
        !  !  !     DISPLAY "END ID MUST BE GREATER THAN OR --               00000920
        !  !  !     -- EQUAL TO THE START ID                                 00000930
        !  !  !  ELSE                                                        00000940
        !  !  !     DISPLAY "INVALID TRANS. ID"                              00000950
        !  !  !  ENDIF                                                       00000960
        !  !  ENDIF                                                          00000970
        !  ELSE                                                            00000980
        !  !  DISPLAY "INVALID TRANS. ID"                                    00000990
        !  ENDIF                                                           00001000
        !  DISPLAY "STATJS CHANGE COMMAND ENDED"                           00001010
        ELSE                                                               00001020
02JI    !  DISPLAY BATCH ID, "BATCH NOT ON FILE"                           00001030
```

```
        ENDIF                                                           00001040
510I    RELEASE RECORD BUFFER (FREE.SRB)                                00001050
500I    RELEASE THE IPM BUFFER (FREE.IPM)                               00001060
        DEQUEUE TERMINAL                                                00001070
        ENDPROG                                                         00001080
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
        RECIRC. ON SCANNER ERROR                                        00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
           PROJECT:     DAS                 73-01709                    00000130
                                                                        00000140
           SUB-SYSTEM:  OPERATOR INTERFACE                              00000150
                                                                        00000160
           MODULE:      DIREC               45                          00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
           THIS MODULE SETS A SWITCH VALUE IN SYSCOM WHICH WHEN         00000230
        INTERROGATED BY SCANNER INPUT SOFTWARE, SCANNER ERRORS WILL     00000240
        BE RECIRCULATED.                                                00000250
                                                                        00000260
                                                                        00000270
                                                                        00000280
        ENQUEUE USER TERMINAL                                           00000290
040I    DISPLAY ON USER TERMINAL "RECIRC. ON SCANNER ERROR"             00000300
11SC    SET SYSCOM FLAG VALUE                                           00000310
02AL    GET AN ALARM BUFFER                                             00000320
02AL    IF ONE RECEIVED, THEN                                           00000330
03AL       PUT IN MESSAGE NUMBER                                        00000340
01AL       PUT ADDRESS OF THE BUFFER IN THE ALARM QUEUE                 00000350
01AL       IF THERE IS NO ROOM, THEN                                    00000360
07AL          PANIC                                                     00000370
           (ELSE)                                                       00000380
           ENDIF                                                        00000390
        ELSE                                                            00000400
07AL       PANIC                                                        00000410
        ENDIF                                                           00000420
        DEQUEUE TERMINAL                                                00000430
        ENDPROG                                                         00000440
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
        GET DISK ORDER FILE RECORD BY ORDER ID ( GET.ORDER )            00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
           PROJECT:     DAS                 73-01709                    00000130
                                                                        00000140
           SUB-SYSTEM:  OPERATOR INTERFACE                              00000150
                                                                        00000160
           MODULE:      DICCGO              77                          00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
        PROCESS:    CALL THE FILE MANAGER TO READ THE DISK ORDER        00000230
                 FILE RECORD. IF SUCCESSFUL, COPY THE ORDER INFORM-     00000240
```

```
            ATION FROM THE SYSCOM RECORD BUFFER TO THE DATA AREAS   00000250
            IN THE PROGRAM AND SET OK. IF AN ERROR OCCURRED, TELL   00000260
            THE USER, CLEAR OK AND TAKE ANY FURTHER APPRORIATE      00000270
            ACTION REQUIRED.                                        00000280
               THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL TO THE00000290
            FILE MANAGER WHETHER ORDER FILE ACCESS WAS SUCCESFUL    00000300
            OR NOT.                                                 00000310
22FM    UPDATE IPM FOR CALL                                         00000320
08FM    CALL FILE MANAGER TO READ ORDER FILE RECORD BY ORDER ID      00000330
22FM    CASENTRY (FOK, FBN, FCC, FON, ? )                           00000340
           CASE FOK ( SUCCESFUL )                                   00000350
05FM          GET ORDER INFORMATION                                 00000360
13DI                                                                00000370
10DI          SET OK                                                00000380
           CASE FBN ( BATCH NOT FOUND )                             00000390
02DI          PRINT/DISPLAY BATCH ID, "BATCH NO LONGER ON FILE"     00000400
10DI          CLEAR OK                                              00000410
           CASE FCC ( INVALID CALL CODE )                           00000420
54DI          PANIC                                                 00000430
           CASE FON ( ORDER NOT ON FILE)                            00000440
02DI          PRINT/DISPLAY ORDER ID, "ORDER ID NOT ON FILE"        00000450
10DI          CLEAR OK                                              00000460
           CASE  ?  ( INVALID COMPLETION CODE )                     00000470
54DI          PANIC                                                 00000480
        ENDCASE                                                     00000490
        RETURN                                                      00000500
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
        FIND RELATIVE DISK ORDER FILE RECORD IN BATCH ( FIND.ORDER ) 00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                        - MODULE HISTORY -                          00000100
                                                                    00000110
                                                                    00000120
           PROJECT:        DAS                  73-01709            00000130
                                                                    00000140
           SUB-SYSTEM:     OPERATOR INTERFACE                       00000150
                                                                    00000160
           MODULE:         DICCFD               76                  00000170
                                                                    00000180
                                                                    00000190
                        - MODULE ABSTRACT -                         00000200
                                                                    00000210
                                                                    00000220
              EACH RELATIVE ORDER RECORD IS SEQUENTIALLY ACCESSED   00000230
           ONE AT A TIME BY CALLING THE FILE MANAGER FOR EACH       00000240
           RELATIVE ORDER NUMBER. IF SUCCESSFUL, COPY THE ORDER     00000250
           INFORMATION FROM THE SYSCOM RECORD BUFFER TO THE DATA    00000260
           AREAS IN THE PROGRAM AND SET OK. IF AN ERROR OCCURRED,   00000270
           TELL THE USER, CLEAR OK AND TAKE ANY FUTHER APPRO-       00000280
           PRIATE ACTION REQUIRED.                                  00000290
              THIS SUBROUTINE WILL EXIT WHEN EITHER:                00000300
           1) THE DISK ORDER FILE HAS BEEN ACCESSED WITHOUT         00000310
              ERROR, OR                                             00000320
           2) THE DISK ORDER FILE HAS BEEN COMPLETELY ACCESSED.     00000330

13DI    DOUNTIL END OF ORDER COUNT OR OK SET                        00000340
12DI                                                                00000350
10DI                                                                00000360
23FM       UPDATE IPM FOR CALL                                      00000370
08FM       CALL FILE MANAGER TO READ ORDER FILE RECORD BY ORDER COUNT 00000380
23FM       CASENTRY (FOK, FBN, FCC, FIR, ? )                        00000390
              CASE FOK ( SUCCESFUL )                                00000400
05FM             GET ORDER INFORMATION                              00000410
13DI                                                                00000420
10DI             SET OK                                             00000430
              CASE FBN ( BATCH NOT FOUND )                          00000440
13DI             UPDATE ORDER COUNT BEYOND MAXIMUM                  00000450
12DI                                                                00000460
```

```
103I            CLEAR OK                                                00000470
              CASE FCC ( INVALID CALL CODE )                            00000480
543I             PANIC                                                  00000490
              CASE FIR ( INVALID RELATIVE RECORD )                      00000500
543I             PANIC                                                  00000510
              CASE  ?  ( INVALID COMPLETION CODE )                      00000520
543I             PANIC                                                  00000530
            ENDCASE                                                     00000540
103I        INCREMENT ORDER COUNT                                       00000550
          ENDDO                                                         00000560
          RETURN                                                        00000570
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
          CHANGE DISK BATCH FILE RECORD STATUS ( CHANGE.BATCH )         00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                          - MODULE HISTORY -                            00000100
                                                                        00000110
                                                                        00000120
             PROJECT:        DAS                  78-01709            00000130
                                                                        00000140
             SUB-SYSTEM:     OPERATOR INTERFACE                         00000150
                                                                        00000160
             MODULE:         DICCCB               72                    00000170
                                                                        00000180
                                                                        00000190
                          - MODULE ABSTRACT -                           00000200
                                                                        00000210
                                                                        00000220
                CALL THE FILE MANAGER TO CHANGE THE DISK BATCH          00000230
              FILE RECORD.                                              00000240
                IF AN ERROR OCCURRED, TELL THE USER, CLEAR OK AND       00000250
              TAKE ANY FURTHER APPROPRIATE ACTION REQUIRED.             00000260
                THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL TO THE    00000270
              FILE MANAGER WHETHER BATCH FILE ACCESS WAS SUCCESSFUL     00000280
              OR NOT.                                                   00000290
15FM      UPDATE IPM FOR CALL                                           00000300
08FM      CALL FILE MANAGER TO CHANGE BATCH STATUS                      00000310
15FM      CASENTRY (FOK, FBN, FCC, FIS, ? )                             00000320
            CASE FOK ( SUCCESSFUL )                                     00000330
103I           SET OK                                                   00000340
            CASE FBN ( BATCH NOT FOUND )                                00000350
               OUTPUT BATCH NOT ON FILE MESSAGE                         00000360
               CLEAR OK                                                 00000370
            CASE FCC ( INVALID CALL CODE )                              00000380
543I           PANIC                                                    00000390
            CASE FIS ( INVALID STATUS)                                  00000400
543I           PANIC                                                    00000410
            CASE  ?  ( INVALID COMPLETION CODE )                        00000420
543I           PANIC                                                    00000430
          ENDCASE                                                       00000440
          RETURN                                                        00000450
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
          READ DISK BATCH FILE RECORD BY BATCH ID ( GET.BATCH )         00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                          - MODULE HISTORY -                            00000100
                                                                        00000110
                                                                        00000120
             PROJECT:        DAS                  78-01709            00000130
                                                                        00000140
             SUB-SYSTEM:     OPERATOR INTERFACE                         00000150
                                                                        00000160
```

```
      MODULE:        DICCGB              71                          00000170
                                                                     00000180
                                                                     00000190
                     - MODULE ABSTRACT -                             00000200
                                                                     00000210
                                                                     00000220
           CALL THE FILE MANAGER TO READ THE DISK BATCH FILE         00000230
         RECORD. IF SUCCESSFUL, COPY THE BATCH INFORMATION FROM      00000240
         THE SYSCOM RECORD BUFFER TO THE DATA AREAS IN THE PRO-      00000250
         GRAM AND SET OK. IF AN ERROR OCCURRED, TELL THE USER,       00000260
         CLEAR OK AND TAKE ANY FURTHER APPROPRIATE ACTION RE-        00000270
         QUIRED.                                                     00000280
           THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL TO THE      00000290
         FILE MANAGER WHETHER BATCH FILE ACCESS WAS SUCCESSFUL       00000300
         OR NOT.                                                     00000310
18FM  UPDATE IPM FOR CALL                                          00000320
08FM  CALL FILE MANAGER TO READ BATCH FILE RECORD BY BATCH ID      00000330
18FM  CASENTRY (FOK, FBN, FCC, ? )                                 00000340
        CASE FOK ( SUCCESSFUL )                                    00000350
01FM      GET BATCH INFORMATION                                    00000360
12DI                                                               00000370
10DI      SET OK                                                   00000380
        CASE FBN ( BATCH NOT FOUND )                               00000390
10DI      CLEAR OK                                                 00000400
        CASE FCC ( INVALID CALL CODE )                             00000410
54DI      PANIC                                                    00000420
        CASE  ?  ( INVALID COMPLETION CODE )                       00000430
54DI      PANIC                                                    00000440
      ENDCASE                                                      00000450
      RETURN                                                       00000460
                                                                   00000010
                                                                   00000020
                                                                   00000030
                                                                   00000040
                                                                   00000050
      FIND CURRENT BATCH IN SYSCOM BATCH DIRECTORY ( FIND.BATCH )  00000060
                                                                   00000070
                                                                   00000080
                                                                   00000090
                     - MODULE HISTORY -                            00000100
                                                                   00000110
                                                                   00000120
        PROJECT:       DAS                 73-01709                00000130
                                                                   00000140
        SUB-SYSTEM:    OPERATOR INTERFACE                          00000150
                                                                   00000160
        MODULE:        DICCFB              70                      00000170
                                                                   00000180
                                                                   00000190
                     - MODULE ABSTRACT -                           00000200
                                                                   00000210
                                                                   00000220
           EACH SYSCOM BATCH DIRECTORY IS EXAMINED ONE AT          00000230
         AT A TIME. IF THE ENTRY CONTAINS A PENDING, ACTIVE        00000240
         OR COMPLETE STATUS; THE DISK BATCH FILE WILL BE           00000250
         ACCESSED.                                                 00000260
           THIS SUBROUTINE WILL EXIT WHEN EITHER:                  00000270
         1) THE DISK BATCH FILE HAS BEEN ACCESSED WITHOUT          00000280
            ERROR, OR                                              00000290
         2) THE BATCH DIRECTORY HAS BEEN COMPLETELY EXAMINED.      00000300
12DI  DOUNTIL BATCH COUNT GREATER THAN MAXIMUM OR OK SET            00000310
10DI                                                               00000320
02FM    EXAMINE SYSCOM BATCH DIRECTORY ENTRY                       00000330
02FM    IF BATCH PENDING, ACTIVE OR COMPLETE, THEN                 00000340
16DI      ACCESS BATCH FILE (GET.BATCH)                            00000350
        ELSE                                                       00000360
10DI      CLEAR OK                                                 00000370
        ENDIF                                                      00000380
12DI    INCREMENT POINTER TO SYSCOM BATCH DIRECTORY                 00000390
12DI    INCREMENT BATCH COUNT                                      00000400
      ENDDO                                                        00000410
      RETURN                                                       00000420
```

```
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
READ DISK TRANSACTION FILE RECORD BY TRANS. ID ( GET.TRANS )        00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                  7G-01709                   00000130
                                                                    00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                              00000150
                                                                    00000160
   MODULE:          OICCGT               80                         00000170
                                                                    00000180
                                                                    00000190
                    - MODULE ABSTRACT -                             00000200
                                                                    00000210
                                                                    00000220
      CALL THE FILE MANAGER TO READ THE TRANSACTION FILE            00000230
   RECORD. IF SUCCESSFUL, COPY THE TRANSACTION DATA FROM            00000240
   THE SYSCOM RECORD BUFFER TO THE DATA AREAS IN THE PRO-           00000250
   GRAM AND SET OK. IF AN ERROR OCCURRED, TELL THE USER,            00000260
   CLEAR OK AND TAKE ANY FURTHER APPROPRIATE ACTION RE-             00000270
   QUIRED.                                                          00000280
      THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL TO THE          00000290
   FILE MANAGER WHETHER TRANSACTION FILE ACCESS WAS SUC-            00000300
   CESSFUL OR NOT.                                                  00000310
19FM  UPDATE IPM FOR CALL                                           00000320
08FM  CALL FILE MANAGER TO READ TRANSACTION FILE RECORD             00000330
19FM  CASENTRY (FOK, FTN, FCC, ? )                                  00000340
         CASE FOK ( SUCCESSFUL )                                    00000350
04FM        GET TRANSACTION INFORMATION                             00000360
14DI                                                                00000370
10DI        SET OK                                                  00000380
         CASE FTN ( TRANSACTION NOT FOUND )                         00000390
10DI        CLEAR OK                                                00000400
         CASE FCC ( INVALID CALL CODE )                             00000410
54DI        PANIC                                                   00000420
         CASE  ?  ( INVALID COMPLETION CODE )                       00000430
54DI        PANIC                                                   00000440
      ENDCASE                                                       00000450
      RETURN                                                        00000460
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
HELP COMMAND (OPERATOR COMMAND LIST DISPLAY)                        00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                  7G-01709                   00000130
                                                                    00000140
   SUB-SYSTEM:      OPERATOR INTERFACE                              00000150
                                                                    00000160
   MODULE:          OIHLP                63                         00000170
                                                                    00000180
                                                                    00000190
                    - MODULE ABSTRACT -                             00000200
                                                                    00000210
                                                                    00000220
   THIS MODULE DISPLAYS ALL OPERATOR INTERFACE COMMANDS ON THE      00000230
USER TERMINAL.                                                      00000240
```

```
                                                                    00000250
                                                                    00000260
                                                                    00000270
        ENQUEUE USER TERMINAL                                       00000280
04DI    DISPLAY ON USER TERMINAL "HELP COMMAND"                     00000290
47DI    DISPLAY OPERATOR COMMAND LIST ON USER TERMINAL              00000300
        DEQUEUE TERMINAL                                            00000310
        ENDPROG                                                     00000320
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
        ADD DISK BATCH FILE RECORD ( ADD.BATCH )                    00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                        - MODULE HISTORY -                          00000100
                                                                    00000110
                                                                    00000120
           PROJECT:     DAS                 73-01709               00000130
                                                                    00000140
           SUB-SYSTEM:  OPERATOR INTERFACE                          00000150
                                                                    00000160
           MODULE:      DICCAB              74                     00000170
                                                                    00000180
                                                                    00000190
                        - MODULE ABSTRACT -                         00000200
                                                                    00000210
                                                                    00000220
              CALL THE FILE MANAGER TO ADD THE DISKETTE BATCH       00000230
           RECORD.                                                  00000240
              THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL TO THE  00000250
           FILE MANAGER WHETHER BATCH FILE ACCESS WAS SUCCESSFUL    00000260
           OR NOT.                                                  00000270

12FM    UPDATE IPM FOR CALL                                         00000280
08FM    CALL FILE MANAGER TO ADD BATCH FILE RECORD                  00000290
12FM    CASENTRY (FOK, FBX, FBA, FAD, FCC, ? )                     00000300
           CASE FOK ( SUCCESSFUL )                                  00000310
10DI          SET OK                                               00000320
           CASE FBX ( BATCH FILE SPACE EXHAUSTED )                  00000330
02DI          DISPLAY "BATCH FILE SPACE EXHAUSTED"                 00000340
10DI          CLEAR OK                                             00000350
           CASE FBA ( BATCH ALREADY ON FILE )                       00000360
02DI          DISPLAY BATCH ID, "BATCH ALREADY ON FILE"            00000370
10DI          CLEAR OK                                             00000380
           CASE FAD ( BATCH ALREADY BEING ADDED )                   00000390
54DI          PANIC                                                00000400
           CASE FCC ( INVALID CALL CODE )                           00000410
54DI          PANIC                                                00000420
           CASE  ?  ( INVALID COMPLETION CODE )                     00000430
54DI          PANIC                                                00000440
        ENDCASE                                                     00000450
        RETURN                                                      00000460

                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
        RESTORE SHIPPING LINE COMMAND                               00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                        - MODULE HISTORY -                          00000100
                                                                    00000110
                                                                    00000120
           PROJECT:     DAS                 73-01709               00000130
                                                                    00000140
           SUB-SYSTEM:  OPERATOR INTERFACE                          00000150
                                                                    00000160
           MODULE:      DIRSL               26                     00000170
```

```
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
      THIS MODULE REQUESTS SORT LINE NUMBER TO BE RESTORED.             00000230
   THE SORT LINE NUMBER IS VALIDATED BY RANGE CHECK. IF THE             00000240
   SORT LINE NUMBER IS ACCEPTABLE, THE RESIDENT SORT TRANSLA-           00000250
   TION TABLE IS UPDATED AND THE FILE MANAGER IS CALLED TO WRITE        00000260
   THE RESIDENT SORT TRANSLATION TABLE TO DISK.                         00000270
   ENQUEUE USER TERMINAL                                                00000280
040I DISPLAY ON USER TERMINAL "RESTORE SORT LINE COMMAND"              00000290
030I ASK USER FOR OLD SORT LINE NUMBER                                 00000300
380I                                                                   00000310
380I RANGE CHECK OLD SORT LINE NUMBER                                  00000320
39FM IF NOT IN RANGE OR RECIRC OR ERROR CHUTE, THEN                    00000330
020I    DISPLAY "INVALID SORT LINE NUMBER" OLD LINE NUMBER             00000340
        DISPLAY "COMMAND ABORTED"                                      00000350
   ELSE                                                                00000360
        IF SORT LINE HAS BEEN REROUTED, THEN                           00000370
380I       UPDATE RESIDENT SORT TRANSLATION TABLE                      00000380
040I       DISPLAY "SORT LINE ASSIGNED"                                00000390
08AL       INITIALIZE PANIC CODE                                       00000400
380I                                                                   00000410
460I       WRITE SORT TRANSLATION TABLE TO DISK (WRITE.SORT)           00000420
        ELSE                                                           00000430
020I       DISPLAY "NO PRIOR ASSIGNMENT OF SORT LINE"                  00000440
           DISPLAY "COMMAND ABORTED"                                   00000450
        ENDIF                                                          00000460
   ENDIF                                                               00000470
   DEQUEUE TERMINAL                                                    00000480
   ENDPROG                                                             00000490

                                                                       00000010
                                                                       00000020
                                                                       00000030
                                                                       00000040
                                                                       00000050
   ASSIGN SHIPPING LINE COMMAND                                        00000060
                                                                       00000070
                                                                       00000080
                                                                       00000090
                        - MODULE HISTORY -                             00000100
                                                                       00000110
                                                                       00000120
      PROJECT:          DAS                 76-01709                 00000130
                                                                       00000140
      SUB-SYSTEM:       OPERATOR INTERFACE                           00000150
                                                                       00000160
      MODULE:           OIASL               25                       00000170
                                                                       00000180
                                                                       00000190
                        - MODULE ABSTRACT -                            00000200
                                                                       00000210
                                                                       00000220
      THIS MODULE REQUESTS SORT LINE NUMBER TO BE (RE)ASSIGNED.        00000230
   THE SORT LINE NUMBER IS VALIDATED BY RANGE CHECK. IF THE OLD        00000240
   SORT LINE NUMBER IS ACCEPTABLE, THE USER IS ASKED FOR THE NEW       00000250
   SORT LINE NUMBER. THIS ALSO IS VALIDATED BY RANGE CHECK. IF THE     00000260
   NEW SORT LINE NUMBER IS ACCEPTABLE, THE RESIDENT SORT TRANSLA-      00000270
   TION TABLE IS UPDATED AND THE FILE MANAGER IS CALLED TO WRITE       00000280
   THE RESIDENT SORT TRANSLATION TABLE TO DISK.                        00000290

   ENQUEUE USER TERMINAL                                               00000300
   CLEAR ABORT                                                         00000310
39FM GET SORT SYSTEM PARAMETERS                                        00000320
040I DISPLAY ON USER TERMINAL "REROUTE SORT LINE COMMAND"              00000330
030I ASK USER FOR OLD SORT LINE NUMBER                                 00000340
370I                                                                   00000350
370I RANGE CHECK OLD SORT LINE NUMBER                                  00000360
   IF NOT IN RANGE OR RECIRC OR ERROR CHUTE, THEN                      00000370
020I    DISPLAY "INVALID SORT LINE NUMBER" OLD LINE NUMBER             00000380
        SET ABORT                                                      00000390
```

```
        ELSE                                                                    00000400
06FM       GET ENTRY FROM SORT TRANSLATION TABLE                                00000410
           IF ALREADY RE-ASSIGNED, THEN                                         00000420
              TELL USER IT IS ALREADY ASSIGNED                                  00000430
              ASK IF WE SHOULD CONTINUE                                         00000440
              IF THE ANSWER IS "NO", THEN                                       00000450
                 SET ABORT                                                      00000460
              (ELSE)                                                            00000470
              ENDIF                                                             00000480
           (ELSE)                                                               00000490
           ENDIF                                                                00000500
           IF ABORT IS NOT SET, THEN                                            00000510
03JI          ASK USER FOR NEW SORT LINE NUMBER                                 00000520
37JI                                                                            00000530
37JI          RANGE CHECK NEW SORT LINE NUMBER                                  00000540
              IF NOT IN RANGE                                                   00000550
                 SET ABORT                                                      00000560
02JI             DISPLAY "INVALID SORT LINE NUMBER" NEW LINE NUMBER             00000570
              ELSE                                                              00000580
06FM             IF OLD SORT LINE EQ NEW SORT LINE, THEN                        00000590
02JI                DISPLAY "ATTEMPT TO ASSIGN SORT LINE TO ITSELF"             00000600
                    DISPLAY "COMMAND ABORTED"                                   00000610
                 ELSE                                                           00000620
37JI                UPDATE RESIDENT SORT TRANSLATION TABLE                      00000630
04JI                DISPLAY "SORT LINE ASSIGNED"                                00000640
08AL                INITIALIZE PANIC CODE                                       00000650
37JI                                                                            00000660
46JI                WRITE SORT TRANSLATION TABLE TO DISK (WRITE.SORT)           00000670
              ENDIF                                                             00000680
           ENDIF                                                                00000690
        ENDIF                                                                   00000700
        ELSE                                                                    00000710
           DISPLAY "COMMAND ABORTED"                                            00000720
        ENDIF                                                                   00000730
        DEQUEUE TERMINAL                                                        00000740
        ENDPROG                                                                 00000750

                                                                                00000010
                                                                                00000020
                                                                                00000030
                                                                                00000040
                                                                                00000050
        OPERATOR INTERFACE CONTROL ( ATTENTION LIST )                           00000060
                                                                                00000070
                                                                                00000080
                                                                                00000090
                          - MODULE HISTORY -                                    00000100
                                                                                00000110
                                                                                00000120
           PROJECT:       DAS                    78-01709                      00000130
                                                                                00000140
           SUB-SYSTEM:    OPERATOR INTERFACE                                   00000150
                                                                                00000160
           MODULE:        OICTL                  14                            00000170
                                                                                00000180
                                                                                00000190
                       -MODULE ABSTRACT-                                        00000200
                                                                                00000210
                                                                                00000220
           THIS MODULE CONTAINS THE ATTENTION LIST PROCESSING. THE              00000230
        USER WILL HIT THE ATTENTION KEY, THEN TYPE THE OPERATOR COM-            00000240
        MAND FOLLOWED BY THE RETURN KEY. IN THE ATTENTION LIST PRO-             00000250
        CESSING THE PROGRAM WILL CALL A SUBROUTINE (TASK SELECT) TO             00000260
        SELECT ONE OF FOUR LOADING TASKS, DEPENDING ON WHICH ONES               00000270
        ARE NOT ACTIVE AT THE TIME.  THE NUMBER ASSOCIATED WITH THE             00000280
        COMMAND TYPED WILL BE PASSED TO THE LOADING TASK.  THEN AN              00000290
        END ATTENTION IS PERFORMED.                                             00000300
        THE ATTACHED TASK WILL THEN ENQUEUE A LOADING SUBROUTINE                00000310
        (LOADIT) AND PASS THE COMMAND NUMBER.  THE SUBROUTINE WILL              00000320
        THEN LOAD THE REQUIRED PROGRAM.                                         00000330

        DOUNTIL  FOREVER                                                        00000340
26JI       WAIT FOR USER TO ENTER COMMAND                                       00000350
```

```
01JI      CASENTRY  (COMMAND-1,COMMAND-2,...,COMMAND-N)                    00000360
             CASE COMMAND-1                                                00000370
                CALL TASK SELECT AND PASS COMMAND NUMBER                   00000380
             CASE COMMAND-2                                                00000390
                CALL TASK SELECT AND PASS COMMAND NUMBER                   00000400
              :                                                            00000410
              :                                                            00000420
             CASE COMMAND-N                                                00000430
                CALL TASK SELECT AND PASS COMMAND NUMBER                   00000440
          ENDCASE                                                          00000450
       ENDDO                                                               00000460
       ENDPROG                                                             00000470
                                                                           00000480
                        ****************                                   00000490
                                                                           00000500
        * FORMAT OF SUBROUTINE TO ATTACH LOADING TASK *                     00000510
                                                                           00000520
                                                                           00000530
       IF FIRST LOADING TASK IS NOT ACTIVE, THEN                           00000540
          PASS COMMAND NUMBER AND ATTACH THE TASK                          00000550
       ELSE                                                                00000560
          IF SECOND LOADING TASK IS NOT ACTIVE, THEN                       00000570
             PASS COMMAND NUMBER AND ATTACH THE TASK                       00000580
          ELSE                                                             00000590
             IF THIRD LOADING TASK IS NOT ACTIVE, THEN                     00000600
                PASS COMMAND NUMBER AND ATTACH THE TASK                    00000610
             ELSE                                                          00000620
                PASS COMMAND NUMBER AND ATTACH THE FOURTH TASK             00000630
             ENDIF                                                         00000640
          ENDIF                                                            00000650
       ENDIF                                                               00000660
       RETURN                                                              00000670
                                                                           00000680
                        ****************                                   00000690
                                                                           00000700
       * GENERAL FORMAT OF TASKS TO CALL LOADING SUBROUTINE. *             00000710
                                                                           00000720
       ENQUEUE USER TERMINAL                                               00000730
       ENQUEUE SUBROUTINE (LOAD IT)                                        00000740
       CALL SUBROUTINE (LOAD IT) AND PASS COMMAND NUMBER                   00000750
       DEQUEUE SUBROUTINE                                                  00000760
       DEQUEUE USER TERMINAL                                               00000770
       ENDTASK                                                             00000780
                                                                           00000790
                        ****************                                   00000800
                                                                           00000810
       * FORMAT OF LOADING SUBROUTINE (LOAD IT) *                          00000820
                                                                           00000830
       CASENTRY (COMMAND-?,COMMAND-1,COMMAND-2,...,COMMAND-N)              00000840
          CASE COMMAND-?                                                   00000850
07AL         PANIC                                                         00000860
          CASE COMMAND-1                                                   00000870
             LOAD COMMAND-1 PROGRAM                                        00000880
          CASE COMMAND-2                                                   00000890
             LOAD COMMAND-2 PROGRAM                                        00000900
          :                                                                00000910
          :                                                                00000920
                                                                           
          CASE COMMAND-N                                                   00000930
             LOAD COMMAND-N PROGRAM                                        00000940
       ENDCASE                                                             00000950
       GET LOAD CONDITION CODE                                             00000960
       IF CONDITION CODE NOT EQUAL TO 'NO ERRORS', THEN                    00000970
          IF CONDITION CODE NOT EQUAL TO 'NO ROOM FOR PROGRAM', AND,       00000980
             CONDITION CODE NOT EQUAL TO 'NO ROOM FOR LOADER', THEN        00000990
             DISPLAY '*01* PROGRAM LOAD ERROR'                             00001000
             DISPLAY ERROR CODE                                            00001010
          ELSE                                                             00001020
             DISPLAY '**** RETRY COMMAND LATER'                            00001030
          ENDIF                                                            00001040
       (ELSE)                                                              00001050
       ENDIF                                                               00001060
       RETURN                                                              00001070
```

```
                                                                        00001080
                                                                        00001090
NOTE - FOR COMMANDS 'READ BATCH' AND 'WRITE BATCH', THE DIS-            00001100
       KETTE IN USE FLAG MUST BE CHECKED BEFORE THE CORRES-             00001110
       PONDING PROGRAM IS LOADED.  IF THE FLAG IS ALREADY SET,          00001120
       'DISKETTE ALREADY IN USE' IS DISPLAYED.  OTHERWISE, IF           00001130
       THERE WERE NO LOAD ERRORS, THE FLAG IS THEN SET.                 00001140
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
HOLD ON SCANNER ERROR                                                   00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                - MODULE HISTORY -                                      00000100
                                                                        00000110
                                                                        00000120
  PROJECT:      DAS                 73-01709                          00000130
                                                                        00000140
  SUB-SYSTEM:   OPERATOR INTERFACE                                      00000150
                                                                        00000160
  MODULE:       OIHLT               46                                00000170
                                                                        00000180
                                                                        00000190
                - MODULE ABSTRACT -                                     00000200
                                                                        00000210
                                                                        00000220
  THIS MODULE SETS A SWITCH VALUE IN SYSCOM WHICH WHEN                  00000230
INTERROGATED BY SCANNER INPUT SOFTWARE, SCANNER ERRORS WILL             00000240
BE HALTED.                                                              00000250
                                                                        00000260
                                                                        00000270
                                                                        00000280
    ENQUEUE USER TERMINAL                                               00000290
04OI    DISPLAY ON USER TERMINAL "HOLD ON SCANNER ERROR"                 00000300
11SC    SET SYSCOM FLAG VALUE                                           00000310
02AL    GET AN ALARM BUFFER                                             00000320
02AL    IF ONE RECEIVED, THEN                                           00000330
03AL       PUT IN MESSAGE NUMBER                                        00000340
01AL       PUT ADDRESS OF BUFFER ON THE ALARM QUEUE                     00000350
01AL       IF NO ROOM, THEN                                             00000360
07AL          PANIC                                                     00000370
           (ELSE)                                                       00000380
           ENDIF                                                        00000390
    ELSE                                                                00000400
07AL       PANIC                                                        00000410
    ENDIF                                                               00000420
    DEQUEUE TERMINAL                                                    00000430
    ENDPROG                                                             00000440
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
OPEN PSC COMMUNICATIONS COMMAND                                         00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                - MODULE HISTORY -                                      00000100
                                                                        00000110
                                                                        00000120
  PROJECT:      DAS                 73-01709                          00000130
                                                                        00000140
  SUB-SYSTEM:   OPERATOR INTERFACE                                      00000150
                                                                        00000160
  MODULE:       OIPSC               27                                00000170
                                                                        00000180
                                                                        00000190
                - MODULE ABSTRACT -                                     00000200
```

```
                                                                        00000210
                                                                        00000220
      THIS MODULE INITIATES PSC COMMUNICATIONS BY ENQUEUING A           00000230
    RESTART CODE ON THE PSC COMMUNICATION QUEUE. THE USER IS            00000240
    NOTIFIED OF INITIATION. IF THE PSC COMMUNICATIONS ARE ALREADY       00000250
    OPEN THE USER IS TOLD AND THE COMMAND IS ABORTED.                   00000260

                                                                        00000270
                                                                        00000280
                                                                        00000290
    ENQUEUE USER TERMINAL                                               00000300
040I DISPLAY ON USER TERMINAL "OPEN PSC COMMUNICATIONS COMMAND"         00000310
07PS GET PSC COMMUNICATIONS FLAG FROM SYSCOM                            00000320
    IF THE FLAG EQ COMMUNICATIONS CLOSED, THEN                          00000330
01PS   DEQUEUE A PSC OUTPUT BUFFER                                      00000340
01PS   IF BUFFER FOUND, THEN                                            00000350
05PS     UPDATE PSC OUTPUT BUFFER                                       00000360
02PS     ENQUEUE PSC OUTPUT BUFFER ON PSC OUTPUT QUEUE                  00000370
02PS     IF PSC OUTPUT QUEUE FULL                                       00000380
540I       PANIC                                                        00000390
         (ELSE)                                                         00000400
         ENDIF                                                          00000410
       ELSE                                                             00000420
540I     PANIC                                                          00000430
       ENDIF                                                            00000440
040I   DISPLAY "PSC COMMUNICATION INITIATED"                           00000450
    ELSE                                                                00000460
040I   DISPLAY "PSC COMMUNICATIONS ALREADY OPEN"                       00000470
040I   DISPLAY "COMMAND ABORTED"                                        00000480
    ENDIF                                                               00000490
    DEQUEUE TERMINAL                                                    00000500
    ENDPROG                                                             00000510

                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
    ADD DISK TRANSACTION (ORDER) FILE RECORD ( ADD.TRANS )              00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
    PROJECT:        DAS                 78-01709                        00000130
                                                                        00000140
    SUB-SYSTEM:     OPERATOR INTERFACE                                  00000150
                                                                        00000160
    MODULE:         DICCAT              81                              00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
                                                                        00000220
        CALL THE FILE MANAGER TO ADD THE TRANSACTION FILE               00000230
      RECORD.                                                           00000240
        THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL TO THE            00000250
      FILE MANAGER WHETHER TRANSACTION FILE ACCESS WAS SUC-             00000260
      CESSFUL OR NOT.                                                   00000270

13FM UPDATE IPM FOR CALL                                                00000280
08FM CALL FILE MANAGER TO ADD TRANSACTION FILE RECORD                   00000290
13FM CASENTRY (FOK, FTA, FTX, FOX, FCC, ? )                             00000300
       CASE FOK ( SUCCESSFUL )                                          00000310
100I     SET OK                                                         00000320
       CASE FTA ( TRANSACTION ALREADY ON FILE )                         00000330
020I     PRINT/DISPLAY TRANS. ID, "TRANSACTION ALREADY ON FILE"         00000340
100I     CLEAR OK                                                       00000350
       CASE FTX ( TRANSACTION FILE SPACE EXHAUSTED )                    00000360
020I     PRINT/DISPLAY "TRANSACTION FILE SPACE EXHAUSTED"               00000370
100I     CLEAR OK                                                       00000380
       CASE FOX ( ORDER FILE SPACE EXHAUSTED )                          00000390
```

```
02JI          PRINT/DISPLAY "ORDER FILE SPACE EXHAUSTED"                    00000400
10JI          CLEAR OK                                                      00000410
           CASE FCC ( INVALID CALL CODE )                                   00000420
54JI          PANIC                                                         00000430
           CASE  ?  ( INVALID COMPLETION CODE )                             00000440
54JI          PANIC                                                         00000450
        ENDCASE                                                             00000460
        RETURN                                                              00000470
                                                                            00000010
                                                                            00000020
                                                                            00000030
                                                                            00000040
                                                                            00000050
        DELETE DISK BATCH FILE RECORD ( DELETE.BATCH )                      00000060
                                                                            00000070
                                                                            00000080
                                                                            00000090
                        - MODULE HISTORY -                                  00000100
                                                                            00000110
                                                                            00000120
           PROJECT:     DAS                    76-01709                    00000130
                                                                            00000140
           SUB-SYSTEM:  OPERATOR INTERFACE                                  00000150
                                                                            00000160
           MODULE:      DICCDB                 85                          00000170
                                                                            00000180
                                                                            00000190
                        - MODULE ABSTRACT -                                 00000200
                                                                            00000210
                                                                            00000220
              CALL THE FILE MANAGER TO DELETE THE DISK BATCH                00000230
           FILE RECORD(S).                                                  00000240
              IF AN ERROR OCCURRED, TELL THE USER, CLEAR OK AND             00000250
           TAKE ANY FURTHER APPROPRIATE ACTION REQUIRED.                    00000260
              THIS SUBROUTINE WILL EXIT AFTER A SINGLE CALL TO THE          00000270
           FILE MANAGER WHETHER BATCH FILE ACCESS WAS SUCCESSFUL            00000280
           OR NOT.                                                          00000290
                                                                            00000300
                                                                            00000310
                                                                            00000320
14FM    UPDATE IPM FOR CALL                                                 00000330
08FM    CALL FILE MANAGER TO DELETE BATCH(ES)                               00000340
14FM    CASENTRY (FOK, FIR, FCC, ? )                                        00000350
           CASE FOK ( SUCCESSFUL )                                          00000360
           CASE FIR ( INVALID RELATIVE BATCH ID )                           00000370
54JI          PANIC                                                         00000380
           CASE FCC ( INVALID CALL CODE )                                   00000390
54JI          PANIC                                                         00000400
           CASE  ?  ( INVALID COMPLETION CODE )                             00000410
54JI          PANIC                                                         00000420
        ENDCASE                                                             00000430
        RETURN                                                              00000440
                                                                            00000010
                                                                            00000020
                                                                            00000030
                                                                            00000040
                                                                            00000050
        ORDER STATUS REPORT                                                 00000060
                                                                            00000070
                                                                            00000080
                                                                            00000090
                        - MODULE HISTORY -                                  00000100
                                                                            00000110
                                                                            00000120
           PROJECT:     DAS                    76-01709                    00000130
                                                                            00000140
           SUB-SYSTEM:  OPERATOR INTERFACE                                  00000150
                                                                            00000160
           MODULE:      DIOSR                  17                          00000170
                                                                            00000180
                                                                            00000190
                        - MODULE ABSTRACT -                                 00000200
```

```
                                                                        00000210
                                                                        00000220
      THIS MODULE REQUESTS A BATCH ID. THE FILE MANAGER IS              00000230
    CALLED TO READ THE BATCH FILE INFORMATION.                          00000240
      IF THE BATCH INFORMATION IS FOUND, THE USER IS ASKED IF           00000250
    ALL ORDERS ARE TO BE LISTED. IF ALL ORDERS ARE NOT TO BE            00000260
    LISTED, A SPECIFIC ORDER ID IS REQUESTED, AND THE FILE              00000270
    MANAGER IS CALLED TO READ THE ORDER FILE INFORMATION. IF THE        00000280
    ORDER INFORMATION IS FOUND, OR ALL ORDERS TO BE LISTED,             00000290
    REQUEST IF EXCEPTIONS ARE TO BE LISTED AND WHETHER OR NOT           00000300
    HARDCOPY OUTPUT IS REQUIRED. IF HARDCOPY IS REQUIRED, THEN          00000310
    THE TERMINAL IS DEQUEUED AND THE SYSTEM PRINTER IS ENQUEUED.        00000320
      THE BATCH ID, HEADER AND STATUS ARE THEN OUTPUT TO THE            00000330
    ENQUEUED DEVICE.                                                    00000340
      IF ALL ORDERS TO BE LISTED, THE FILE MANAGER IS CALLED TO         00000350
    SEQUENTIALLY READ EACH ORDER RECORD. FOR EACH ORDER RECORD,         00000360
    OR THE SPECIFICLY REQUESTED ORDER RECORD, THE ORDER ID, ALL         00000370
    QUANTITIES AND ORDER PERCENT COMPLETE ARE OUTPUT.                   00000380
      IF AN EXCEPTION LIST WAS REQUIRED, THE FILE MANAGER IS            00000390
    CALLED TO SEQUENTIALLY READ EACH TRANSACTION BLOCK. EACH            00000400
    TRANSACTION RECORD ASSOCIATED WITH THE ORDER IS EXAMINED. IF        00000410
    THE TRANSACTION STATUS IS AN EXCEPTION, THE TRANSACTION IN-         00000420
    FORMATION IS OUTPUT TO THE ENQUEUED DEVICE.                         00000430
      THE TERMINAL/PRINTER IS DEQUEUED BEFORE THE PROGRAM ENDS.         00000440

      ENQUEUE USER TERMINAL                                             00000450
040I  DISPLAY ON USER TERMINAL "ORDER STATUS REPORT"                    00000460
030I  ASK USER FOR BATCH ID                                             00000470
120I                                                                    00000480
480I  DEQUEUE AN IPM BUFFER (GET.IPM)                                   00000490
490I  DEQUEUE RECORD BUFFER (GET.SRB)                                   00000500
      INITIALIZE PANIC CODE                                             00000510
290I                                                                    00000520
160I  ACCESS BATCH FILE (GET.BATCH)                                     00000530
100I  IF BATCH RECORD FOUND (OK SET), THEN                              00000540
030I  !  ASK USER IF ALL ORDERS TO BE LISTED                           00000550
290I  !  IF NOT ALL ORDERS, THEN                                       00000560
030I  !     ASK USER FOR ORDER ID                                      00000570
130I  !                                                                00000580
      !     INITIALIZE PANIC CODE                                      00000590
290I  !                                                                00000600
180I  !     ACCESS ORDER FILE (GET.ORDER)                              00000610
100I  !     IF ORDER RECORD FOUND (OK SET), THEN                       00000620
100I  !        SET SPECIFIC ORDER                                      00000630
      !     (ELSE)                                                     00000640
      !     ENDIF                                                      00000650
      !  ELSE                                                          00000660
100I  !     SET OK                                                     00000670
100I  !     CLEAR SPECIFIC ORDER                                       00000680
      !  ENDIF                                                         00000690
100I  !  IF ORDER FOUND (OK SET), THEN                                 00000700
100I  !  !  SET ORDER SEQUENCE                                         00000710
030I  !  !  ASK USER IF EXCEPTIONS TO BE LISTED                        00000720
290I  !  !  IF EXCEPTIONS TO BE LISTED, THEN                           00000730
100I  !  !     SET EXCEPTIONS                                         00000740
      !  !  ELSE                                                       00000750
100I  !  !     CLEAR EXCEPTIONS                                       00000760
      !  !  ENDIF                                                      00000770
030I  !  !  ASK USER IF HARDCOPY OUTPUT IS REQUIRED                    00000780
      !  !  IF HARDCOPY REQUIRED, THEN                                 00000790
100I  !  !     SET HARDCOPY                                           00000800
      !  !     DEQUEUE USER TERMINAL                                  00000810
      !  !     ENQUEUE PRINTER                                        00000820
      !  !  ELSE                                                       00000830
100I  !  !     CLEAR HARDCOPY                                         00000840
      !  !  ENDIF                                                      00000850
290I  !  !  INITIALIZE DATE, TIME, PAGE COUNT AND LINE COUNT           00000860
070I  !  !  PRINT/DISPLAY BATCH HEADINGS                               00000870
070I  !  !  PRINT/DISPLAY BATCH INFORMATION                            00000880
070I  !  !  PRINT/DISPLAY ORDER HEADINGS                               00000890
      !  !  INITIALIZE PANIC CODE                                      00000900
290I  !  !                                                             00000910
130I  !  !  INITIALIZE ORDER COUNT                                     00000920
```

```
13DI    !  !  DOUNTIL ORDER COUNT GREATER THAN MAXIMUM                          00000930
10DI    !  !  !  IF SPECIFIC ORDER SET, THEN                                    00000940
13DI    !  !  !     UPDATE ORDER COUNT BEYOND MAXIMUM                           00000950
12DI    !  !  !                                                                 00000960
        !  !  !  ELSE                                                           00000970
17DI    !  !  !     FIND THE FIRST/NEXT ORDER IN THIS BATCH (FIND.ORDER)        00000980
        !  !  !  ENDIF                                                          00000990
10DI    !  !  !  IF ORDER RECORD FOUND (OK SET), THEN                           00001000
13DI    !  !  !  !  COMPUTE ORDER PERCENT COMPLETE                             00001010
        !  !  !  !  SET LIST ORDER INFORMATION                                 00001020
07DI    !  !  !  !  PRINT/DISPLAY ORDER INFORMATION AND PERCENT COMPLETE       00001030
        !  !  !  !  --- (FORMS.CTL) ---                                         00001040
10DI    !  !  !  !  IF ABORT SET THEN                                          00001050
13DI    !  !  !  !     SET ORDER COUNT GREATER THAN MAX.                       00001060
        !  !  !  !  ELSE                                                       00001070
10DI    !  !  !  !  !  IF EXCEPTIONS SET, THEN                                00001080
        !  !  !  !  !  !  IF EXCEPTIONS IN THIS ORDER THEN                    00001090
        !  !  !  !  !  !  !  CLEAR LIST ORDER INFORMATION                      00001100
07DI    !  !  !  !  !  !  !  PRINT/DISPLAY TRANSACTION HEADINGS               00001110
29DI    !  !  !  !  !  !  !                                                    00001120
        !  !  !  !  !  !  !  INITIALIZE PANIC CODE                            00001130
29DI    !  !  !  !  !  !  !                                                    00001140
29DI    !  !  !  !  !  !  !  INITIALIZE REPORT TRANSACTION COUNT              00001150
14DI    !  !  !  !  !  !  !  INITIALIZE TRANS. CURRENT BLOCK NUMBER           00001160
14DI    !  !  !  !  !  !  !  INITIALIZE TRANSACTION COUNT                     00001170
14DI    !  !  !  !  !  !  !  DOUNTIL TRANSACTION COUNT > MAXIMUM --           00001180
        !  !  !  !  !  !  !  -- OR BATCH NO LONGER ON FILE                    00001190
19DI    !  !  !  !  !  !  !     FIND THE FIRST/NEXT EXCEPTION RECORD          00001200
        !  !  !  !  !  !  !     --- (FIND.TRANS) ---                          00001210
        !  !  !  !  !  !  !     IF BATCH STILL ON FILE THEN                   00001220
10DI    !  !  !  !  !  !  !        IF EXCEPTION RECORD FOUND THEN             00001230
        !  !  !  !  !  !  !           IF TRANS. STATUS = MIS-SORT THEN        00001240
20FM    !  !  !  !  !  !  !              SET UP TO READ THE TRANS. DIR.       00001250
08FM    !  !  !  !  !  !  !              CALL FILE MANAGER FOR DIR. REC.      00001260
20FM    !  !  !  !  !  !  !              IF RETURN CODE = ERROR THEN          00001270
20FM    !  !  !  !  !  !  !                 IF RETURN CODE = NOT ON FILE      00001280
        !  !  !  !  !  !  !                    PRINT ABORT MSG.               00001290
        !  !  !  !  !  !  !                 ELSE                              00001300
54DI    !  !  !  !  !  !  !                    PANIC                          00001310
        !  !  !  !  !  !  !                 ENDIF                             00001320
        !  !  !  !  !  !  !              ELSE                                 00001330
03FM    !  !  !  !  !  !  !                 GET MIS-SORT DEST. FROM REC.      00001340
        !  !  !  !  !  !  !                 IF MIS-SORT DEST. = -1 THEN       00001350
        !  !  !  !  !  !  !                    SET OUTPUT = '?'               00001360
        !  !  !  !  !  !  !                 ELSE                              00001370
        !  !  !  !  !  !  !                    PLACE DEST. IN OUTPUT          00001380
        !  !  !  !  !  !  !                 ENDIF                             00001390
        !  !  !  !  !  !  !              ENDIF                                00001400
        !  !  !  !  !  !  !           ELSE                                    00001410
        !  !  !  !  !  !  !              SET OUTPUT TO BLANKS                 00001420
        !  !  !  !  !  !  !           ENDIF                                   00001430
        !  !  !  !  !  !  !           IF BATCH STILL ON FILE THEN             00001440
        !  !  !  !  !  !  !              SET LIST TRANSACTION INFO.           00001450
07DI    !  !  !  !  !  !  !              PRINT/DISPLAY TRANS INFORMATION      00001460
        !  !  !  !  !  !  !              --- (FORMS.CTL) ---                  00001470
        !  !  !  !  !  !  !           (ELSE)                                  00001480
        !  !  !  !  !  !  !           ENDIF                                   00001490
10DI    !  !  !  !  !  !  !           IF ABORT SET THEN                       00001500
14DI    !  !  !  !  !  !  !              SET TRANS. COUNT > MAX               00001510
13DI    !  !  !  !  !  !  !              SET ORDER COUNT > MAX.               00001520
        !  !  !  !  !  !  !           (ELSE)                                  00001530
        !  !  !  !  !  !  !           ENDIF                                   00001540
        !  !  !  !  !  !  !        (ELSE)                                     00001550
        !  !  !  !  !  !  !        ENDIF                                      00001560
        !  !  !  !  !  !  !     ELSE                                          00001570
04DI    !  !  !  !  !  !  !        OUTPUT "BATCH NO LONGER ON FILE            00001580
        !  !  !  !  !  !  !        --- REPORT TERMINATED" ---                 00001590
10DI    !  !  !  !  !  !  !        SET ABORT                                  00001600
        !  !  !  !  !  !  !     ENDIF                                         00001610
        !  !  !  !  !  !  !  ENDDO                                           00001620
        !  !  !  !  !  !  !  CLEAR LIST TRANSACTION INFORMATION              00001630
        !  !  !  !  !  !  ELSE                                               00001640
04DI    !  !  !  !  !  !     PRINT/DISPLAY "NO EXCEPTIONS IN THIS ORDER"     00001650
```

```
        !  !  !  !  !  !  ENDIF                                                  00001660
        !  !  !  !  !  (ELSE)                                                   00001670
        !  !  !  !  !  ENDIF                                                   00001680
        !  !  !  !  !  IF HARD COPY SET THEN                                   00001690
        !  !  !  !  !     UPDATE LINE COUNT > MAX.                             00001700
        !  !  !  !  !  ELSE                                                    00001710
        !  !  !  !  !     MOVE REPORT TO TOP OF SCREEN                         00001720
        !  !  !  !  !  ENDIF                                                   00001730
        !  !  !  !  ENDIF                                                     00001740
        !  !  !  ELSE                                                        00001750
02OI    !  !  !  !  PRINT/DISPLAY "BATCH NO LONGER ON FILE - RPT TERM"       00001760
10OI    !  !  !  !  SET ABORT                                                00001770
        !  !  !  !  IF HARD COPY NOT SET THEN                                00001780
        !  !  !  !     MOVE REPORT TO TOP OF SCREEN                          00001790
        !  !  !  !  (ELSE)                                                   00001800
        !  !  !  !  ENDIF                                                    00001810
        !  !  !  ENDIF                                                       00001820
        !  !  ENDDO                                                         00001830
        !  !  IF HARD COPY AND EXCEPTIONS NOT SET THEN                      00001840
        !  !     MOVE REPORT TO TOP OF SCREEN                               00001850
        !  !  ELSE                                                          00001860
        !  !  ENDIF                                                         00001870
        !  (ELSE)                                                           00001880
        !  ENDIF                                                            00001890
        ELSE                                                               00001900
02OI       PRINT/DISPLAY BATCH ID, "BATCH NOT ON FILE"                     00001910
        ENDIF                                                              00001920
51OI    RELEASE RECORD BUFFER (FREE.SRB)                                   00001930
50OI    RELEASE THE IPM BUFFER (FREE.IPM)                                  00001940
        DEQUEUE TERMINAL/PRINTER                                           00001950
        ENDPROG                                                            00001960
              FORMS.CTL SUBROUTINE                                         00001970
                                                                           00001980
10OI    IF HARDCOPY SET, THEN                                              00001990
29OI       IF LINE COUNT GREATER THAN PRINTER PAGE, THEN                   00002000
10OI          SET OK                                                       00002010
           ELSE                                                            00002020
10OI          CLEAR OK                                                     00002030
           ENDIF                                                           00002040
        ELSE                                                               00002050
29OI       IF LINE COUNT GREATER THAN TERMINAL PAGE, THEN                  00002060
10OI          SET OK                                                       00002070
           ELSE                                                            00002080
10OI          CLEAR OK                                                     00002090
           ENDIF                                                           00002100
        ENDIF                                                              00002110
10OI    IF PAGE OVERFLOW (OK SET), THEN                                    00002120
29OI       INCREMENT PAGE COUNT                                            00002130
07OI       PRINT/DISPLAY BATCH HEADINGS                                    00002140
07OI       PRINT/DISPLAY BATCH INFORMATION                                 00002150
07OI       PRINT/DISPLAY ORDER HEADINGS                                    00002160
29OI       INITIALIZE LINE COUNT                                           00002170
10OI       IF LIST TRANSACTION INFORMATION SET, THEN                       00002180
07OI          PRINT/DISPLAY ORDER INFORMATION                              00002190
07OI          PRINT/DISPLAY TRANSACTION HEADINGS                           00002200
29OI          INCREMENT LINE COUNT                                         00002210
           (ELSE)                                                          00002220
           ENDIF                                                           00002230
        (ELSE)                                                             00002240
        ENDIF                                                              00002250
10OI    IF LIST ORDER INFORMATION SET, THEN                                00002260
07OI       PRINT/DISPLAY ORDER INFORMATION                                 00002270
29OI       INCREMENT LINE COUNT                                            00002280
        (ELSE)                                                             00002290
        ENDIF                                                              00002300
10OI    IF LIST TRANSACTION INFORMATION SET, THEN                          00002310
07OI       PRINT/DISPLAY TRANSACTION INFORMATION                           00002320
29OI       INCREMENT REPORT TRANSACTION COUNT                              00002330
29OI       INCREMENT LINE COUNT                                            00002340
        (ELSE)                                                             00002350
        ENDIF                                                              00002360
        IF REPORT CANCELLED AND HARD COPY SET THEN                         00002370
           PRINT REPORT CANCELLED MESSAGE                                  00002380
```

```
      (ELSE)                                                                    00002390
      ENDIF                                                                     00002400
      RETURN                                                                    00002410
                                                                                00000010
                                                                                00000020
                                                                                00000030
                                                                                00000040
                                                                                00000050
      ALARM MESSAGE PROGRAM                                                     00000060
                                                                                00000070
                                                                                00000080
                                                                                00000090
                        - MODULE HISTORY -                                      00000100
                                                                                00000110
                                                                                00000120
         PROJECT:       DAS                     7G-01709                       00000130
                                                                                00000140
         SUB-SYSTEM:    ALARM MESSAGES                                         00000150
                                                                                00000160
         MODULE:        ALMPG                   1                              00000170
                                                                                00000180
                                                                                00000190
                        - MODULE ABSTRACT -                                     00000200
                                                                                00000210
                                                                                00000220
      THIS PROGRAM WILL DEQUEUE AN ALARM MESSAGE FROM THE ALARM                 00000230
   MESSAGE QUEUE. IF THE QUEUE IS EMPTY, IT WILL DELAY TWO                      00000240
   SECONDS AND START ALL OVER. IF THERE IS A MESSAGE, IT WILL GET               00000250
   THE MESSAGE DATA AND RETURN THE DATA BUFFER TO THE ALARM MES-                00000260
   SAGE BUFFER QUEUE. IF THE ADDRESS ON THE QUEUE WAS LESS THAN                 00000270
   THE ADDRESS OF $SYSCOM, THEREFORE NOT A VALID ADDRESS, THIS                  00000280
   PROGRAM WILL OUTPUT A FATAL ERROR MESSAGE WITH THE NUMBER AND                00000290
   DEQUEUE ANOTHER MESSAGE. IF THE ADDRESS WAS VALID, THE MESSAGE               00000300
   NUMBER IS USED TO FORMAT THE MESSAGE. THE MESSAGE IS THEN OUT-               00000310
   PUT TO THE APPROPRIATE LOG DEVICE, AND THE SEQUENCE STARTS OVER.             00000320
      INITIALIZATION                                                            00000330
39FM  GET NUMBER OF LOG DEVICES FROM SYSCOM                                     00000340
      ATTACH THAT MANY OUTPUT TASKS                                             00000350
      ATTACH TIME OUT TASK                                                      00000360
      DOUNTIL 1=0                                                               00000370
01AL     GET AN ELEMENT FROM THE ALARM MESSAGE QUEUE (ALMQUE)                   00000380
01AL     IF NO ELEMENT RECEIVED,  THEN                                          00000390
            DELAY AWHILE                                                        00000400
         ELSE                                                                   00000410
02AL        IF THE BUFFER ADDRESS > $SYSCOM,  THEN                              00000420
02AL           MOVE DATA FROM THE BUFFER TO AN INTERNAL BUFFER                  00000430
02AL           RETURN BUFFER TO THE BUFFER POOL (BUFFQUE)                       00000440
02AL           IF THE BUFFER POOL IS NOT FULL,  THEN                            00000450
11AL              SET COMPRESS FLAG                                             00000460
03AL              IF THE MESSAGE NUMBER IS WITHIN RANGE (1-N),  THEN            00000470
                     CASENTRY (MSG1,MSG2,....,MSGN)                             00000480
03AL                    CASE MSG1                                               00000490
                           FORMAT MSG1                                          00000500
03AL                    CASE MSG2                                               00000510
                           FORMAT MSG2                                          00000520
                        :                                                       00000530
                        :                                                       00000540
03AL                    CASE MSGN                                               00000550
                           FORMAT MSGN                                          00000560
                     ENDCASE                                                    00000570
                     PUT MESSAGE IN THE OUTPUT BUFFER                           00000580
10AL                 CALL THE DATE AND TIME SUBROUTINE                          00000590
12AL                 IF NOT A COMPRESSABLE MESSAGE, THEN                        00000600
11AL                    RESET COMPRESS FLAG                                     00000610
                     (ELSE)                                                     00000620
                     ENDIF                                                      00000630
11AL                 IF COMPRESS FLAG SET, THEN                                 00000640
13AL                    CALL COMPRESS SUBROUTINE                                00000650
                     (ELSE)                                                     00000660
                     ENDIF                                                      00000670
04AL                 GET ADDRESS OF TCW TABLE IN SYSCOM                         00000680
04AL                 ADD 2*(MSG#-1)                                             00000690
04AL                 GET TCW AT THIS ADDRESS                                    00000700
```

```
                    SAVE TCW POINTER                                        00000710
05AL                GET POINTER TO OUTPUT TASK EVENT TABLE                  00000720
04AL                DOUNTIL TCW=0                                           00000730
04AL                   IF TCW<0, THEN                                       00000740
05AL                      MOVE EVENT ADDR TO POST INSTRUCTION               00000750
05AL                      POST THE TASK START EVENT                         00000760
                       (ELSE)                                               00000770
                       ENDIF                                                00000780
05AL                   INCREMENT PTR TO NEXT TASK START EVENT               00000790
04AL                   SHIFT TCW TO THE LEFT, 1 BIT                         00000800
                    ENDDO                                                   00000810
09AL                POST TIMER START EVENT                                  00000820
                    GET TCW                                                 00000830
06AL                GET POINTER TO END EVENT TABLE                          00000840
04AL                DOUNTIL TCW = 0                                         00000850
04AL                   IF TCW<0, THEN                                       00000860
06AL                      MOVE EVENT ADDR TO WAIT AND RESET INSTRCTN        00000870
06AL                      WAIT FOR EVENT TO OCCUR                           00000880
06AL                      RESET THE EVENT                                   00000890
                       (ELSE)                                               00000900
                       ENDIF                                                00000910
05AL                   INCREMENT POINTER TO NEXT EVENT                      00000920
04AL                   SHIFT TCW TO THE LEFT, 1 BIT                         00000930
                    ENDDO                                                   00000940
09AL                SET TIMER COUNT TO ZERO                                 00000950
09AL                WAIT FOR TIMER END EVENT                                00000960
09AL                RESET TIMER END EVENT                                   00000970
                 ELSE                                                       00000980
08AL                SET PANIC CODE = BAD MSG NUMBER                         00000990
                    SET FLAG = FATAL ERROR                                  00001000
                 ENDIF                                                      00001010
              ELSE                                                          00001020
08AL             SET PANIC CODE = ALARM BUFFER QUEUE FULL                   00001030
                 SET FLAG = FATAL ERROR                                     00001040
              ENDIF                                                         00001050
           ELSE                                                             00001060
08AL          MOVE ADDRESS TO PANIC CODE                                    00001070
              SET FLAG = FATAL ERROR                                        00001080
           ENDIF                                                            00001090
        ENDIF                                                               00001100
08AL    IF A FATAL ERROR WAS ENCOUNTERED, THEN                              00001110
08AL       FORMAT "FATAL ERROR" AND PANIC # TO OUTPUT BUFFER                00001120
10AL       CALL DATE AND TIME SUBROUTINE                                    00001130
13AL       CALL MESSAGE COMPRESSION SUBROUTINE                              00001140
05AL       POST ALL OUTPUT TASKS                                            00001150
06AL       WAIT FOR ALL END EVENTS                                          00001160
        (ELSE)                                                              00001170
        ENDIF                                                               00001180
     ENDDO                                                                  00001190
     EXIT                                                                   00001200
                                                                            00001210
                                                                            00001220
                                                                            00001230
     TIME OUT TASK (TIMRTSK)                                                00001240
                                                                            00001250
     GET ADDRESSES OF ALL LOG DEVICE PRINT ECB'S                            00001260
     DOUNTIL 1=0                                                            00001270
        WAIT FOR TIMER START EVENT TO BE POSTED                             00001280
        RESET THE TIMER START EVENT                                         00001290
        SET UP TIME OUT COUNT                                               00001300
        DOWHILE TIME OUT COUNT > 0                                          00001310
           DELAY A LITTLE                                                   00001320
           DECREMENT THE TIME OUT COUNT                                     00001330
        ENDDO                                                               00001340
        POST ALL LOG DEVICE PRINT ECB'S                                     00001350
        POST TIMER END EVENT                                                00001360
     ENDDO                                                                  00001370
     EXIT                                                                   00001380
                                                                            00001390
                                                                            00001400
                                                                            00001410
     MESSAGE COMPRESSION SUBROUTINE   (MSGTXT)                              00001420
                                                                            00001430
```

```
SAVE REGISTERS                                                  00001440
GET TEXT ADDRESS                                                00001450
CALCULATE THE END ADDRESS FROM THE INDEX WORD                   00001460
DO WHILE POINTER < END ADDRESS                                  00001470
   GET CURRENT BYTE                                             00001480
   IF CURRENT BYTE IS A BLANK AND NEXT BYTE IS A BLANK, THEN    00001490
      GET NUMBER OF BYTES TO END OF MESSAGE                     00001500
      DO WHILE NOT AT END OF MESSAGE                            00001510
         MOVE A BYTE TO THE LEFT                                00001520
         INCREMENT POINTERS                                     00001530
      ENDDO                                                     00001540
      ADJUST INDEX WORD                                         00001550
      ADJUST END ADDRESS                                        00001560
   ELSE                                                         00001570
      INCREMENT POINTER                                         00001580
   ENDIF                                                        00001590
ENDDO                                                           00001600
RESTORE REGISTERS                                               00001610
RETURN                                                          00001620
                                                                00001630
                                                                00001640
                                                                00001650
GET DATE AND TIME SUBROUTINE (DATES)                            00001660
                                                                00001670
MOVE PUNCTUATION INTO THE OUTPUT BUFFER                         00001680
ADJUST BUFFER INDEX WORD                                        00001690
GET TIME AND DATE                                               00001700
GET A POINTER TO THE TEXT BUFFER                                00001710
GET A POINTER TO THE TIME AND DATE BUFFER                       00001720
SET A COUNTER = 6                                               00001730
DOUNTIL COUNTER = 0                                             00001740
   CONVERT TIME AND DATE ENTRY TO EBCDIC                        00001750
   PUT CONVERTED ENTRY IN THE BUFFER                            00001760
   CHANGE LEADING SPACES TO LEADING ZEROES                      00001770
   INCREMENT POINTERS                                           00001780
ENDDO                                                           00001790
RETURN                                                          00001800
                                                                00001810
                                                                00000010
                                                                00000020
                                                                00000030
                                                                00000040
                                                                00000050
ALARM MESSAGE TERMINAL 3 OUTPUT TASK                            00000060
                                                                00000070
                                                                00000080
                                                                00000090
                  - MODULE HISTORY -                            00000100
                                                                00000110
                                                                00000120
   PROJECT:       DAS                 7G-01709                 00000130
                                                                00000140
   SUB-SYSTEM:    ALARM MESSAGES                               00000150
                                                                00000160
   MODULE:        ALTRM3              4                        00000170
                                                                00000180
                                                                00000190
                  - MODULE ABSTRACT -                           00000200
                                                                00000210
                                                                00000220
   THIS TASK ENQUEUES LOG DEVICE 3. IT THEN WAITS FOR A         00000230
START EVENT TO BE POSTED. WHEN POSTED, IT WILL PRINT THE        00000240
CONTENTS OF THE OUTPUT BUFFER ON THE LOG DEVICE. THEN IT        00000250
RESETS ITS START EVENT AND POSTS ITS END EVENT. THEN RE-        00000260
TURNS TO WAIT FOR ITS START EVENT AGAIN.                        00000270
4AL  ENQUEUE LOG DEVICE 003                                     00000280
     DOUNTIL 1=0                                                00000290
5AL     WAIT FOR START EVENT TO BE POSTED                       00000300
        PRINT ALARM MESSAGE ON THE LOG DEVICE                   00000310
5AL     RESET THE START EVENT                                   00000320
6AL     POST THE END EVENT                                      00000330
     ENDDO                                                      00000340
     DEQUEUE THE LOG DEVICE                                     00000350
     EXIT                                                       00000360
```

```
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
ALARM MESSAGE TERMINAL 2 OUTPUT TASK                                00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                    78-01709                00000130
                                                                    00000140
   SUB-SYSTEM:      ALARM MESSAGES                                  00000150
                                                                    00000160
   MODULE:          ALTRM2                 3                        00000170
                                                                    00000180
                                                                    00000190
                    - MODULE ABSTRACT -                             00000200
                                                                    00000210
                                                                    00000220
   THIS TASK ENQUEUES LOG DEVICE 2. IT THEN WAITS FOR A             00000230
START EVENT TO BE POSTED. WHEN POSTED, IT WILL PRINT THE            00000240
CONTENTS OF THE OUTPUT BUFFER ON THE LOG DEVICE. THEN IT            00000250
RESETS ITS START EVENT AND POSTS ITS END EVENT. THEN RE-            00000260
TURNS TO WAIT FOR ITS START EVENT AGAIN.                            00000270
4AL ENQUEUE LOG DEVICE 002                                          00000280
    DOUNTIL 1=0                                                     00000290
5AL    WAIT FOR START EVENT TO BE POSTED                            00000300
       PRINT ALARM MESSAGE ON THE LOG DEVICE                        00000310
5AL    RESET THE START EVENT                                        00000320
6AL    POST THE END EVENT                                           00000330
    ENDDO                                                           00000340
    DEQUEUE THE LOG DEVICE                                          00000350
    EXIT                                                            00000360
                                                                    00000010
                                                                    00000020
                                                                    00000030
                                                                    00000040
                                                                    00000050
ALARM MESSAGE TERMINAL 1 OUTPUT TASK                                00000060
                                                                    00000070
                                                                    00000080
                                                                    00000090
                    - MODULE HISTORY -                              00000100
                                                                    00000110
                                                                    00000120
   PROJECT:         DAS                    78-01709                00000130
                                                                    00000140
   SUB-SYSTEM:      ALARM MESSAGES                                  00000150
                                                                    00000160
   MODULE:          ALTRM1                 2                        00000170
                                                                    00000180
                                                                    00000190
                    - MODULE ABSTRACT -                             00000200
                                                                    00000210
                                                                    00000220
   THIS TASK ENQUEUES LOG DEVICE 1. IT THEN WAITS FOR A             00000230
START EVENT TO BE POSTED. WHEN POSTED, IT WILL PRINT THE            00000240
CONTENTS OF THE OUTPUT BUFFER ON THE LOG DEVICE. THEN IT            00000250
RESETS ITS START EVENT AND POSTS ITS END EVENT. THEN RE-            00000260
TURNS TO WAIT FOR ITS START EVENT AGAIN.                            00000270
4AL ENQUEUE LOG DEVICE 001                                          00000280
    DOUNTIL 1=0                                                     00000290
5AL    WAIT FOR START EVENT TO BE POSTED                            00000300
       PRINT ALARM MESSAGE ON THE LOG DEVICE                        00000310
5AL    RESET THE START EVENT                                        00000320
6AL    POST THE END EVENT                                           00000330
    ENDDO                                                           00000340
    DEQUEUE THE LOG DEVICE                                          00000350
    EXIT                                                            00000360
```

```
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
CHANGE TRANSACTION STATUS                                               00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                    78-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      FILE MANAGER                                        00000150
                                                                        00000160
   MODULE:          FMCTS                  36                          00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
                                                                        00000220
                                                                        00000230
      THIS SUBROUTINE PROCESSES THE CHANGE TRANSACTION STATUS           00000240
   FILE MANAGER CALL CODES. IT CHECKS THE VALIDITY OF THE INPUT         00000250
   STATUS, SEARCHES THE TRANSACTION DIRECTORY FOR THE                   00000260
   INPUT TRANSACTION NUMBER, THEN CHANGES THE TRANSACTION               00000270
   STATUS ON FILE TO THE INPUT STATUS. IT ALSO UPDATES THE              00000280
   ORDER STATUS COUNTS ASSOCIATED WITH THIS TRANSACTION.                00000290
16FM  GET INPUTS                                                        00000300
17FM                                                                    00000310
16FM  GET INPUT TRANSACTION NUMBER                                      00000320
17FM                                                                    00000330
16FM  IF CALL CODE = CHANGE STATUS FROM SCANNERS AND NEW STATUS --      00000340
      -- EQUAL IN SORTATION THEN                                       00000350
16FM     GET DIRECTORY RECORD FROM INPUT BUFFER                         00000360
      ELSE                                                              00000370
03FM     COMPUTE TRANSACTION DIRECTORY SECTOR FOR THIS TRANS. NUMBER    00000380
28FM     READ THE TRANSACTION DIRECTORY SECTOR                          00000390
03FM     INDEX INTO SECTOR FOR TRANSACTION DIRECTORY RECORD             00000400
      ENDIF                                                             00000410
03FM  GET RELATIVE BATCH NUMBER FROM DIRECTORY RECORD                   00000420
02FM  INDEX INTO THE RESIDENT BATCH FILE BY RELATIVE BATCH NUMBER       00000430
03FM  IF THE TRANSACTION DIRECTORY RECORD UNUSED OR THE BATCH -         00000440
02FM  -STATUS = ADDING, DELETING OR FREE THEN                           00000450
16FM     SET COMPLETION CODE = TRANSACTION NOT ON FILE                  00000460
17FM                                                                    00000470
      ELSE                                                              00000480
16FM     SAVE INPUT STATUS AS NEW STATUS                                00000490
17FM                                                                    00000500
17FM     IF CALL CODE = CHANGE STATUS FROM STATUS SCANNER THEN          00000510
17FM        IF NEW STATUS NOT 'EXCESS REPACK, STOCK-OUT, -              00000520
         - STAGED FOR SHIPMENT, IN SORTATION, OR MIS-SORT' THEN         00000530
17FM           SET COMPLETION CODE = INVALID STATUS                     00000540
           (ELSE)                                                       00000550
           ENDIF                                                        00000560
        ELSE                                                            00000570
16FM        IF NEW STATUS NOT 'EXCESS REPACK, STOCK-OUT, -              00000580
            - STAGED FOR SHIPMENT, IN SORTATION, -                      00000590
            - OR NOT PICKED THEN                                        00000600
16FM           SET COMPLETION CODE = INVALID STATUS                     00000610
            ELSE                                                        00000620
16FM           IF REC. BATCH ID NOT EQUAL TO INPUT BATCH ID THEN        00000630
02FM                                                                    00000640
16FM              SET COMPLETION CODE = TRANSACTION NOT IN BATCH        00000650
              (ELSE)                                                    00000660
            ENDIF                                                       00000670
        ENDIF                                                           00000680
      ENDIF                                                             00000690
16FM  IF COMPLETION CODE = SUCCESSFUL THEN                              00000700
17FM                                                                    00000710
03FM     GET RELATIVE TRANS. NO. FROM TRANS. DIR. RECORD                00000720
04FM     COMPUTE TRANSACTION FILE SECTOR NUMBER                         00000730
28FM     READ THE TRANSACTION FILE SECTOR                               00000740
```

```
16FM       IF TRANS. NO. IN RECORD = INPUT TRANSACTION NUMBER THEN      00000750
04FM                                                                    00000760
04FM          INDEX TO TRANSACTION RECORD IN SECTOR                     00000770
              SAVE CURRENT TRANS. STATUS AS OLD STATUS                  00000780
04FM          SET THE TRANS. STATUS = NEW STATUS                        00000790
04FM          GET THE RELATIVE ORDER RECORD NUMBER FROM TRANS. RECORD   00000800
28FM          WRITE THE TRANSACTION FILE SECTOR                         00000810
05FM          IF NEW STATUS = 'MIS-SORT' THEN                           00000820
28FM             READ TRANS. DIR. SECTOR                                00000830
03FM             INDEX INTO SECTOR FOR TRANS. DIRECTORY RECORD          00000840
03FM             SET MIS-SORT DESTINATION IN TRANS. DIRECTORY RECORD    00000850
16FM                                                                    00000860
28FM             WRITE THE TRANSACTION DIRECTORY SECTOR                 00000870
              (ELSE)                                                    00000880
              ENDIF                                                     00000890
05FM          COMPUTE THE ORDER FILE SECTOR NUMBER                      00000900
28FM          READ THE ORDER FILE SECTOR                                00000910
05FM          INDEX TO ORDER RECORD IN SECTOR                           00000920
05FM          INDEX TO OLD STATUS COUNT IN ORDER RECORD                 00000930
05FM          SUBTRACT 1 FROM THE OLD STATUS COUNT                      00000940
05FM          INDEX TO NEW STATUS COUNT IN ORDER RECORD                 00000950
05FM          ADD 1 TO THE NEW STATUS COUNT                             00000960
28FM          WRITE THE ORDER FILE SECTOR                               00000970
           ELSE                                                         00000980
16FM          SET RETURN COMPLETION CODE = TRANSACTION NOT ON FILE      00000990
           ENDIF                                                        00001000
        (ELSE)                                                          00001010
        ENDIF                                                           00001020
16FM    RETURN TO CALLER                                                00001030
17FM                                                                    00001040
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
        FILE MANAGER PANIC PROCESSOR                                    00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                          - MODULE HISTORY -                            00000100
                                                                        00000110
                                                                        00000120
           PROJECT:       DAS                  73-01709                00000130
                                                                        00000140
           SUB-SYSTEM:    FILE MANAGER                                 00000150
                                                                        00000160
           MODULE:        FMPAN                48                      00000170
                                                                        00000180
                                                                        00000190
                          - MODULE ABSTRACT -                           00000200
           THIS SUBROUTINE IS CALLED FROM SEVERAL FILE MANAGER          00000210
        MODULES TO SEND A PANIC CODE TO THE ALARM MESSAGE SUB-SYSTEM    00000220
        TO PRINT A FATAL ERROR MESSAGE AND TO SUSPEND EXECUTION OF      00000230
        THE FILE MANAGER SUB-SYSTEM.                                    00000240
35FM    GET INPUTS                                                      00000250
07AL                                                                    00000260
08AL    PUT PANIC CODE INPUT INTO ALARM MSG QUEUE                       00000270
        DOUNTIL NEW IPL                                                 00000280
           WAIT FOR IMPOSSIBLE EVENT                                    00000290
        ENDDO                                                           00000300
        RETURN (NOT REALLY)                                             00000310
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
        USER INTERFACE TO FILE MANAGER                                  00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                          - MODULE HISTORY -                            00000100
                                                                        00000110
                                                                        00000120
           PROJECT:       DAS                  73-01709                00000130
```

```
                                                                        00000140
      SUB-SYSTEM:     FILE MANAGER                                      00000150
                                                                        00000160
      MODULE:         FM                    47                          00000170
                                                                        00000180
                                                                        00000190
                      - MODULE ABSTRACT -                               00000200
                                                                        00000210
                                                                        00000220
      THIS SUBROUTINE PERFORMS ALL THE FUNCTIONS NEEDED TO              00000230
    CALL THE FILE MANAGER. A COPY OF THIS SUBROUTINE IS TO              00000240
    BE INCLUDED IN EACH DAS SUBSYSTEM THAT CALLS THE FILE               00000250
    MANAGER.                                                            00000260
08FM  GET INPUTS                                                        00000270
34FM  ENQUEUE THE FILE MANAGER RESOURCE                                 00000280
27FM  RESET FILE MANAGER COMPLETION EVENT                               00000290
25FM  POST FILE MANAGER ENTRY EVENT                                     00000300
27FM  WAIT ON FILE MANAGER COMPLETION EVENT                             00000310
34FM  DEQUEUE THE FILE MANAGER RESOURCE                                 00000320
08FM  RETURN TO CALLER                                                  00000330
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
    ADD BATCH COMPLETE                                                  00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                      - MODULE HISTORY -                                00000100
                                                                        00000110
                                                                        00000120
      PROJECT:        DAS                   73-01709                  00000130
                                                                        00000140
      SUB-SYSTEM:     FILE MANAGER                                      00000150
                                                                        00000160
      MODULE:         FMADC                 90                          00000170
                                                                        00000180
                                                                        00000190
                      - MODULE ABSTRACT -                               00000200
                                                                        00000210
                                                                        00000220
      THIS SUBROUTINE PROCESSES THE ADD BATCH COMPLETE FILE MANAGER     00000230
    CALL CODE. IT FINDS THE BATCH RECORD WITH THE STATUS OF             00000240
    ADDING AND CHANGES IT TO PENDING IN BOTH THE RESIDENT BATCH         00000250
    FILE AND THE DISK BATCH FILE.                                       00000260
38FM  GET INPUTS                                                        00000270
02FM  GET ADDRESS OF RESIDENT BATCH FILE                                00000280
      SET RELATIVE BATCH NUMBER = ZERO                                  00000290
02FM  DOUNTIL END OF RESIDENT BATCH FILE OR BATCH WITH STATUS --        00000300
      -- ADDING FOUND                                                  00000310
         GET FIRST/NEXT RECORD                                          00000320
02FM     IF RECORD STATUS NOT ADDING THEN                               00000330
            ADD 1 TO RELATIVE BATCH NUMBER                              00000340
         (ELSE)                                                         00000350
         ENDIF                                                          00000360
      ENDDO                                                             00000370
      IF BATCH FOUND THEN                                               00000380
05FM     COMPUTE FIRST ORDER FILE SECTOR ADDRESS FROM --                00000390
         -- RELATIVE BATCH NUMBER                                       00000400
28FM     UPDATE RESIDENT ORDER FILE                                     00000410
01FM     COMPUTE BATCH FILE SECTOR ADDRESS FROM RELATIVE BATCH NO.      00000420
28FM     READ THE BATCH FILE SECTOR                                     00000430
01FM     INDEX TO THE BATCH RECORD IN THE SECTOR                        00000440
01FM     SET BATCH STATUS TO PENDING                                    00000450
28FM     WRITE THE BATCH FILE SECTOR                                    00000460
02FM     SET RESIDENT BATCH RECORD STATUS TO PENDING                    00000470
      ELSE                                                              00000480
38FM     SET RETURN COMPLETION CODE = BATCH NOT ON FILE                 00000490
      ENDIF                                                             00000500
38FM  RETURN TO CALLER                                                  00000510
```

```
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
READ TRANSACTION DIRECTORY ENTRY VIA TRANSACTION NUMBER                 00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                 76-01709                        00000130
                                                                        00000140
   SUB-SYSTEM:      FILE MANAGER                                        00000150
                                                                        00000160
   MODULE:          FMRTD               41                              00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
                                                                        00000220
   THIS SUBROUTINE READS A TRANSACTION DIRECTORY ENTRY INTO THE         00000230
CALLER'S BUFFER WHEN GIVEN A TRANSACTION NUMBER AS INPUT.               00000240
20FM  GET INPUTS                                                        00000250
03FM  USING TRANSACTION NO. INPUT, COMPUTE TRANS. DIR. SECTOR NO.       00000260
28FM  READ TRANSACTION DIRECTORY SECTOR                                 00000270
03FM  INDEX INTO SECTOR TO TRANSACTION DIRECTORY ENTRY                  00000280
03FM  IF ENTRY IS UNUSED OR BATCH STATUS = ADDING/DELETING OR FREE THE  00000290
20FM     SET RETURN COMPLETION CODE = TRANSACTION NOT ON FILE           00000300
      ELSE                                                              00000310
03FM     COPY TRANS. DIR. ENTRY INTO USER BUFFER                        00000320
20FM                                                                    00000330
08FM     GET RELATIVE TRANSACTION NO. FROM DIRECTORY RECORD             00000340
04FM     COMPUTE TRANSACTION FILE SECTOR FROM REL. TRANS. NO.           00000350
28FM     READ THE TRANSACTION FILE SECTOR                               00000360
04FM     INDEX TO TRANSACTION RECORD                                    00000370
20FM     IF INPUT TRANSACTION NO. NOT EQUAL TO TRANSACTION NO. --       00000380
04FM     -- IN RECORD THEN                                              00000390
20FM        SET RETURN COMPLETION CODE = TRANSACTION NOT ON FILE        00000400
         (ELSE)                                                         00000410
         ENDIF                                                          00000420
      ENDIF                                                             00000430
20FM  RETURN TO CALLER                                                  00000440
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
READ ORDER RECORD VIA ORDER I.D.                                        00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                 76-01709                        00000130
                                                                        00000140
   SUB-SYSTEM:      FILE MANAGER                                        00000150
                                                                        00000160
   MODULE:          FMOID               33                              00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
                                                                        00000220
   THIS SUBROUTINE PROCESSES THE READ ORDER RECORD VIA                  00000230
ORDER I.D. FILE MANAGER CALL CODE. IT CHECKS FOR A VALID                00000240
BATCH I.D. THEN SEARCHES THE ORDER FILE                                 00000250
FOR THE INPUT ORDER I.D.. WHEN FOUND THIS ROUTINE PLACES                00000260
THE REQUESTED ORDER RECORD INTO THE CALLERS BUFFER.                     00000270
```

```
22FM    GET INPUTS                                                      00000280
33FM    CALL ROUTINE TO SEARCH RESIDENT BATCH FILE FOR INPUT BATCH I.D. 00000290
33FM    IF INPUT BATCH ID FOUND THEN                                    00000300
01FM       COMPUTE THE BATCH FILE SECTOR FOR THE INPUT BATCH            00000310
28FM       READ THE BATCH FILE SECTOR                                   00000320
01FM       INDEX TO BATCH RECORD IN BATCH FILE SECTOR                   00000330
01FM       GET THE NUMBER OF ORDERS FROM THE BATCH RECORD               00000340
05FM       COMPUTE STARTING SECTOR IN ORDER FILE FOR BATCH              00000350
           DOUNTIL ALL ORDER SECTORS SEARCHED OR RECORD FOUND           00000360
28FM          READ FIRST/NEXT ORDER FILE SECTOR                         00000370
05FM          DOUNTIL ALL VALID ORDER RECORDS IN SECTOR SEARCHED OR -    00000380
              - RECORD FOUND                                            00000390
05FM             GET FIRST/NEXT ORDER RECORD IN SECTOR                  00000400
05FM             IF RECORD ORDER I.D. = INPUT ORDER I.D. THEN            00000410
22FM                                                                    00000420
05FM                COPY ORDER RECORD INTO USER BUFFER                  00000430
22FM                                                                    00000440
                 (ELSE)                                                 00000450
                 ENDIF                                                  00000460
              ENDDO                                                     00000470
           ENDDO                                                        00000480
           IF ORDER RECORD NOT FOUND THEN                               00000490
22FM          SET COMPLETION CODE = ORDER NOT FOUND                     00000500
           (ELSE)                                                       00000510
           ENDIF                                                        00000520
        ELSE                                                            00000530
22FM       SET COMPLETION CODE = BATCH NOT FOUND                        00000540
        ENDIF                                                           00000550
22FM    RETURN TO CALLER                                                00000560
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
        READ TRANSACTION RECORD VIA TRANSACTION NUMBER                  00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                         - MODULE HISTORY -                             00000100
                                                                        00000110
                                                                        00000120
           PROJECT:      DAS                  79-01709                 00000130
                                                                        00000140
           SUB-SYSTEM:   FILE MANAGER                                   00000150
                                                                        00000160
           MODULE:       FMTID                39                       00000170
                                                                        00000180
                                                                        00000190
                         - MODULE ABSTRACT -                            00000200
                                                                        00000210
                                                                        00000220
           THIS SUBROUTINE COMPUTES THE TRANSACTION DIRECTORY ENTRY      00000230
        FROM THE INPUT TRANSACTION NUMBER.  THEN, UPON READING THE      00000240
        ASSOCIATED DIRECTORY ENTRY, THE TRANSACTION FILE RELATIVE       00000250
        RECORD ADDRESS CAN BE OBTAINED AND USED TO ACCESS THE DISK-     00000260
        RESIDENT TRANSACTION FILE TO OBTAIN THE RELEVANT TRANSACTION    00000270
        RECORD WHICH IS THEN DUTIFULLY COPIED INTO THE CALLER'S BUFFER. 00000280
19FM    GET INPUTS                                                      00000290
03FM    FROM TRANSACTION NUMBER, COMPUTE TRANSACTION DIRECTORY SECTOR NO00000300
28FM    READ TRANSACTION DIRECTORY SECTOR                               00000310
03FM    INDEX INTO SECTOR FOR TRANSACTION ENTRY                         00000320
03FM    IF ENTRY IS UNUSED OR BATCH STATUS = ADDING/DELETING OR FREE THE00000330
19FM       SET RETURN COMPLETION CODE = TRANSACTION NOT ON FILE         00000340
        ELSE                                                            00000350
03FM       GET RELATIVE TRANSACTION NO. FROM DIRECTORY                  00000360
04FM       COMPUTE TRANSACTION FILE SECTOR NUMBER                       00000370
28FM       READ TRANSACTION FILE SECTOR                                 00000380
04FM       INDEX INTO SECTOR FOR TRANSACTION RECORD                     00000390
19FM       IF INPUT TRANSACTION NO. IS EQUAL TO TRANSACTION NO. --      00000400
04FM       -- IN RECORD THEN                                            00000410
04FM          COPY RECORD INTO CALLER'S BUFFER                          00000420
           ELSE                                                         00000430
19FM          SET RETURN COMPLETION CODE = TRANSACTION NOT ON FILE      00000440
           ENDIF                                                        00000450
```

```
19FM                                                                    00000460
      ENDIF                                                             00000470
19FM  RETURN TO CALLER                                                  00000480
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
      READ ORDER RECORD VIA RELATIVE ORDER NUMBER                       00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
         PROJECT:       DAS                  76-01709                  00000130
                                                                        00000140
         SUB-SYSTEM:    FILE MANAGER                                   00000150
                                                                        00000160
         MODULE:        FMROL                40                        00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
         THIS SUBROUTINE FIRST CHECKS THE BATCH ID INPUT FOR VALIDITY   00000230
      AND THEN DETERMINES THE NO. OF ORDERS IN THE SPECIFIED BATCH.     00000240
      WITH THE NO. OF ORDERS IN THE BATCH KNOWN, THE RELATIVE ORDER     00000250
      NO. CAN BE VERIFIED.  HAVING A GOOD RELATIVE ORDER NO., IT'S      00000260
      A TRIVIAL EXERCISE TO ACCESS THE ORDER FILE ON DISK (IN THE       00000270
      PROPER BATCH SEGMENT, OF COURSE) TO GET THE ORDER RECORD          00000280
      AND COPY IT INTO THE CALLER'S BUFFER.  NOW LET'S GET TO THE       00000290
      MEAT...                                                           00000300
23FM  GET INPUTS                                                        00000310
33FM  CALL ROUTINE TO SEARCH RES. BATCH FILE FOR BATCH ID INPUT         00000320
33FM  IF THE BATCH WAS FOUND THEN                                       00000330
01FM     READ THE BATCH FILE SECTOR WHICH CONTAINS THE INDICATED BATCH  00000340
01FM     GET THE NO. OF ORDERS IN THE BATCH FROM THE RECORD             00000350
23FM     IF THE RELATIVE ORDER NO. INPUT > NO. OF ORDERS OR < 1 THEN    00000360
23FM        SET COMPLETION CODE = INVALID REL. RECORD NO.               00000370
         ELSE                                                           00000380
05FM        COMPUTE STARTING SECTOR NO. IN ORDER FILE FOR BATCH         00000390
05FM        COMPUTE SECTOR NO. FOR ORDER RECORD                         00000400
28FM        READ SECTOR FROM ORDER FILE                                 00000410
05FM        INDEX INTO SECTOR FOR ORDER RECORD                          00000420
05FM        COPY ORDER RECORD INTO USER BUFFER                          00000430
23FM                                                                    00000440
         ENDIF                                                          00000450
      ELSE                                                              00000460
23FM     SET COMPLETION CODE = BATCH NOT ON FILE                        00000470
      ENDIF                                                             00000480
23FM  RETURN TO CALLER                                                  00000490
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
      READ TRANSACTION FILE BLOCK                                       00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
         PROJECT:       DAS                  76-01709                  00000130
                                                                        00000140
         SUB-SYSTEM:    FILE MANAGER                                   00000150
                                                                        00000160
         MODULE:        FMRTB                37                        00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
```

```
      THIS SUBROUTINE PROCESSES THE READ TRANSACTION FILE BLOCK      00000230
    FILE MANAGER CALL CODE. IT SEARCHES THE RESIDENT BATCH FILE      00000240
    FOR THE INPUT BATCH I.D.. THEN INPUTS THE REQUESTED TRANS.       00000250
    FILE SECTOR AND PASSES THE SECTOR TO THE CALLER.                 00000260
                                                                     00000270
21FM  GET INPUTS                                                     00000280
33FM  CALL ROUTINE TO SEARCH RESIDENT BATCH FILE FOR INPUT BATCH I.D.00000290
33FM  IF THE BATCH RECORD FOUND THEN                                 00000300
01FM     COMPUTE THE BATCH FILE SECTOR FOR THIS BATCH                00000310
28FM     READ THE BATCH FILE SECTOR                                  00000320
01FM     INDEX TO THE BATCH RECORD IN THE SECTOR                     00000330
01FM     GET THE NUMBER OF TRANSACTIONS IN THIS BATCH                00000340
04FM     COMPUTE THE MAX. NO. OF TRANS. FILE SECTORS IN THIS BATCH   00000350
21FM     IF THE INPUT RELATIVE BLOCK NO. > MAX. TRANS. SECTORS THEN  00000360
21FM        SET COMPLETION CODE = INVALID RELATIVE REC./BLOCK NO.    00000370
         ELSE                                                        00000380
04FM        COMPUTE THE TRANSACTION FILE SECTOR NUMBER               00000390
28FM        READ THE TRANS. FILE SECTOR                              00000400
21FM        COPY DATA INTO USER'S BUFFER                             00000410
         ENDIF                                                       00000420
      ELSE                                                           00000430
21FM     SET COMPLETION CODE = BATCH NOT ON FILE                     00000440
      ENDIF                                                          00000450
21FM  RETURN TO CALLER                                               00000010
                                                                     00000020
                                                                     00000030
                                                                     00000040
                                                                     00000050
    READ/WRITE BLOCK (SECTOR) FROM/TO DISK FILE                      00000060
                                                                     00000070
                                                                     00000080
                                                                     00000090
                    - MODULE HISTORY -                               00000100
                                                                     00000110
                                                                     00000120
       PROJECT:        DAS                  78-01709                00000130
                                                                     00000140
       SUB-SYSTEM:     FILE MANAGER                                  00000150
                                                                     00000160
       MODULE:         FMRW3                42                       00000170
                                                                     00000180
                                                                     00000190
                    - MODULE ABSTRACT -                              00000200
                                                                     00000210
                                                                     00000220
       THIS MODULE READS/WRITES SECTORS FROM/TO DISK FILES           00000230
    INDICATED BY RELATIVE FILE NUMBER (FILE CODE).                   00000240
28FM  GET INPUTS                                                     00000250
    CHECK VALIDITY OF BLOCK NUMBER/FILE CODE                         00000260
    IF INPUTS ARE VALID THEN                                         00000270
       IF READ SECTOR WAS REQUESTED THEN                             00000280
          IF BLOCK ALREADY IN MEMORY THEN                            00000290
             COPY BLOCK INTO CALLERS BUFFER                          00000300
          ELSE                                                       00000310
             READ BLOCK(BLK INPUT) FROM FILE(FILE CODE INPUT)        00000320
          ENDIF                                                      00000330
       ELSE                                                          00000340
          COPY CALLERS BUFFER TO MEMORY BLOCK                        00000350
          WRITE BLOCK(BLK INPUT) TO FILE(FILE CODE INPUT)            00000360
       ENDIF                                                         00000370
       IF DISK ERROR ON READ/WRITE THEN                              00000380
35FM      PANIC! (DO IT WITH CALL CODE FROM USER)                    00000390
09FM                                                                 00000400
          WAIT FOR IPL RELIEF...                                     00000410
       (ELSE)                                                        00000420
       ENDIF                                                         00000430
    ELSE                                                             00000440
35FM   PANIC!                                                        00000450
    ENDIF                                                            00000460
28FM  RETURN TO CALLER                                               00000470
                                                                     00000010
                                                                     00000020
                                                                     00000030
                                                                     00000040
```

```
                                                                        00000050
ADD TRANSACTION RECORD TO FILES                                         00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                     78-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      FILE MANAGER                                        00000150
                                                                        00000160
   MODULE:          FMATR                   30                          00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
   THIS SUBROUTINE PROCESSES THE ADD TRANSACTION RECORD                 00000230
FILE MANAGER CALL CODE. IT CHECKS FOR ANY CONFLICTS OF THE              00000240
NEW TRANSACTION RECORD WITH THE TRANSACTION RECORDS ALREADY             00000250
ON FILE THEN ADDS THE NEW TRANSACTION RECORD TO THE FILE                00000260
IF NO CONFLICTS WERE FOUND. WHEN THE LAST TRANSACTION RECORD            00000270
IS ADDED TO THE FILE, THE ORDER RECORDS ARE ADDED AND THE               00000280
BATCH RECORD COUNTS ARE UPDATED TO SHOW THE ACTUAL NUMBER               00000290
OF ORDERS AND TRANSACTIONS IN THIS BATCH. THE STATUS OF THE             00000300
BATCH RECORD IS THEN CHANGED TO 'PENDING'.                              00000310
13FM    GET INPUTS                                                      00000320
02FM    DOUNTIL BATCH BEING ADDED FOUND OR END OF RESIDENT BATCH FILE   00000330
           GET FIRST/NEXT RESIDENT BATCH RECORD                         00000340
           IF RECORD STATUS = ADDING THEN                               00000350
              SAVE THE RELATIVE BATCH NUMBER                            00000360
           (ELSE)                                                       00000370
           ENDIF                                                        00000380
        ENDDO                                                           00000390
        IF END OF RESIDENT BATCH FILE FOUND THEN                        00000400
13FM       SET COMPLETION CODE = INVALID CALL CODE                      00000410
        ELSE                                                            00000420
03FM       COMPUTE TRANSACTION DIRECTORY SECTOR FOR INPUT TRANS. NUMBER 00000430
           IF CURRENT SECTOR NOT EQUAL TO COMPUTED SECTOR THEN          00000440
              WRITE CURRENT SECTOR                                      00000450
              READ TRANSACTION DIRECTORY SECTOR                         00000460
           (ELSE)                                                       00000470
           ENDIF                                                        00000480
03FM       INDEX TO TRANSACTION DIRECTORY RECORD IN SECTOR              00000490
03FM       IF TRANSACTION DIRECTORY RECORD IS USED THEN                 00000500
03FM          GET RELATIVE BATCH NUMBER FROM RECORD                     00000510
02FM          INDEX INTO RESIDENT BATCH FILE BY RELATIVE BATCH NUMBER   00000520
02FM          IF THE BATCH STATUS NOT DELETING OR NOT FREE THEN         00000530
03FM             GET RELATIVE TRANS. NO. FROM DIRECTORY RECORD          00000540
04FM             COMPUTE TRANS. FILE SECTOR NUMBER                      00000550
                 IF COMPUTED TRANS. FILE SECTOR NOT IN MEMORY THEN      00000560
                    READ TRANSACTION FILE SECTOR                        00000570
                 (ELSE)                                                 00000580
                 ENDIF                                                  00000590
04FM             INDEX TO TRANSACTION RECORD IN SECTOR                  00000600
13FM             IF INPUT TRANS. NO. NOT EQUAL TO REC. TRANS. NO. THEN  00000610
13FM                SET COMPLETION CODE = TRANSACTION ALREADY ON FILE   00000620
                 (ELSE)                                                 00000630
                 ENDIF                                                  00000640
              (ELSE)                                                    00000650
              ENDIF                                                     00000660
           (ELSE)                                                       00000670
           ENDIF                                                        00000680
31FM       GET NO. OF TRANSACTIONS ADDED THUSFAR                        00000690
           IF NUMBER OF TRANS. G.T. OR E.Q. MAX. THEN                   00000700
13FM          SET COMPLETION CODE = TRANS. FILE SPACE EXHAUSTED         00000710
           (ELSE)                                                       00000720
           ENDIF                                                        00000730
13FM       IF COMPLETION CODE = SUCCESSFUL THEN                         00000740
31FM          ADD 1 TO NUMBER OF TRANSACTIONS ADDED THUSFAR             00000750
03FM          BUILD TRANSACTION DIRECTORY RECORD IN SECTOR              00000760
31FM          GET NUMBER OF ORDERS FOUND IN BATCH THUSFAR               00000770
```

```
            DOWHILE NUMBER OF ORDERS NOT ZERO                                  00000780
29FM           GET FIRST/NEXT ORDER ID FROM TABLE                              00000790
13FM           IF TABLE ORDER ID = INPUT ORDER ID THEN                         00000800
13FM              INDEX TO STATUS COUNTER FOR THIS ORDER                       00000810
29FM              ADD 1 TO NUMBER OF TRANSACTIONS OF THIS STATUS               00000820
               (ELSE)                                                          00000830
               ENDIF                                                           00000840
            ENDDO                                                              00000850
29FM        IF ORDER ID NOT FOUND IN TABLE THEN                                00000860
               IF NO. OF ORDERS IN TABLE G.T. OR E.Q. MAX. THEN                00000870
13FM              SET COMPLETION CODE = ORDER FILE SPACE EXHAUSTED             00000880
               ELSE                                                            00000890
29FM              ADD INPUT ORDER ID TO TABLE                                  00000900
                  CLEAR ALL STATUS COUNTS                                      00000910
13FM              INDEX TO STATUS COUNTER FOR THIS ORDER                       00000920
29FM              SET NUMBER OF TRANSACTIONS OF THIS STATUS = 1                00000930
                  ADD 1 TO NUMBER OF ORDERS FOUND IN BATCH THUS FAR            00000940
               ENDIF                                                           00000950
            (ELSE)                                                             00000960
            ENDIF                                                              00000970
13FM        IF COMPLETION CODE = SUCCESSFUL THEN                               00000980
               SAVE RELATIVE ORDER NUMBER                                      00000990
04FM           PLACE NEW TRANS. RECORD INTO CURRENT SECTOR BUFFER               00001000
13FM                                                                           00001010
13FM           IF THIS IS THE LAST TRANSACTION IN THE BATCH THEN                00001020
04FM              COMPUTE THE TRANSACTION FILE SECTOR                          00001030
                  WRITE THE CURRENT SECTOR BUFFER TO THE TRANS. FILE           00001040
29FM              GET NUMBER OF ORDERS IN ORDER TABLE                          00001050
                  DOUNTIL NUMBERS OF ORDERS = ZERO                             00001060
                     DOUNTIL ORDER SECTOR BUILT OR NUMBER OF -                 00001070
                     - ORDERS = ZERO                                           00001080
29FM                    GET NEXT DATA FOR NEXT ORDER FROM TABLE                00001090
05FM                    BUILD ORDER RECORD IN CURRENT SECTOR BUFFER             00001100
29FM                                                                           00001110
31FM                    SUBTRACT 1 FROM NUMBER OF ORDERS                       00001120
                     ENDDO                                                     00001130
05FM                 COMPUTE ORDER FILE SECTOR FOR CURRENT -                   00001140
                     - SECTOR BUFFER                                           00001150
                     WRITE THE CURRENT SECTOR BUFF. TO THE ORDER FILE          00001160
                  ENDDO                                                        00001170
03FM              WRITE THE BUFFERED TRANS. DIR. SECTOR TO DISK                00001180
01FM              COMPUTE THE BATCH FILE SECTOR                                00001190
                  READ THE BATCH FILE SECTOR                                   00001200
01FM              INDEX INTO SECTOR FOR BATCH RECORD                           00001210
01FM              PLACE THE NUMBER OF TRANS. IN THE BATCH RECORD               00001220
01FM              PLACE THE NUMBER OF ORDERS IN THE BATCH RECORD               00001230
                  WRITE THE BATCH SECTOR                                       00001240
09FM              GET A F.M. CALL BUFFER                                       00001250
                  IF ONE RECEIVED THEN                                         00001260
38FM                 SET UP BUFFER FOR ADD COMPLETE CALL                       00001270
08FM                 CALL THE FILE MANAGER                                     00001280
08FM                 IF F.M. RETURN CODE = SUCCESSFUL THEN                     00001290
09FM                    RETURN CALL BUFFER TO FREE POOL                        00001300
                        IF FREE POOL FULL THEN                                 00001310
54OI                       PANIC                                               00001320
                        (ELSE)                                                 00001330
                        ENDIF                                                  00001340
                     ELSE                                                      00001350
54OI                    PANIC                                                  00001360
                     ENDIF                                                     00001370
                  ELSE                                                         00001380
54OI                 PANIC                                                     00001390
                  ENDIF                                                        00001400
               ELSE                                                            00001410
04FM              IF THE CURRENT TRANS. FILE SECTOR BUFFER FULL THEN           00001420
04FM                 COMPUTE THE TRANSACTION FILE SECTOR                       00001430
                     WRITE THE CURRENT SECTOR BUFFER TO THE -                  00001440
                     - TRANSACTION FILE                                        00001450
                  (ELSE)                                                       00001460
                  ENDIF                                                        00001470
               ENDIF                                                           00001480
            (ELSE)                                                             00001490
            ENDIF                                                              00001500
```

```
            (ELSE)                                                          00001510
            ENDIF                                                           00001520
         ENDIF                                                              00001530
13FM     IF COMPLETION CODE = TRANSACTION ALREADY ON FILE, -                00001540
         - TRANSACTION FILE SPACE EXHAUSTED, OR ORDER FILE -                00001550
         - SPACE EXHAUSTED THEN                                             00001560
02FM        INDEX INTO RESIDENT BATCH FILE BY RELATIVE BATCH NO.            00001570
02FM        SET RESIDENT BATCH RECORD STATUS = DELETING                     00001580
09FM        GET A F.M. CALL BUFFER                                          00001590
            IF ONE RECEIVED THEN                                            00001600
14FM           SET RELATIVE BATCH NO. IN BATCH DELETION CALL BUFFER         00001610
08FM           CALL FILE MANAGER TO DELETE THE BATCH                        00001620
08FM           IF ERROR RETURNED FROM CALL THEN                             00001630
540I              PANIC                                                     00001640
               ELSE                                                         00001650
09FM              RETURN CALL BUFFER TO FREE POOL                           00001660
                  IF FREE POOL FULL THEN                                    00001670
540I                 PANIC                                                  00001680
                  (ELSE)                                                    00001690
                  ENDIF                                                     00001700
               ENDIF                                                        00001710
            ELSE                                                            00001720
540I           PANIC                                                        00001730
            ENDIF                                                           00001740
         (ELSE)                                                             00001750
         ENDIF                                                              00001760
13FM     RETURN TO CALLER                                                   00001770
                                                                            00000010
                                                                            00000020
                                                                            00000030
                                                                            00000040
                                                                            00000050
         READ BATCH RECORD VIA BATCH I.D.                                   00000060
                                                                            00000070
                                                                            00000080
                                                                            00000090
                           - MODULE HISTORY -                               00000100
                                                                            00000110
                                                                            00000120
            PROJECT:       DAS                  76-01709                    00000130
                                                                            00000140
            SUB-SYSTEM:    FILE MANAGER                                     00000150
                                                                            00000160
            MODULE:        FMBID                34                          00000170
                                                                            00000180
                                                                            00000190
                           - MODULE ABSTRACT -                              00000200
                                                                            00000210
                                                                            00000220
            THIS SUBROUTINE PROCESSES THE READ BATCH FILE RECORD            00000230
         VIA BATCH ID FILE MANAGER CALL CODE.  IT SEARCHES THE              00000240
         RESIDENT BATCH FILE TO CHECK THE VALIDITY AND EXISTANCE            00000250
         OF THE BATCH ID INPUT, RESOLVES THE RELATIVE BATCH                 00000260
         NUMBER, AND ACCESSES THE BATCH FILE ON DISK TO READ                00000270
         THE INDICATED BATCH RECORD AND COPY THE DATA INTO THE              00000280
         CALLER'S BUFFER.                                                   00000290
18FM     GET INPUTS                                                         00000300
33FM     CALL ROUTINE TO SEARCH RESIDENT BATCH FILE FOR INPUT BATCH I.D.    00000310
33FM     IF BATCH FOUND IN RESIDENT BATCH FILE THEN                         00000320
33FM        GET RELATIVE BATCH NUMBER                                       00000330
28FM        READ BATCH FILE SECTOR CONTAINING INDICATED BATCH RECORD        00000340
01FM        COPY BATCH RECORD INTO CALLER'S BUFFER                          00000350
18FM                                                                        00000360
         ELSE                                                               00000370
18FM        SET COMPLETION CODE = BATCH NOT ON FILE                         00000380
         ENDIF                                                              00000390
18FM     RETURN TO CALLER                                                   00000400
                                                                            00000010
                                                                            00000020
                                                                            00000030
                                                                            00000040
                                                                            00000050
         SET UP BATCH DELETION QUEUE FOR FMDLD                              00000060
                                                                            00000070
                                                                            00000080
```

```
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                  73-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      FILE MANAGER                                        00000150
                                                                        00000160
   MODULE:          FMDEL                32                          00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
                                                                        00000220
                                                                        00000230
   THIS MODULE RECEIVES ONE OR MORE RELATIVE BATCH NUMBERS              00000240
FOR BATCHES TO BE DELETED FROM THE SYSTEM. IT THEN PERFORMS             00000250
VALIDITY CHECKS ON THE INPUTS, MAKES THE NECESSARY BATCH                00000260
STATUS ADJUSTMENTS, AND PUTS THE BATCH NUMBERS INTO A                   00000270
WORK QUEUE FOR THE BATCH DELETION TASK (FMDLD).                         00000280
14FM  GET RELATIVE BATCH NUMBER(S) INPUT                                00000290
14FM  DOUNTIL ALL INPUTS VERIFIED OR INVALID INPUT DETECTED             00000300
14FM     GET FIRST(NEXT) REL. BATCH NO. IN INPUT LIST                   00000310
         IF RELATIVE BATCH NUMBER IN RANGE THEN                         00000320
02FM        GET BATCH RECORD IN RESIDENT BATCH FILE                     00000330
02FM        IF RECORD STATUS = UNUSED,ACTIVE,OR ADDING THEN             00000340
14FM           SET RETURN COMPLETION CODE TO "INVALID REC. NO."         00000350
            (ELSE)                                                      00000360
            ENDIF                                                       00000370
         ELSE                                                           00000380
14FM        SET RETURN COMPLETION CODE TO "INVALID REC. NO."            00000390
         ENDIF                                                          00000400
      ENDDO                                                             00000410
14FM  IF ALL INPUTS WERE FOUND TO BE VALID THEN                         00000420
14FM     DOUNTIL ALL BATCHES IN INPUT LIST ARE PROCESSED                00000430
14FM        GET FIRST(NEXT) REL. BATCH NO. FROM LIST                    00000440
02FM        GET BATCH RECORD IN RESIDENT BATCH FILE                     00000450
02FM        SET STATUS IN RECORD TO "DELETING"                          00000460
32FM        PUT RELATIVE BATCH NO. INTO QUEUE FOR DELETION TASK         00000470
            IF QUEUE IS FULL THEN                                       00000480
35FM           PANIC!                                                   00000490
            (ELSE)                                                      00000500
            ENDIF                                                       00000510
28FM        READ SECTOR FROM BATCH FILE FOR INDICATED BATCH             00000520
01FM        INDEX TO INDICATED BATCH RECORD IN SECTOR                   00000530
01FM        SET STATUS IN RECORD TO "DELETING"                          00000540
28FM        WRITE MODIFIED SECTOR BACK TO BATCH FILE ON DISK            00000550
         ENDDO                                                          00000560
      (ELSE)                                                            00000570
      ENDIF                                                             00000580
14FM  RETURN                                                            00000590
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
CHANGE BATCH STATUS                                                     00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                    - MODULE HISTORY -                                  00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                  73-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      FILE MANAGER                                        00000150
                                                                        00000160
   MODULE:          FMCBS                35                          00000170
                                                                        00000180
                                                                        00000190
                    - MODULE ABSTRACT -                                 00000200
                                                                        00000210
                                                                        00000220
                                                                        00000230
```

```
          THIS SUBTOUTINE PROCESSES THE CHANGE BATCH STATUS FILE          00000240
        MANAGER CALL CODE. IT SEARCHES THE BATCH FILE FOR THE             00000250
        INPUT BATCH I.D. AND IF FOUND CHANGES THE BATCH STATUS            00000260
        TO THE INPUT STATUS.                                              00000270
15FM    GET INPUTS                                                        00000280
33FM    CALL ROUTINE TO SEARCH RESIDENT BATCH FILE FOR INPUT BATCH I.D.   00000290
33FM    IF BATCH NOT FOUND THEN                                           00000300
15FM       SET COMPLETION CODE = BATCH NOT ON FILE                        00000310
        ELSE                                                              00000320
15FM       IF INPUT BATCH STATUS INVALID THEN                             00000330
15FM          SET COMPLETION CODE = INVALID STATUS                        00000340
           ELSE                                                           00000350
01FM          COMPUTE BATCH FILE SECTOR FROM RELATIVE BATCH NUMBER        00000360
28FM          READ THE BATCH FILE SECTOR                                  00000370
01FM          INDEX INTO SECTOR FOR BATCH RECORD                          00000380
01FM          SET RECORD STATUS = INPUT STATUS                            00000390
28FM          WRITE THE BATCH FILE SECTOR                                 00000400
02FM          INDEX INTO RESIDENT BATCH FILE BY RELATIVE BATCH NUMBER     00000410
02FM          SET RESIDENT BATCH RECORD STATUS = INPUT STATUS             00000420
           ENDIF                                                          00000430
        ENDIF                                                             00000440
15FM    RETURN TO CALLER                                                  00000450
                                                                          00000010
                                                                          00000020
                                                                          00000030
                                                                          00000040
                                                                          00000050
        TASK TO PROCESS BATCH DELETIONS                                   00000060
                                                                          00000070
                                                                          00000080
                                                                          00000090
                        - MODULE HISTORY -                                00000100
                                                                          00000110
                                                                          00000120
           PROJECT:     DAS                 73-01709                     00000130
                                                                          00000140
           SUB-SYSTEM:  FILE MANAGER                                      00000150
                                                                          00000160
           MODULE:      FMDLD               44                           00000170
                                                                          00000180
                                                                          00000190
                        - MODULE ABSTRACT -                               00000200
                                                                          00000210
                                                                          00000220
           THIS MODULE IS A TASK WHICH OBTAINS RELATIVE BATCH NUMBERS     00000230
        FROM A QUEUE WHICH INDICATE THE BATCHES TO BE DELETED.            00000240
        IT THEN WRITES AN INITIALIZED SECTOR TO ALL SECTORS IN THE        00000250
        TRANS. FILE THAT ARE ASSOCIATED WITH THE BATCH BEING DELETED.     00000260
        FINALLY THIS TASK MAKES THE APPROPRIATE STATUS ADJUSTMENT         00000270
        ENTRIES IN THE DISK AND RESIDENT BATCH FILES.                     00000280
        DO UNTIL CPU STOPPED                                              00000290
32FM       DOWHILE DATA STILL AVAILABLE IN QUEUE                          00000300
32FM          GET NEXT RELATIVE BATCH NUMBER FROM QUEUE                   00000310
04FM          COMPUTE FIRST TRANSACTION SECTOR FOR THIS BATCH             00000320
04FM          DOUNTIL ALL TRANSACTION SECTORS INITIALIZED                 00000330
34FM             ENQUEUE FILE MANAGER RESOURCE                            00000340
04FM             WRITE INITIALIZED TRANSACTION SECTOR                     00000350
34FM             DEQUEUE FILE MANAGER RESOURCE                            00000360
              ENDDO                                                       00000370
34FM          ENQUEUE FILE MANAGER RESOURCE                               00000380
              GET RELATIVE BATCH NUMBER                                   00000390
              READ BATCH FILE SECTOR FOR INDICATED BATCH                  00000400
01FM          SET STATUS TO UNUSED IN INDICATED BATCH RECORD              00000410
              WRITE MODIFIED SECTOR BACK TO BATCH FILE                    00000420
02FM          SET STATUS TO UNUSED IN RESIDENT BATCH FILE RECORD          00000430
34FM          DEQUEUE FILE MANAGER RESOURCE                               00000440
           ENDDO                                                          00000450
           DELAY FOR AWHILE                                               00000460
        ENDDO                                                             00000470
                                                                          00000010
                                                                          00000020
                                                                          00000030
                                                                          00000040
                                                                          00000050
```

```
ADD BATCH RECORD TO FILES                                                  00000060
                                                                           00000070
                                                                           00000080
                                                                           00000090
                    - MODULE HISTORY -                                     00000100
                                                                           00000110
                                                                           00000120
   PROJECT:         DAS                    79-01709                       00000130
                                                                           00000140
   SUB-SYSTEM:      FILE MANAGER                                          00000150
                                                                           00000160
   MODULE:          FMABR                  31                             00000170
                                                                           00000180
                                                                           00000190
                    - MODULE ABSTRACT -                                    00000200
                                                                           00000210
                                                                           00000220
     THIS SUBROUTINE PROCESSES THE ADD BATCH RECORD FILE MANAGER           00000230
   CALL CODE. IT FINDS A FREE BATCH RECORD AND ADDS THE NEW                00000240
   BATCH RECORD TO THE FILES.                                              00000250
12FM  GET INPUTS                                                           00000260
      SET RECORD POINTER = ZERO                                            00000270
      SET RECORD USE COUNT = ZERO                                          00000280
02FM  DOUNTIL END OF RESIDENT BATCH FILE OR ERROR FOUND                    00000290
         GET FIRST/NEXT RESIDENT BATCH RECORD                              00000300
02FM     IF RECORD STATUS NOT UNUSED OR DELETING THEN                      00000310
            ADD 1 TO RECORD USE COUNT                                      00000320
         (ELSE)                                                            00000330
         ENDIF                                                             00000340
12FM     IF INPUT BATCH I.D. = RECORD BATCH I.D. THEN                       00000350
02FM                                                                       00000360
02FM        IF RECORD STATUS NOT DELETING OR UNUSED THEN                   00000370
12FM           SET COMPLETION CODE = BATCH ALREADY ON FILE                 00000380
            ELSE                                                           00000390
02FM           IF RECORD STATUS = UNUSED THEN                              00000400
                  SAVE RECORD NO. IN RECORD POINTER                        00000410
               (ELSE)                                                      00000420
               ENDIF                                                       00000430
            ENDIF                                                          00000440
         ELSE                                                              00000450
02FM        IF RECORD STATUS = ADDING THEN                                 00000460
12FM           SET COMPLETION CODE = BATCH ALREADY BEING ADDED             00000470
            ELSE                                                           00000480
02FM           IF RECORD STATUS = UNUSED THEN                              00000490
                  SAVE RECORD NO. IN RECORD POINTER                        00000500
               (ELSE)                                                      00000510
               ENDIF                                                       00000520
            ENDIF                                                          00000530
         ENDIF                                                             00000540
      ENDDO                                                                00000550
12FM  IF COMPLETION CODE = SUCCESSFUL THEN                                 00000560
         IF RECORD COUNT G.T. OR E.Q. 10 THEN                              00000570
            SET RECORD POINTER = ZERO                                      00000580
         (ELSE)                                                            00000590
         ENDIF                                                             00000600
         IF RECORD POINTER NOT ZERO THEN                                   00000610
01FM        COMPUTE BATCH FILE SECTOR ADDRESS FROM RECORD_POINTER          00000620
28FM        READ THE BATCH FILE SECTOR                                     00000630
01FM        PLACE NEW BATCH RECORD INTO SECTOR                             00000640
01FM        SET NEW BATCH RECORD STATUS = ADDING                           00000650
01FM        SET NEW BATCH RECORD COUNTS = ZERO                             00000660
28FM        WRITE THE UPDATED BATCH SECTOR                                 00000670
02FM        BUILD RESIDENT BATCH RECORD WITH STATUS = ADDING               00000680
         ELSE                                                              00000690
12FM        SET COMPLETION CODE = BATCH FILE SPACE EXHAUSTED               00000700
         ENDIF                                                             00000710
      (ELSE)                                                               00000720
      ENDIF                                                                00000730
12FM  RETURN TO CALLER                                                     00000740
                                                                           00000010
                                                                           00000020
                                                                           00000030
                                                                           00000040
                                                                           00000050
SEARCH RESIDENT BATCH FILE SUBROUTINE                                      00000060
```

```
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                     76-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      FILE MANAGER                                        00000150
                                                                        00000160
   MODULE:          FMSRB                   43                          00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
   THIS SUBROUTINE SEARCHES THE RESIDENT BATCH FILE IN                  00000230
$SYSCOM FOR THE BATCH I.D. INPUT BY THE CALLER.                         00000240
IT WILL RETURN A "0" IF THE BATCH I.D. IS NOT ON FILE,                  00000250
OR THE RELATIVE BATCH NO. IF THE BATCH IS FOUND.                        00000260
33FM  GET INPUTS                                                        00000270
      SET RETURN CODE = 0                                               00000280
      DOUNTIL END OF DATA OR BATCH I.D. FOUND                           00000290
02FM     GET FIRST/LAST RECORD FROM RESIDENT BATCH FILE                 00000300
02FM     IF BATCH I.D. FROM RECORD = BATCH I.D. INPUT THEN               00000310
33FM                                                                    00000320
02FM        IF BATCH STATUS NOT UNUSED, ADDING, OR DELETING THEN        00000330
33FM           SET RETURN CODE = RELATIVE BATCH NUMBER                  00000340
            (ELSE)                                                      00000350
            ENDIF                                                       00000360
         (ELSE)                                                         00000370
         ENDIF                                                          00000380
      ENDDO                                                             00000390
33FM  RETURN TO CALLER                                                  00000400
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
COLD START DATABASE INITIALIZATION                                      00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
   PROJECT:         DAS                     76-01709                    00000130
                                                                        00000140
   SUB-SYSTEM:      FILE MANAGER                                        00000150
                                                                        00000160
   MODULE:          FMINT                   33                          00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
   THIS MODULE IS CALLED BY FMGR DURING SYSTEM COLD START TO            00000230
START THE DELETION PROCESS FOR ANY BATCHES THAT HAVE A STATUS           00000240
OF "ADDING OR DELETING".  THIS MODULE ALSO BUILDS THE RESIDENT          00000250
BATCH FILE IN $SYSCOM FROM CURRENT DATA IN THE DISK FILE "BATCH"        00000260
AND COPIES THE INFORMATION FROM THE DISK FILE "XLATE" INTO              00000270
THE RESIDENT TRANSLATION FILE, ALSO IN $SYSCOM, AND COPIES              00000280
INFORMATION FROM THE DISK FILE "ORDER" INTO THE RESIDENT                00000290
ORDER FILE FOUND IN THE FILE MANAGER SUB-SYSTEM.                        00000300
24FM  GET INPUTS                                                        00000310
28FM  BUILD THE RESIDENT ORDER FILE                                     00000320
01FM  DOUNTIL ALL SECTORS IN BATCH FILE ARE PROCESSED                   00000330
28FM     READ FIRST(NEXT) BATCH FILE SECTOR                             00000340
01FM     DOUNTIL ALL RECORDS IN SECTOR ARE PROCESSED                    00000350
01FM        GET FIRST(NEXT) RECORD IN SECTOR                            00000360
01FM        IF STATUS IN RECORD = ADDING OR DELETING THEN               00000370
01FM           SET STATUS IN RECORD TO DELETING                         00000380
32FM           PUT RELATIVE BATCH NUMBER IN QUEUE FOR DELETION TASK     00000390
            (ELSE)                                                      00000400
            ENDIF                                                       00000410
```

```
02FM          BUILD RESIDENT BATCH FILE RECORD IN $SYSCOM                00000420
              ENDDO                                                     00000430
28FM          WRITE SECTOR BACK TO BATCH FILE                           00000440
           ENDDO                                                        00000450
28FM       READ SORT TRANSLATION FILE SECTOR INTO $SYSCOM               00000460
06FM                                                                    00000470
24FM       RETURN                                                       00000480
                                                                        00000010
                                                                        00000020
                                                                        00000030
                                                                        00000040
                                                                        00000050
           FILE MANAGER CONTROL                                         00000060
                                                                        00000070
                                                                        00000080
                                                                        00000090
                        - MODULE HISTORY -                              00000100
                                                                        00000110
                                                                        00000120
              PROJECT:        DAS                  73-01709             00000130
                                                                        00000140
              SUB-SYSTEM:     FILE MANAGER                              00000150
                                                                        00000160
              MODULE:         FMGR                 29                   00000170
                                                                        00000180
                                                                        00000190
                        - MODULE ABSTRACT -                             00000200
                                                                        00000210
                                                                        00000220
              THIS MODULE CONTROLS ALL INPUT TO THE FILE MANAGER. IT    00000230
           WAITS ON THE FILE MANAGER ENTRY EVENT TO BE POSTED.          00000240
           IT THEN CHECKS FOR A VALID CALL CODE FOUND IN THE BUFFER     00000250
           PASSED TO IT BY THE CALLER. IF THE CALL CODE IS VALID        00000260
           THE SUBROUTINE THAT PROCESSES THAT CODE IS CALLED.           00000270
           WHEN THE SUBROUTINE RETURNS, THIS MODULE WILL POST           00000280
           THE COMPLETION EVENT BACK TO THE CALLER. THE INFORMATION     00000290
           REQUESTED BY THE CALLER IS HELD IN THE BUFFER THAT WAS       00000300
           PASSED TO THE FILE MANAGER UPON ENTRY.                       00000310
           ATTACH BATCH DELETION WORK TASK (FMDLD)                      00000320
           DOUNTIL CPU STOPPED                                          00000330
25FM          WAIT FOR THE FILE MANAGER ENTRY EVENT TO BE POSTED        00000340
09FM          IF THE CALL CODE FOUND IN THE BUFFER IS WITHIN RANGE THEN 00000350
26FM             GET ADDRESS OF CALL CODE DISPATCH TABLE                00000360
                 INDEX INTO DISPATCH TABLE BY CONTROL CODE              00000370
                 GET WORK SUBROUTINE ADDRESS FROM DISPATCH TABLE        00000380
                 PLACE ADDRESS INTO SUBROUTINE CALL                     00000390
                 SET FIRST PARAMETER = INPUT BUFFER ADDRESS             00000400
09FM             SET COMPLETION CODE IN BUFFER = SUCCESSFUL             00000410
                 CALL THE WORK SUBROUTINE                               00000420
              ELSE                                                      00000430
09FM             SET COMPLETION CODE = INVALID CALL CODE                00000440
              ENDIF                                                     00000450
27FM          POST ADDRESS OF INPUT BUFFER BACK TO THE CALLER           00000460
           ENDDO                                                        00000470
           PROGSTOP                                                     00000480
0026                ;
0027                ; L E G E N D
0028                ;#    NUMBER, NUMBER OF
0029                ;AD   ADDRESS       CH   CHANNEL       CHAR CHARACTER    CM   COMMAND
0030                ;COMM COMMUNICATION                     COUT COMMAND OUT
0031                ;CP   COMPUTE       CR   CARRIAGE RETURN                 CT   CONTROL
0032                ;DC   DECODE, DIVERT-COMPLETE PHOTO
0033                ;DEC  DECREMENT     DIN  DATA IN DOUT DATA OUT
0034                ;DT   DATA          DV   DIVERT
0035                ;EC   ENCODE        EOF  END OF FILE  ER   ERROR
0036                ;ID   INDUCT PHOTO INC   INCREMENT     INT  INTERRUPT
0037                ;JDV  DIVERT NUMBER                    JINDU INDUCT PHOTO NUMBER
0038                ;JKB  KEYBOARD NUMBER
0039                ;JPPI PULSE POSITION INDICATOR NUMBER, =0 MEANS REAL TIME CLOCK
0040                ;JUD  UPDATE PHOTO NUMBER
0041                ;KB   KEYBOARD
0042                ;LAQ  LOOK-AHEAD QUEUE
0043                ;LE   LENGTH        LF   LINE FEED, LANE FULL PHOTO
0044                ;M    MEMORY        MG   MERGE PHOTO  MS   MOST SIGNIFICANT
0045                ;MUZ  MOVING UPDATABLE ZONE
0046                ;NL   NEW LINE
0047                ;O    OPTION
```

```
0048                ;PC   PROGRAM COUNTER, PROCESS       PCR  PROCESSOR
0049                ;PLE  PACKAGE LENGTH
0050                ;PP   PACKAGE PRESENT PHOTO          PPI  PULSE POSITION INDICATOR
0051                ;PR   PRINT
0052                ;RC   RECORD       RDC  SILENT 700 REMOTE DEVICE CT    RE   READ
0053                ;RLS  RELEASE MECHANISM
0054                ;RTC  REAL TIME CLOCK
0055                ;SEC  SECOND       SIN  STATUS IN
0056                ;SP   STACK POINTER
0057                ;T    TIME         TE   TRAILING EDGE PHOTO
0058                ;TR   TRAINING PHOTO
0059                ;TTY  TELETYPE
0060                ;UD   UPDATE PHOTO
0061                ;WR   WRITE        XM   TRANSMIT     XQ   EXECUTE
0062                ;
0063                ; S U B R O U T I N E S  ( M O D U L E S )
0064                ;ALL FLAGS ARE ALWAYS EFFECTED UNLESS OTHERWISE STATED.
0065                ;  REGISTERS A, B, C, D, E, H, L
0066                ;  EFFECTED, OR SP, PC EFFECTED UNEXPECTEDLY, WILL BE MENTIONED.
0067                ;
0068                ; C O M M E N T S
0069                ;COMMENTS ARE GROUPED BY SEMICOLONS.  CONTINUATION OF THE SAME
0070                ;  COMMENT WILL BE INDENTED 2 COLUMNS.
0071                ;
0072                ;
0073                ; R E F E R E N C E S
0074                ;
0075                ;(1) "APPLICATION MANUAL FOR PROGRAMMER SORT CONTROLLER"
0076                ;
0077                ;(2) PROGRAMMABLE SORT CONTROLLER SOFTWARE ORGANIZATION CHARTS
0078                ;    AND SOFTWARE MANUAL
0079                ;
0080                ;*************************************************************************
0081                ;ASSEMBLER PRESET VALUES
0082  4000          RAMAD    EQU   4000H      ;RAM STARTING ADDRESS
0083  0001          CVY01    EQU   1          ;FLAG FOR CONVEYOR RELAY CH
0084  4300          DCBPC    EQU   4300H      ;DIVERT-COMPLETE PHOTO-BLOCKING PROCESSOR
0085  03DD          EXEC     EQU   03DDH
0086  03EB          DEBUG    EQU   03EBH
0087  0966          COLDS    EQU   0966H
0088  0C19          DEVII    EQU   0C19H
0089  0C85          APEND    EQU   0C85H
0090  0CA2          CASSW    EQU   0CA2H
0091  0CB8          CLAKB    EQU   0CB8H
0092  0CC4          CLRBP    EQU   0CC4H
0093  0CC9          CLRKB    EQU   0CC9H
0094  0CDF          CLDP     EQU   0CDFH
0095  0D17          CPKBP    EQU   0D17H
0096  0D1B          AD16A    EQU   0D1BH
0097  0D25          CPLFM    EQU   0D25H
0098  0D28          CPMSK    EQU   0D28H
0099  0D3E          DC2BD    EQU   0D3EH
0100  0D51          DCBCD    EQU   0D51H
0101  0D81          DCDV     EQU   0D81H
0102  0D9E          DIGPC    EQU   0D9EH
0103  0DEA          DSPLA    EQU   0DEAH
0104  0DF4          DSPLM    EQU   0DF4H
0105  3A08          DELAY    EQU   3A08H
0106  3A89          DPHL     EQU   3A89H
0107  3A70          DPBD     EQU   3A70H
0108  0E1C          ERMPC    EQU   0E1CH
0109  0E01          ECDIG    EQU   0E01H
0110  0F08          ERPC     EQU   0F08H
0111  0F13          ERRPC    EQU   0F13H
0112  0F76          FLASH    EQU   0F76H
0113  0F91          GDTRE    EQU   0F91H
0114  0FA2          GITRE    EQU   0FA2H
0115  0FAC          GUTRE    EQU   0FACH
0116  0FAD          GUTR1    EQU   0FADH
0117  0FBA          GTREP    EQU   0FBAH
0118  0FCB          GETFS    EQU   0FCBH
0119  100E          GETKB    EQU   100EH
0120  1065          GETPH    EQU   1065H
0121  1097          LNKFS    EQU   1097H
0122  10C4          MOVE     EQU   10C4H
0123  10CF          MOVEB    EQU   10CFH
0124  10D8          MTKBB    EQU   10D8H
0125  1109          OFCVY    EQU   1109H
0126  110F          OPUT     EQU   110FH
0127  113F          PHPPB    EQU   113FH
0128  114F          PUTFS    EQU   114FH
0129  116C          PUTKB    EQU   116CH
0130                ;
0131                ;
0132                ; P R O M   D A T A
0133                ;
0134                ;
0135                ;FACTORY PRESET PARAMETERS.
```

```
0136                    ;
0137   0000             ORG   3D00H
0138   3D00 004B  RAMT1: DW    4B00H     ;COLD START SP VALUE, = RAM TOP + 1
0139   3D02 02    NPPI:  DB    2         ;# PPI'S
0140   3D03 02    NINDU: DB    2         ;# INDUCTION STATIONS
0141   3D04 06    NUPDA: DB    6         ;# UPDATE PHOTOS
0142   3D05 05    NDVT:  DB    5         ;# DIVERTS
0143   3D06 03    P5080: DB    3         ;STARTING ADDRESS OF 1ST PM5080 BOARD;
0144   3D07 FF           DB    0FFH      ;  MUST BE 0FFH (FOR COUT OR SIN);
0145   3D08 02    NKB:   DB    2         ;  TOTAL # KB'S
0146   3D09 01    MAKB:  DB    1         ;MASTER KB # FOR SYSTEM OPERATIONS & CASSETTES
0147   3D0A 06    NDPLA: DB    6         ;# DISPLAY DIGITS, MUST BE 2, 4 OR 6
0148   3D0B 00    NXLTB: DB    0         ;# TRANSLATE TABLES, <= 16,
0149                                     ;  0 MEANS NO TRANSLATION OPTION
0150   3D0C 01    TBSIZ: DB    1         ;TRANSLATE TABLE SIZE IN BYTES / 256,
0151                                     ;  MUST BE 1, 2, 4, 8, 10H, 20H, OR 40H
0152   3D0D       COMAX: DS    3         ;ENTRY CODE UPPER BOUND IN BCD
0153   3D10 FFFF  ERO:   DW    0FFFFH    ;ERROR MESSAGES
0154   3D12              DS    6
0155   3D18 FFFF         DW    0FFFFH    ;DIAG MODE ERROR MESSAGE MASK
0156   3D1A              DS    6
0157   3D20 FFFF         DW    0FFFFH    ;TRAIN MODE ERROR MESSAGE MASK
0158   3D22 00    ERMKI: DB    00H       ;PSC INPUT MESSAGE MASK BYTE 0
0159   3D23 00           DB    00H       ;BYTE 1
0160   3D24 08           DB    08H       ;BYTE 2
0161   3D25 09           DB    09H       ;BYTE 3
0162   3D26 00           DB    00H       ;BYTE 4
0163   3D27 00           DB    00H       ;BYTE 5
0164   3D28 00    ERMKO: DB    00H       ;PSC OUTPUT MESSAGE MASK BYTE 0
0165   3D29 03           DB    03H       ;BYTE 1
0166   3D2A 01           DB    01H       ;BYTE 2
0167   3D2B 11           DB    11H       ;BYTE 3
0168   3D2C 14           DB    14H       ;BYTE 4
0169   3D2D 00           DB    00H       ;BYTE 5
0170   3D2E FFFF         DW    0FFFFH    ;MUST BE ALL 1'S FOR FATAL ERRORS
0171   3D30 00    ERCHU: DB    0         ;ER CHUTE #
0172                                     ;  = 0 MEANS CONVEYOR END
0173   3D31 00    SIDEO: DB    0         ;SIDE INDUCT OPTION:
0174                                     ;  BIT 1 FOR AUTO-REINDUCT AFTER RECIRCULATION,
0175                                     ;  BIT 4 FOR SIDE-INDUCT MERGING
0176   3D32 30    SENDO: DB    30H       ;"SEND" OPTION CODE,
0177                                     ;  BITS 0 TO 3 (LS NIBBLE) FOR LENGTH OF FIXED-
0178                                     ;  LENGTH CODE WITHOUT "SEND" (0 MEANS OFF),
0179                                     ;  BIT 4 FOR ENTER-CODE-BY-"SEND",
0180                                     ;  BIT 5 FOR REPEAT-BY-"SEND".
0181   3D33 01    NICC:  DB    1         ;# INTER-COMPUTER COMMUNICATION CHANNELS
0182   3D34 01    DQLIM: DB    1
0183   3D35 0000         DW    0
0184   3D37 02    QLIMI: DB    2         ;LAQ SIZE LIMIT, MUST BE 1 TO 8
0185   3D38 00    DVTRS: DB    0         ;DIVERT RESET OPTION:
0186                                     ;  0 FOR MAINTAINED DV & NO RESET UNTIL
0187                                     ;  NEXT PACKAGE
0188                                     ;  1 FOR MEASURED RESET ACCORDING TO PACKAGE LE
0189                                     ;  2 FOR HARDWARE RESET INITIATED BY PULSE
0190   3D39 00    SCNRO: DB    0         ;SCANNER OPTIONS:
0191                                     ;  BIT 0 TO ACTIVATE DIVER # DEFAULT,
0192                                     ;  BIT 1 TO ACTIVATE TRANSLATE DEFAULT;
0193                                     ;  BIT 6 TO ACTIVATE PARITY CHECKING,
0194                                     ;  BIT 7 FOR ODD PARITY
0195   3D3A 00    SERDV: DB    0         ;AUTO JDV FOR SCANNER FAIL
0196                                     ;  =0 MEANS RECIRCULATION
0197   3D3B 00    DCO:   DB    0         ;DV-COMPLETE OPTION:
0198                                     ;  -0 MEANS NO DIVERT-COMPLETE
0199                                     ;  =1 MEANS 1 DIVERT-COMPLETE PHOTO PER DIVERT
0200                                     ;  = 2 MEANS UPDATE PHOTOS USED
0201                                     ;        AS DIVERT-COMPLETE PHOTOS
0202   3D3C 02    LFO:   DB    2         ;LANE FULL PHOTO OPTION CODE,
0203                                     ;  BIT 0 (LS BIT) FOR KB-INHIBIT,
0204                                     ;  BIT 1 FOR DV-INHIBIT,
0205                                     ;  BIT 4 FOR AUTO SEND AFTER CLEARING LF
0206   3D3D 00    LFEDV: DB    0         ;LANE FULL ERROR DIVERT (FUTURE EXPANSION)
0207   3D3E 0000  SYS9:  DW    0         ;MUST BE 0
0208   3D40 00    SYSTM: DB    0         ;MUD= LAST UD # FOR PPI #1
0209   3D41 01           DB    1         ;MUD= LAST UD # FOR PPI #2
0210   3D42 00           DB    0         ;MUST BE 0
0211   3D43 03           DB    3         ;MDV= LAST DV # FOR UD # 1
0212   3D44 00           DB    0         ;MUST BE 0
0213   3D45              DS    1         ;FID= FOLLOWING SIDE INDUCT # FOR UD # 1
0214                    ;
0215   3D46              ORG   3DC0H
0216   3DC0 0080  XLTBS: DW    8000H     ;TRANSLATE TABLE # 1 BASE ADDRESS
0217   3DC2 0090         DW    9000H     ;TRANSLATE TABLE # 2 AD
0218   3DC4 00A0         DW    0A000H    ;TRANSLATE TABLE # 3 AD
0219                    ;
0220   3DC6              ORG   3DF0H
0221   3DF0 004C  TRSDB: DW    4C00H
0222   3DF2 FF4F  TLAST: DW    4FFFH
0223                    ;
```

```
0224                    ; R A M   D A T A
0225                    ;  (PUT IN FRONT FOR CONDITIONAL MACRO EXPANSIONS)
0226                    ;
0227                    ;
0228                    ;FIELD SET PARAMETERS FOR RUNNING MODE.
0229                    ;
0230    3DF4                        ORG   RAMAD
0231    4000 0000       FLAGS:      DW    0           ;BIT 0 (LS BIT) TO ENABLE ALL ERROR MESSAGES;
0232                                                  ;BIT 7 (128) FOR AUTOMATIC INDUCTION
0233    4002 0000       SLIP:       DW    0           ;INERTIAL SLIPPAGE ALLOWANCE FOR
0234                                                  ;  HOT START
0235    4004 0000       PLEMX:      DW    0           ;T FROM RLS TO TEC2
0236                                                  ;  = MAX PLE + TE DELAY + SAFETY FACTOR
0237    4006 0000       HDSEN:      DW    0           ;HALF HOST MODE ENABLED
0238    4008 0000       DVH:        DW    0           ;FOR DVTRS=1, = MAX DIVERT HOLD TIME - PLE,
0239                                                  ;  FOR DVTRS=2, = PULSE LENGTH
0240    400A 0000       PPD:        DW    0           ;PP DWELL FOR PUTTING IN GAP
0241    400C 0000       GAP:        DW    0           ;ADDITIONAL RELEASE DELAY FOR A PACKAGE NOT
0242                                                  ;  FULLY STOPPED
0243                    DCTT:                         ;OVERALL MAX TIME FROM DIVERT SET
0244                                                  ;  TO DIVERT-COMPLETE PHOTO BLOCKED
0245                    DCZSZ:                        ;DC ZONE SIZE
0246    400E 0000       DCPSZ:      DW    0           ;MIN PACKAGE SIZE AT DC
0247    4010            TTBP:       DS    01E0H       ;FOR TIMING TABLES
0248                    ;
0249                    ;
0250    41F0                        ORG   RAMAD+0200H
0251                    ;
0252                    ;BREAKPOINT VARIABLES.
0253                    ;
0254    4200 0000       RS1AD:      DW    0           ;RESTART
0255    4202 EB03       RS2AD:      DW    DEBUG       ;  MODULE
0256    4204 0000       RS3AD:      DW    0           ;  AD'S
0257    4206 0000       RS4AD:      DW    0           ;
0258    4208 0000       RS5AD:      DW    0           ;
0259    420A 0000       RS6AD:      DW    0           ;
0260    420C 00         BPM:        DB    0           ;BREAKPOINT M CONTENT (TO BE RESTORED)
0261    420D 0000       ENTAD:      DW    0           ;"GO" COMMAND ENTRANCE AD
0262    420F 0000       REGAF:      DW    0           ;REGISTERS A,F VALUES
0263    4211 0000       REGBC:      DW    0           ;REGISTERS B,C VALUES
0264    4213 0000       REGDE:      DW    0           ;REGISTERS D,E VALUES
0265    4215 0000       REGHL:      DW    0           ;REGISTERS H,L VALUES
0266    4217 0000       BP:         DW    0           ;BREAKPOINT
0267    4219 00         CVCHN:      DB    0           ;COMPOSITE CHN # FOR CONVEYOR RELAY
0268                    ;
0269                    ;DEBUGGER AND CASSETTE VARIABLES.
0270                    ;
0271    421A 00         CMCH:       DB    0           ;COMMAND CHARACTER
0272    421B 00         JRC:        DB    0           ;RECORD INDEX
0273    421C 00         RCLE:       DB    0           ;RECORD LENGTH
0274    421D 0000       DUMP0:      DW    0           ;MEMORY DUMP START AD
0275    421F 0000       DUMP9:      DW    0           ;MEMORY DUMP END AD
0276    4221 0000       LOFFS:      DW    0           ;LOAD M OFFSET
0277    4223 0000       MISMP:      DW    0           ; POINTS TO MOST RECENT MISMATCH BYTE
0278                    ;
0279                    ;VALUES NOT EFFECTED BY COLD START.
0280                    ;
0281    4225 3C         MODE:       DB    '<'         ;'<' FOR RUN, '=' FOR DIAGNOSTIC,
0282                                                  ;  '>' FOR TRAINING, ':' FOR INTER-COMPUTER
0283    4226 08         MODEE:      DB    8           ;NOT=0 WILL ENABLE MODE TRANSITION
0284    4227 0000       FSP0:       DW    0           ;BASE AD OF FIELD-SET PARAMETERS
0285    4229 35         RS:         DB    35H         ;ASCII 5
0286    422A 35         LMR:        DB    35H         ;ASCII 5
0287    422B 00         ICMO:       DB    0           ;INTERCOMPUTER MODE ACTIVE FLAG
0288    422C 0200       TDLTM:      DW    0002H       ;MESSAGE TIMEOUT IN 0.1 SEC INCREMENTS
0289    422E 0F         VICLG:      DB    0FH         ;I/C LIGHT REFRESH TIME IN 0.1 SEC INCREMENTS
0290                    ;
0291                    ;TABLE POINTERS TO BE SET BY COLD START & NEVER CHANGED.
0292                    ;
0293                    CSRAM:
0294    422F 0000       PRET:       DW    0           ;PRE-INDUCT TIME TABLE: IF STRAIGHT MERGE
0295                                                  ;  THEN MAX T FROM RLS TO ID,
0296                                                  ;  ELSE UDBPC TO BMG
0297    4231 0000       TET:        DW    0           ;TE DELAY TABLE
0298    4233 0000       IDT:        DW    0           ;MIN T FROM ID-BLOCKING TO UD1-BLOCKING TABLE;
0299                                                  ;  IF SIDE-INDUCT, RELEASE TO UDB1 TIME
0300    4235 0000       ZNT:        DW    0           ;UDB1 TO UDB2 TIMING TABLE
0301    4237 0000       UDT:        DW    0           ;UPDATE PHOTO TIMING TABLE
0302    4239 0000       DVT:        DW    0           ;DIVERT TIMING TABLE
0303    423B 0000       DCT:        DW    0           ;MIN T LAST UD-BLOCKING TO DC-BLOCKING TABLE
0304                    ;
0305    423D 0000       LDVDB:      DW    0           ;LDV DATABASE IN PROM
0306    423F 0000       FIDDB:      DW    0           ;SIDE INDUCT # FOLLOWING UPDATE
0307    4241 0000       ICDB:       DW    0           ;INTER-COMPUTER DATABASE
0308    4243 0000       PPDB:       DW    0           ;PACKAGE-PRESENT DATABASE
0309    4245 0000       PPIDB:      DW    0           ;PPI DATABASE FOR TREE PARAMETERS
0310                                                  ;  PPI # 0 MEANS RTC
```

```
0311   4247 0000     KBDB:    DW    0          ;KB DATA BASE.
0312   4249 0000     KBIBF:   DW    0          ;  KB INPUT BUFFERS.
0313   424B 0000     KBOBF:   DW    0          ;  KB ECHO BUFFERS
0314   424D 0000     XLDB:    DW    0          ;TRANSLATE DATABASE
0315   424F 0000     RLSDB:   DW    0          ;RLS DATABASE
0316   4251 0000     UDDB:    DW    0          ;UPDA PHOTO DATABASE
0317   4253 0000     LFDB:    DW    0          ;LF PHOTO DATABASE
0318   4255 0000     DVDB:    DW    0          ;DIVERT DATABASE FOR DC
0319   4257 0000     DCDB:    DW    0          ;DV-COMPLETE DATABASE FOR DC
0320   4259 0000     OUTDB:   DW    0          ;PM5007 OUTPUT BOARD IMAGES
0321   425B 0000     FS:      DW    0          ;FREE STORAGE FOR TCB'S IN RUNNING MODE
0322                 ;
0323                 ;VALUES TO BE SENT IN COLD START & NEVER CHANGED.
0324                 ;
0325   425D AA55     RAM01:   DW    55AAH      ;FOR TESTING VALIDITY OF RAM
0326   425F AA55     RAM02:   DW    55AAH      ;  ON POWER-UP
0327   4261          HTSL:    DS    5          ;FOR PM5008 INITIALIZATION
0328   4266 04FF     MKBCO:   DW    0FF04H     ;MASTER KB COUNT AD
0329   4268 00       SL700:   DB    0          ;NOT=0 MEANS MASTER KB IS SILENT 700
0330   4269 0000     K5080:   DW    0          ;PM5080 SIN AD FOR 1ST KB
0331   426B 00       N5080:   DB    0          ;# PM5080 BOARDS
0332   426C 00       P5008:   DB    0          ;1ST PM5008 BOARD AD
0333   426D 00       P5007:   DB    0          ;1ST PM5007 BOARD AD
0334   426E 00       N5007:   DB    0          ;# PM5007 BOARDS
0335   426F 00       PPCHN:   DB    0          ;1ST PP PM5008 COMPOSITE CHANNEL #
0336   4270 00       IDCHN:   DB    0          ;1ST ID CH #
0337   4271 00       UDCHN:   DB    0          ;1ST UD CH #
0338   4272 00       DCCHN:   DB    0          ;1ST DC CH #
0339   4273 00       LFCHN:   DB    0          ;1ST LF CH #
0340   4274 00       TRCHN:   DB    0          ;1ST TR CH #
0341   4275 00       ILCHN:   DB    0          ;OUTPUT CHANNEL # FOR INTER-COMPUTER LIGHT
0342                 ;
0343                 ;
0344                 ;SCRATCH VARIABLES FOR ALL MODES. INITIALLY ALL 0.
0345                 ;
0346   4276 00       XYZ9:    DB    0          ;KB
0347   4277 00       XYZ8:    DB    0          ;  DISPLAY
0348   4278 00       XYZ7:    DB    0          ;  DIGITS
0349   4279 00       XYZ6:    DB    0          ;
0350   427A 00       XYZ5:    DB    0          ;
0351   427B 00       XYZ4:    DB    0          ;
0352   427C 00       XYZ3:    DB    0          ;
0353   427D 00       XYZ2:    DB    0          ;
0354   427E 00       XYZ1:    DB    0          ;
0355   427F 00       XYZ0:    DB    0          ;(LS ASCII DIGIT)
0356   4280 00       XYZ:     DB    0          ;MUST BE 0 (SEE TYPEM)
0357   4281 0000     PFSP:    DW    0          ;SP FOR POWER FAIL HOT START
0358   4283 0000     NNOIS:   DW    0          ;# NOISE INT'S PER 8 RTC INT'S
0359   4285 00       PCRON:   DB    0          ;FLAG TO FLASH PCRON LIGHT
0360   4286 0000     FLSHR:   DW    0          ;SPARE FLASHER FLAG
0361   4288 0000     RGAF:    DW    0          ;ERRPC
0362   428A 0000     RGBC:    DW    0          ;  INPUT
0363   428C 0000     RGDE:    DW    0          ;  REGISTERS'
0364   428E 0000     RGHL:    DW    0          ;  VALUES
0365   4290 0000     RGPC:    DW    0          ;
0366   4292 00       INDEX:   DB    0          ;DEVICE INDEX
0367   4293 00       DEVIC:   DB    0          ;DEVICE CODE
0368   4294 00       JINDU:   DB    0          ;INDUCTION STATION #
0369   4295 00       JUPDA:   DB    0          ;CURRENT UPDA PHOTO #
0370   4296 00       JDV:     DB    0          ;CURRENT DIVERT # AND
0371   4297 0000     IDENT:   DW    0          ;  PACKAGE IDENTIFICATION CODE
0372   4299 0000     UDBT:    DW    0          ;UPDA BLOCKING TIME
0373   429B 0000     DVST:    DW    0          ;DIVERT SET TIME
0374   429D 0000     VALUE:   DW    0          ;D4BCD INPUT
0375   429F 0000     CODE:    DW    0          ;PACKAGE CODE LS 2 BYTES; OR PARAMETER CODE
0376   42A1 0000     CODE9:   DW    0          ;  PACKAGE CODE MS BYTE; OR OLD PARAMETER CODE
0377                 ;
0378                 ;MMNPL DATA
0379                 ;
0380   42A3 00       MNPLC:   DB    0          ;DIGIT COUNT ( >= 0 ) WHEN ER = 2 OR 5
0381   42A4 0000     MLOC:    DW    0          ;POINTS TO M BYTE TO BE DISPLAYED
0382   42A6 00       MBYTE:   DB    0          ;M REPLACEMENT BYTE
0383   42A7 00       MTBD:    DB    0          ;M TO BE DISPLAYED
0384                 ;
0385                 ;RUNNING MODE SCRATCH VARIABLES. INITIALLY ALL 0.
0386                 ;
0387   42A8 0000     FSQ1:    DW    0          ;START OF FREE STORAGE QUEUE
0388   42AA 0000     FSQ9:    DW    0          ;  END OF FREE STORAGE QUEUE
0389   42AC 00       FSQ0:    DB    0          ;FREE STORAGE OUT FLAG TO LOCK OUT INDUCTS. &
0390   42AD 00       NTRE0:   DB    0          ;  COUNT DOWN OF # EMPTY TREES
0391   42AE 0000     NNTR0:   DW    0          ;COUNT DOWN OF # TIMES TREES ALL EMPTY
0392   42B0 0000     COUNT:   DW    0          ;TOTAL # PACKAGES THRU UPDA #1
0393   42B2 0000     CTIMP:   DW    0          ;POINTS TO CTIME FOR CURRENT PPI
0394   42B4 0000     TCBP:    DW    0          ;TCB POINTER
0395                 ;
0396                 ;TREE PARAMETERS FOR THE CURRENTLY USED PPI.
0397                 ;
0398   42B6 0000     CTIME:   DW    0          ;PPI CLOCK TIME
```

```
0399   42B8 0000      HEAD:     DW    0          ;TREE HEADER
0400   42BA 0000      RLINK:    DW    0          ;POINTS TO TREE ROOT
0401   42BC 00                  DB    0          ;WOULD-BE TREE BALANCE FACTOR
0402   42BD 0000      NXTIM:    DW    0          ;NEXT EVENT DUE TIME
0403   42BF 00        JPPI:     DB    0          ;CURRENT PPI #
0404   42C0 00        MUD:      DB    0          ;LAST UD # FOR THIS PPI
0405   42C1                     DS    5          ;UNUSED FILLER FOR PPIDB DATA TRANSFER
0406   42C6 0000      STKSV:    DW    0          ;SP SAVE
0407   42C8 0000      QP:       DW    0          ;TREE PROCESSOR
0408   42CA 0000      QX:       DW    0          ;  VARIABLES
0409   42CC 0000      QT:       DW    0          ;
0410   42CE 0000      QW:       DW    0          ;
0411   42D0 0000      QS:       DW    0          ;
0412   42D2 0000      QR:       DW    0          ;
0413   42D4 0000      QDIR:     DW    0          ;
0414   42D6 0000      NEWTM:    DW    0          ;NEW EVENT TIME
0415   42D8 0000      PPIDP:    DW    0          ;POINTER TO CTIME IN PPIDB
0416                  ;
0417                  ;KEYBOARD DATA
0418                  ;
0419   42DA 0000      KBDBP:    DW    0          ;KB DATABASE POINTER
0420   42DC 0000      KBCM:     DW    0          ;AD FOR KB COUT OR SIN
0421   42DE 00        CHAR:     DB    0          ;KB INPUT CHARACTER
0422   42DF 00        NDIG:     DB    0          ;# DIGITS ENTERED
0423                  ;
0424                  ;TRANSLATION DATA
0425                  ;
0426   42E0 00        XL:       DB    0          ;TRANSLATE FLAG FOR PACKAGE ENCODING
0427   42E1 0000      DELTA:    DW    0          ;HASH FUNCTION OFFSET
0428   42E3 01        JXL:      DB    1          ;TRANSLATE TABLE #
0429   42E4 01        MJXL:     DB    1          ;MASTER KB TRANSLATE TABLE #
0430   42E5 0000      NCODE:    DW    0          ;# CODE ENTRIES POSSIBLE PER TABLE
0431   42E7 0000      XLTB:     DW    0          ;TRANSLATE TABLE BASE
0432   42E9 0000      XLTBP:    DW    0          ;TRANSLATE TABLE ENTRY POINTER
0433                  ;
0434                  ;INDUCTION DATA
0435                  ;
0436   42EB 0000      PPDBP:    DW    0          ;POINS TO PPDB
0437   42ED 0000      PSEQ:     DW    0          ;POINTS TO PACKAGE SEQUENCE THRU ID
0438   42EF 0000      RLDBP:    DW    0          ;POINTS TO RLSDB
0439   42F1 0000      NEXTP:    DW    0          ;NEXT TCB POINTER
0440                  ;
0441                  ;UPDATE PHOTO DATA
0442                  ;
0443   42F3 0000      PMUZ:     DW    0          ;SEE UDDB
0444   42F5 0000      PMUZP:    DW    0          ;PMUZ POINTER
0445   42F7 0000      LDVP:     DW    0          ;POINTS TO LDV ITEM
0446                  ;
0447                  ;DIVERT DATA
0448                  ;
0449   42F9 0000      OACT:     DW    0          ;CURRENT OACT IN TCB
0450   42FB 00        LFDV:     DB    0          ;FOR LANE-FULL-CLEAR AUTOMATIC RE-INDUCTION
0451   42FC 0000      LFP:      DW    0          ;POINTS TO LF IN LFDB
0452   42FE 00        C5007:    DB    0          ;=N5007 - BOARD #
0453   42FF 0000      OUTIM:    DW    0          ;TEMPORARY PM5007 OUTPUT IMAGE
0454   4301 0000      DCDBP:    DW    0          ;DCDB ENTRY POINTER
0455                  ;
0456                  ;
0457                  ;DIAGNOSTIC & TRAINIING MODE SCRATCH VARIABLES.
0458                  ;  INITIALLY ALL 0.
0459                  ;
0460   4303 00        CMCHR:    DB    0          ;CM CHAR
0461   4304 00        XC:       DB    0          ;EXERCISER FLAG
0462   4305 0000      SOURC:    DW    0          ;SOURCE PHOTO DEVICE CODE (MS BYTE) & INDEX
0463   4307 0000      DEST:     DW    0          ;DESTINATION PHOTO DEVICE CODE & INDEX
0464   4309 0000      BASE:     DW    0          ;TRAINING TABLE BASE ADDRESS
0465                  ;
0466                  ;DIAGNOSTIC MODE & TRAINING MODE INITIALLY NONZERO VARIABLES.
0467                  ;
0468   430B 01        SUBMO:    DB    1          ;SUBMODE #
0469   430C 4000      XCITV:    DW    64         ;EXERCISE INTERVAL IN PPI'S
0470                                             ;  MUST BE A POWER OF 2
0471   430E 01        KPPI:     DB    1          ;CURRENTLY ACTIVE PPI #
0472   430F 02        DVXYZ:    DB    2          ;CURRENT DIVERT STATE.
0473                                             ;  =1 MEANS SET, =2 MEANS RESET
0474                  ;
0475                  ;INTER-COMPUTER MODE DATA
0476                  ;
0477   4310 00        XMRBS:    DB    0          ;TRANSMITTER BUSY FLAG
0478   4311 00                  DB    0          ;
0479   4312 0000                DW    0          ;
0480   4314 0000                DW    0          ;
0481   4316 0000                DW    0          ;
0482   4318           BYT16:    DS    16         ;MESSAGE TEMPORARY BUFFER
0483   4328           DASDB:    DS    16         ;PSC INPUT MESSAGE BUFFER
0484   4338           MOBUF:    DS    16         ;PSC OUTPUT MESSAGE BUFFER
0485   4348 00        CSEQ:     DB    0          ;COUNTER INDICATING BAD SEQ BYTE
```

```
0486  4349 00      CSUM:   DB   0          ;COUNTER INDICATING BAD CHECKSUM
0487  434A 00      CTOUT:  DB   0          ;COUNTER INDICATING MESSAGE TIMEOUT
0488  434B 0000    DASIP:  DW   0          ;POINTER IN PSC INPUT MESSAGE BUFFER
0489  434D 00      DTYBS:  DB   0          ;FLAG INDICATING PSC IS OUTPUTTING TO I/C 5080
0490  434E 00      FDV:    DB   0          ;FLAG INDICATING A DVT$ OF 0
0491  434F 00      FLG9:   DB   0          ;FLAG USED TO INDICATE MESSAGE 9
0492  4350 00      FLINK:  DB   0          ;FLAG INDICATING I/C LINK STATUS
0493  4351 00      FMBAD:  DB   0          ;FLAG INDICATING BAD MESSAGE RECEIVED BY PSC
0494  4352 00      FULBF:  DB   0          ;FLAG INDICATING FULL PSC INPUT MESSAGE BUFFER
0495  4353 00      ICLGT:  DB   0          ;I/C LIGHT COUNTER = VICLG
0496  4354 0000    LMP:    DW   0          ;LOAD MESSAGE POINTER IN FS
0497  4356 0000    MBUFP:  DW   0          ;POINTER IN PSC OUTPUT MESSAGE BUFFER
0498  4358 0000    SMP:    DW   0          ;SEND MESSAGE POINTER IN FS
0499  435A 00      TCONT:  DB   0          ;COUNTER USED IN TRANSACTION TABLE LOADING
0500  435B 0000    TMOUT:  DW   0          ;MESSAGE TIMEOUT = TDLTM
0501  435D 0000    TRSDP:  DW   0          ;POINTER IN TRANSACTION TABLE
0502                ;
0503                ;GENERAL PURPOSE TEMPORARY VARIABLES
0504                ;
0505  435F 0000    T2:     DW   0          ;TEMPORARY
0506  4361          ;       DS   27         ;(FILLER)
0507                ;
0508                ;PHOTO INTERRUPT BUFFER DATA
0509                ;  N.B.: BUFFER OF LENGTH 64 MUST START AT ADDRESS DIVISIBLE BY 64. ****
0510                ;
0511  437C 8043    PH1:    DW   PHBF       ;PHOTO IN-POINTER
0512  437E 8043    PH2:    DW   PHBF       ;PHOTO OUT-POINTER
0513  4380         PHBF:   DS   64         ;CIRCULAR BUFFER FOR 31 OR LESS INTERRUPTS
0514                ;
0515                ;VARIABLE SIZE TABLES AND FREE STORAGE.
0516                ;  N.B.:  MUST START AT MULTIPLE OF 32.  ******************************
0517                ;
0518  43C0          TBS:    DS   100        ;(DYNAMIC TABLES START HERE)
0519                ;
0520                ;
0521                ;MISCELLANEOUS MACROS
0522                ;
0523                ;
0524                ;JUMP TO NXBC IF B,C NOT= AD OF XBC.
0525                ;REGISTER A,D, E EFFECTED.
0526                ;
0527                JBCN    MACRO XBC,NXBC
0528                        LXI   D,XBC
0529                        MOV   A,C
0530                        CMP   E
0531                        JNZ   NXBC
0532                        MOV   A,B
0533                        CMP   D
0534                        JNZ   NXBC
0535                        ENDM
0536                ;
0537                ;
0538                ;JUMP TO MEQ IF M CONTENTS AT H,L & H,L+2 ARE EQUAL, ELSE FALL THRU.
0539                ;
0540                ;OUTPUT: B,C = 1ST CONTENT, D,E = 2ND CONTENT, H,L POINTS TO MS BYTE.
0541                ;
0542                JMEQ    MACRO MEQ
0543                        MOV   C,M       ;1ST CONTENT IN B,C
0544                        INX   H         ;
0545                        MOV   B,M       ;
0546                        INX   H         ;2ND CONTENT IN D,E
0547                        MOV   E,M       ;
0548                        INX   H         ;
0549                        MOV   D,M       ;
0550                        MOV   A,E
0551                        CMP   C
0552                        JNZ   $+8
0553                        MOV   A,D
0554                        CMP   B
0555                        JZ    MEQ
0556                        ENDM
0557                ;
0558                ;
0559                ;H,L = H,L - XXX. A, E, E EFFECTED. FLAG C SET IF <0.
0560                ;
0561                SUBX    MACRO XXX
0562                        XCHG            ;D,E=SUBTRAHEND
0563                        LHLD  XXX       ;H,L=XXX
0564                        MOV   A,E
0565                        SUB   L
0566                        MOV   L,A
0567                        MOV   A,D
0568                        SBB   H
0569                        MOV   H,A
0570                        ENDM
0571                ;
0572                ;
```

```
0573                    ;BALANCED BINARY TREE MACRO DEFINITION
0574                    ;
0575                    ;
0576                    ;GET VALUE OF LEFT LINK
0577                    ;
0578                    GTLLK   MACRO SRCNO,DEST
0579                            IF    SRCNO     ;IS NODE ADDRESS IN (H,L)?
0580                            LHLD  SRCNO             ;NO-NODE POINTER TO (H,L)
0581                            ENDIF
0582                            MOV   E,M               ;MOVE THE LEFT
0583                            INX   H                 ;LINK INTO THE
0584                            MOV   D,M               ;(D,E) REGISTER
0585                            IF    DEST              ;IS IT SAVE LINK MODE?
0586                            XCHG                    ;YES - LINK TO (H,L)
0587                            SHLD  DEST              ;LINK TO MEMORY
0588                            ENDIF
0589                            ENDM
0590                    ;
0591                    ;GET VALUE OF RIGHT LINK
0592                    ;
0593                    GTRLK   MACRO SRCNO,DEST
0594                            IF    SRCNO             ;IS NODE ADDRESS IN (H,L)?
0595                            LHLD  SRCNO             ;NO - NODE POINTER TO (H,L)
0596                            ENDIF
0597                            INX   H                 ;STEP POINTER TO
0598                            INX   H                 ;THE RIGHT LINK
0599                            MOV   E,M               ;MOVE THE RIGHT
0600                            INX   H                 ;LINK INTO THE
0601                            MOV   D,M               ;(D,E) REGISTER
0602                            IF    DEST              ;IS IT SAVE LINK MODE?
0603                            XCHG                    ;YES - LINK TO (H,L)
0604                            SHLD  DEST              ;LINK TO MEMORY
0605                            ENDIF
0606                            ENDM
0607                    ;
0608                    ;SET VALUE OF LEFT LINK
0609                    ;
0610                    PTLLK   MACRO DSTNO,SOURC
0611                    MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
0612                            IF    DSTNO             ;TEST ASSUMPTION
0613                    MCTEM   SET   0                 ;FALSE ASSUMPTION
0614                            ENDIF
0615                            IF    SOURC             ;IS NEW LINK IN (D,E)?
0616                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
0617                            XCHG                    ;YES - SAVE NODE POINTER
0618                            ENDIF
0619                            LHLD  SOURC             ;NEW LINK TO (H,L)
0620                            XCHG                    ;NEW LINK TO (D,E)
0621                            ENDIF
0622                            IF    DSTNO             ;IS NODE ADDRESS IN (H,L)?
0623                            LHLD  DSTNO             ;NO - NODE POINTER TO (H,L)
0624                            ENDIF
0625                            MOV   M,E               ;MOVE THE NEW
0626                            INX   H                 ;LEFT LINK VALUE
0627                            MOV   M,D               ;INTO THE NODE
0628                            ENDM
0629                    ;
0630                    ;SET VALUE OF RIGHT LINK
0631                    ;
0632                    PTRLK   MACRO DSTNO,SOURC
0633                    MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
0634                            IF    DSTNO             ;TEST ASSUMPTION
0635                    MCTEM   SET   0                 ;FALSE ASSUMPTION
0636                            ENDIF
0637                            IF    SOURC             ;IS NEW LINK IN (D,E)?
0638                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
0639                            XCHG                    ;YES - SAVE NODE POINTER
0640                            ENDIF
0641                            LHLD  SOURC             ;NEW LINK TO (H,L)
0642                            XCHG                    ;NEW LINK TO (D,E)
0643                            ENDIF
0644                            IF    DSTNO             ;IS NODE ADDRESS IN (H,L)?
0645                            LHLD  DSTNO             ;NO - NODE POINTER TO (H,L)
0646                            ENDIF
0647                            INX   H                 ;NOW STEP TO THE
0648                            INX   H                 ;RIGHT LINK BYTES
0649                            MOV   M,E               ;MOVE THE NEW
0650                            INX   H                 ;RIGHT LINK VALUE
0651                            MOV   M,D               ;INTO THE NODE
0652                            ENDM
0653                    ;
0654                    ;GET BALANCE FACTOR OF NODE
0655                    ;
0656                    GTBAL   MACRO NODPT,DEST
0657                            IF    NODPT             ;IS NODE ADDRESS IN (H,L)?
0658                            LHLD  NODPT             ;NO - NODE POINTER TO (H,L)
0659                            ENDIF
```

```
0660                            INX   H                 ;STEP POINTER TO
0661                            INX   H                 ;THE BALANCE BYTE
0662                            INX   H
0663                            INX   H
0664                            MOV   A,M               ;BALANCE CODE TO A-REG
0665                            IF    DEST              ;TEST IF SAVE NODE
0666                            STA   DEST              ;YES - CODE TO MEMORY
0667                            ENDIF
0668                            ORA   A                 ;SET CONDITION CODES
0669                            ENDM
0670                    ;
0671                    ;PUT BALANCE FACTOR INTO A NODE
0672                    ;
0673                    PTBAL   MACRO NODPT,SOURC
0674                            IF    SOURC             ;IS FACTOR IN A-REG?
0675                            LDA   SOURC             ;NO - BALANCE TO A-REG
0676                            ENDIF
0677                            IF    NODPT             ;IS NODE ADDRESS IN (H,L)
0678                            LHLD  NODPT             ;NO - NODE POINTER TO (H,L)
0679                            ENDIF
0680                            INX   H                 ;STEP POINTER TO
0681                            INX   H                 ;THE BALANCE BYTE
0682                            INX   H
0683                            INX   H
0684                            MOV   M,A               ;BALANCE CODE TO NODE
0685                            ENDM
0686                    ;
0687                    ;
0688    4424                    ORG   1400H     ;RUN MODE EXEC
0689                    ;
0690                    ;RUNNING MODE EXECUTIVE.
0691                    ;
0692                    ;INPUT: INTERRUPTS DISABLED, MODE DATA, COLD-START DATA.
0693                    ;ALL REGISTERS & MEMORY EFFECTED.
0694                    ;
0695    1400 CDAC1A     REXEC:  CALL  RINI      ;;INI FOR RUNNING MODE
0696                    ;
0697                    ;CLEAR KB'S
0698                    ;
0699    1403 FB                 EI              ;;ENABLE INTERRUPTS
0700    1404 CDB80C             CALL  CLAKB     ;CLEAR ALL KB'S
0701    1407 C30D14             JMP   RXEC1     ;ENTER MAIN LOOP
0702                    ;
0703                    ;PROCESS EVERYTHING IN PSCBF
0704                    ;
0705    140A CD2F1B     RPHPC:  CALL  PHPC      ;PROCESS ALL PHOTOS
0706    140D CD6510     RXEC1:  CALL  GETPH     ;GET B=DEVIC,
0707    1410 C20A14             JNZ   RPHPC     ;  C=INDEX
0708                    ;
0709                    ;FLASH WATCH-DOG LIGHT
0710                    ;
0711    1413 CD760F             CALL  FLASH     ;FLASH PSC-ON LIGHT
0712                    ;
0713                    ;IF FREE STORAGE IS OUT THEN LOCK OUT INDUCTS
0714                    ;
0715    1416 3AAC42             LDA   FSQ0      ;IF FREE STORAGE
0716    1419 3D                 DCR   A         ;  IS
0717    141A FA3614             JM    RKBPC     ;  OUT
0718    141D CA3C14             JZ    REVNT     ;IF INDUCTS HAVE NOT YET
0719    1420 32AC42             STA   FSQ0      ;  BEEN LOCKED OUT
0720                    ;
0721    1423 3A033D             LDA   NINDU     ;C= INDUCT $,
0722    1426 4F                 MOV   C,A       ;  = NINDU INITIALLY
0723    1427 CD1422     RMCID:  CALL  MCLPC     ;MASTER CLEAR INDUCT
0724    142A 3E0F               MVI   A,15      ;ERROR
0725    142C CD130F             CALL  ERRPC     ;  15
0726    142F 0D                 DCR   C         ;DOUNTIL ALL
0727    1430 C22714             JNZ   RMCID     ;  CLEARED
0728    1433 C33C14             JMP   REVNT     ;IGNORE KB INPUTS
0729                    ;
0730                    ;PROCESS A CHAR IN KBIBF
0731                    ;
0732    1436 CD0E10     RKBPC:  CALL  GETKB     ;A = INPUT CHAR, C = KB $,
0733                                            ;  KBDBP = HL POINTS TO KBDB BYTE 4
0734    1439 C4501B             CNZ   KBPC      ;PROCESS ANY KB INPUT CHAR
0735                    ;
0736                    ;PROCESS EVENTS DUE OR OVERDUE IN ALL AVL TREES
0737                    ;  TREE $ 0 IS FOR RTC, TREES $ 1 TO $ NPPI ARE FOR PPI'S
0738                    ;  (BEWARE OF CRITICAL SECTIONS);
0739                    ;  RTC TREE IS USED FOR LOW PRIORITY DELAYED TASKS
0740                    ;
0741    143C 3A023D     REVNT:  LDA   NPPI      ;A= $ PPI'S = $ TREES - 1
0742    143F 32AD42             STA   NTRE0     ;COUNT DOWN $ EMPTY TREES
0743    1442 2A4542             LHLD  PPIDB     ;POINT TO CTIME IN PPIDB
0744    1445 22D842     NWTRE:  SHLD  PPIDP     ;SAVE PPIDB POINTER
0745    1448 E5                 PUSH  H         ;SAVE AGAIN (GTREP MAY CHANGE PPIDP)
0746    1449 F5                 PUSH  PSW       ;SAVE TREE COUNT
0747    144A 11B642             LXI   D,CTIME   ;D,E POINTS TO CTIME
```

```
0748   144D 0610            MVI   B,10H      ;GET TREE
0749   144F F3              DI                ;;  PARAMETERS
0750   1450 CDCF10          CALL  MOVEB       ;;  (CRITICAL SECTION WITH
0751   1453 FB              EI                ;;  RTCIN & PPIIN)
0752   1454 2ABA42          LHLD  RLINK       ;IF THE TREE
0753   1457 7D              MOV   A,L         ;  IS NOT
0754   1458 B4              ORA   H           ;  EMPTY
0755   1459 CA8E14          JZ    TREE0       ;
0756   145C 2ABD42          LHLD  NXTIM       ;H,L=NXTIM
0757   145F 3AB642          LDA   CTIME       ;IF CTIME-NXTIM >= 0,
0758   1462 95              SUB   L           ;  I.E.,
0759   1463 3AB742          LDA   CTIME+1     ;  IF EVENT
0760   1466 9C              SBB   H           ;  IS OVERDUE
0761   1467 FA9214          JM    NXTRE       ;  OR DUE
0762                ;
0763                ;  ;REMOVE EVENT FROM TREE & EXECUTE IT
0764                ;
0765   146A CDAE27          CALL  DEQUE       ;H,L POINTS TO EVENT TCB
0766   146D 22B442          SHLD  TCBP        ;TCBP POINTS TO TCB
0767   1470 2AD842          LHLD  PPIDP       ;POINT D,E TO
0768   1473 EB              XCHG              ;  TREE HEADER IN PPIDB
0769   1474 13              INX   D           ;POINT TO TREE PARAMETERS
0770   1475 13              INX   D           ;  IN PPIDB
0771   1476 21B842          LXI   H,HEAD      ;POINT H,L TO CTIME
0772   1479 060E            MVI   B,14        ;UPDATE
0773   147B CDCF10          CALL  MOVEB       ;  PPIDB
0774   147E 219214          LXI   H,NXTRE     ;SIMULATE A SUBROUTINE
0775   1481 E5              PUSH  H           ;  RETURN ADDRESS NXTRE
0776   1482 2AB442          LHLD  TCBP        ;POINT H,L TO
0777   1485 110700          LXI   D,7         ;  BYTE 7
0778   1488 19              DAD   D           ;  IN TCB
0779   1489 5E              MOV   E,M         ;POINT
0780   148A 23              INX   H           ;  D,E TO
0781   148B 56              MOV   D,M         ;  EVENT
0782   148C EB              XCHG              ;H,L=ENTRY, D,E POINTS TO
0783                                          ;  BYTE 8 OF TCB
0784   148D E9              PCHL              ;XQ EVENT WITH SIMULATED CALL
0785                ;
0786                ;  ;EXAMINE NEXT EVENT TREE
0787                ;
0788   148E 21AD42  TREE0:  LXI   H,NTRE0     ;COUNT DOWN
0789   1491 35              DCR   M           ;  # EMPTY TREES
0790   1492 F1      NXTRE:  POP   PSW         ;A= #TREES LEFT TO BE EXAMINED
0791   1493 E1              POP   H           ;POINT TO PPIDB
0792   1494 011000          LXI   B,10H       ;POINT TO NEXT CTIME
0793   1497 09              DAD   B           ;  IN PPIDB
0794   1498 3D              DCR   A           ;DO FOR ALL
0795   1499 F24514          JP    NWTRE       ;  TREES
0796                ;
0797                ;IF FSQ0 NOT= 0 & TREE EMPTY FOR SOME TIME THEN COLD START
0798                ;  (A WAIT LOOP COUNT NNTR0 IS USED RATHER THAN THE RTC
0799                ;  SO THAT THE LARGER THE SYSTEM, THE LONGER THE WAIT)
0800                ;
0801   149C 2AAC42          LHLD  FSQ0        ;L=FSQ0, H=NTRE0
0802   149F 2D              DCR   L           ;IF FSQ0 SET, I.E.,
0803   14A0 FA0D14          JM    RXEC1       ;  FREE STORAGE OUT
0804   14A3 24              INR   H           ;IF ALL TREES
0805   14A4 C2B914          JNZ   ENNTR       ;  EMPTY
0806   14A7 2AAE42          LHLD  NNTR0       ;WAIT UNTIL
0807   14AA 2B              DCX   H           ;  ALL TREES
0808   14AB 22AE42          SHLD  NNTR0       ;  HAVE BEEN
0809   14AE 7D              MOV   A,L         ;  EMPTY FOR
0810   14AF B4              ORA   H           ;  SOME
0811   14B0 C20D14          JNZ   RXEC1       ;  TIME
0812                ;
0813   14B3 CD6609          CALL  COLDS       ;COLD START & RESTART
0814   14B6 C30014          JMP   REXEC       ;  RUNNING MODE
0815                ;
0816   14B9 210020  ENNTR:  LXI   H,2000H     ;IF ANY TREE NONEMPTY THEN
0817   14BC 22AE42          SHLD  NNTR0       ;  INITIALIZE WAIT COUNT
0818   14BF C30D14          JMP   RXEC1       ;DO EXEC FOREVER
0819                ; INTERCOMPUTE(I/C) RUNNING MODE EXECUTIVE
0820                ;
0821                ; INPUT: INTERRUPTS DISABLED, MODE DATA, COLD-START DATA
0822                ;        ALL REGISTERS AND MEMORY AFFECTED
0823                ;
0824                ;
0825   14C2 CDAC1A  ICEXC:  CALL  RINI        ;INI FOR I/C RUNNING MODE
0826                ;
0827                ;CLEAR KB'S AND ENABLE INTERRUPTS
0828   14C5 FB              EI                ;ENABLE INTERRUPTS
0829   14C6 CDB80C          CALL  CLAKB       ;CLEAR ALL KB'S
0830   14C9 C3CF14          JMP   ICEX1       ;ENTER MAIN LOOP
0831                ;
0832                ;PROCESS EVERYTHING IN PSCBF
0833                ;
0834   14CC CD2F1B  IPHPC:  CALL  PHPC        ;PROCESS ALL PHOTOS
```

```
0835  14CF CD6510  ICEX1:   CALL  GETPH    ;GET B=DEVIC,
0836  14D2 C2CC14           JNZ   IPHPC    ;   C=INDEX
0837                ;
0838                ;FLASH WATCH-DOG LIGHT
0839                ;
0840  14D5 CD760F           CALL  FLASH    ;FLASH PSC-ON LIGHT
0841                ;
0842                ;IF FREE STORAGE RUNNOUT,LOCKOUT KB INPUTS BUT PROCESS DAS MESSAGES
0843                ;L AND I (ACCEPT BUT DON'T PROCESS D). THIS WILL ALLOW SYSTEM TO CLEAR
0844                ;PACKAGES CURRENTLY IN SYSTEM WITHOUT ACCEPTING NEW PACKAGES WHILE FS
0845                ;IS OUT. AFTER A 'SUFFICIENT' TIME, THE PSCWILL COLDSTART(WARMSTART)
0846                ;ITSELF WHEN ALL THE MESSAGES IN FS HAVE BEEN SENT TO DAS.
0847                ;
0848                ;IF FS IS OUT THEN LOCK OUT INDUCTS
0849                ;
0850  14D8 3AAC42           LDA   FSQ0     ;TEST FOR FS RUNOUT
0851  14DB 3D               DCR   A
0852  14DC FAF814           JM    IKBPC    ;JMP IF NO FS RUNOUT
0853  14DF CAFE14           JZ    FLBCK    ;JMP IF FS RUNOUT IN PROGRESS(IGNORE KB)
0854  14E2 32AC42           STA   FSQ0     ;FS RUNOUT 1ST PASS LOOP
0855  14E5 3A033D           LDA   NINDU    ;C= INDUCT $ = NINDU INITIALLY
0856  14E8 4F               MOV   C,A
0857  14E9 CD1422  IMCID:   CALL  MCLPC    ;MASTER CLEAR INDUCT
0858  14EC 3E0F             MVI   A,15     ;ERROR
0859  14EE CD130F           CALL  ERRPC    ;  15
0860  14F1 0D               DCR   C        ;DO UNTIL ALL
0861  14F2 C2E914           JNZ   IMCID    ;   CLEARED
0862  14F5 C3FE14           JMP   FLBCK    ;IGNORE KB INPUTS
0863                ;
0864                ; PROCESS A CHAR IN KBIBF
0865                ;
0866  14F8 CD0E10  IKBPC:   CALL  GETKB    ;A=INPUT CHAR,C=KB#,
0867                                       ;  KBDBP=HL POINTS TO KBDB BYTE 4
0868  14FB C4501B           CNZ   KBPC     ;PROCESS ANY KB INPUT CHAR(WILL ONLY
0869                                       ;PROCESS A 'CLEAR' FOR DAS SYSTEM)
0870                ;
0871                ;CHECK TO SEE IF A 16 BYTE MESSAGE FROM DAS HAS ARRIVED(IS MESSAGE INPUT
0872                ;BUFFER FULL?)
0873                ;
0874  14FE 3A5243  FLBCK:   LDA   FULBF
0875  1501 D601             SUI   01H
0876  1503 CA4A15           JZ    CDSCK    ;JMP IF MESSAGE RECEIVED FROM DAS
0877                ;CHECK FOR MESSAGE TIMEOUT(TIMEOUT IF 1ST BYTE OF MESSAGE WAS RECEIVED
0878                ;MORE THAN TDLTM AGO)
0879  1506 3A5B43           LDA   TMOUT    ;A=TMOUT
0880  1509 47               MOV   B,A      ;B=TMOUT
0881  150A 3EFF             MVI   A,0FFH   ;A=FF
0882  150C B8               CMP   B        ;COMPARE TMOUT LSB TO FF
0883  150D CA1315           JZ    TMCAL    ;CHECK FURTHER
0884  1510 C31D15           JMP   CALTM    ;1ST BYTE OF MESSAGE HAS BEEN SENT
0885  1513 3E7F     TMCAL:  MVI   A,7FH    ;A=7F
0886  1515 47               MOV   B,A      ;B=7F
0887  1516 3A5C43           LDA   TMOUT+1  ;A=TMOUT+1
0888  1519 B8               CMP   B        ;COMPARE TMOUT MSB TO 7F
0889  151A CA6A15           JZ    IEVNT    ;BYPASS TIMEOUT CALC IF 1ST BYTE OF
0890                                       ;MESSAGE HASN'T BEEN SENT YET(AS TMOUT=7FFF YET)
0891  151D 2A4542  CALTM:   LHLD  PPIDB    ;H,L=LOC OF RTC CTIME LSB
0892  1520 5E               MOV   E,M      ;E=CTIME LSB
0893  1521 23               INX   H        ;H,L=CTIME MSG
0894  1522 56               MOV   D,M      ;D,E=CTIME
0895  1523 2A5B43           LHLD  TMOUT    ;H,L=TMOUT
0896  1526 7B               MOV   A,E      ;A=CTIME LSB
0897  1527 95               SUB   L        ;SUB TMOUT LSB(ALSO SETS BORROW FLAG)
0898  1528 4F               MOV   C,A      ;C=LSB SUBTRACTION
0899  1529 7A               MOV   A,D      ;D=CTIME MSG
0900  152A 9C               SBB   H        ;SUB(TMOUT MSG+BORROW FLAG SETTING)
0901  152B 47               MOV   B,A      ;BIC NOW CONTAINS CTIME-TMOUT
0902  152C 2A2C42           LHLD  TDLTM    ;H,L=TDLTM VALUE
0903  152F 79               MOV   A,C      ;A=LSB OF CTIME-TMOUT
0904  1530 95               SUB   L        ;A=LSB SUBTRACTION (CTIME-TMOUT(-TDLTM
0905  1531 78               MOV   A,B      ;A=MSB
0906  1532 9C               SBB   H        ;A=MSB SUBTRACTION (WITH BORROW FLAG)
0907  1533 F23915           JP    MTMOT    ;JMP IF (CTIME-TMOUT)=> TDLTM
0908  1536 C36A15           JMP   IEVNT    ;JMP IF NO TIMEOUT
0909                ;MESSAGE TIMEOUT. RESET MESSAGE INPUT POINTER DASDB, INCREMENT COUNTER
0910                ;CTOUT FOR DEBUGGING PURPOSES
0911  1539 212843  MTMOT:   LXI   H,DASDB  ;DASIP= DASDB LOC
0912  153C 224B43           SHLD  DASIP
0913  153F 3A4A43           LDA   CTOUT    ;A=CTOUT
0914  1542 C601             ADI   1        ;CTOUT=CTOUT+1
0915  1544 324A43           STA   CTOUT
0916  1547 C36A15           JMP   IEVNT    ;JMP TO IEVNT
0917                ;
0918                ; CHECK MESSAGE FOR GOOD SEQ BYTE AND CHECKSUM
0919                ;
0920  154A 3E00     CDSCK:  MVI   A,0      ;A=0
0921  154C 325243           STA   FULBF            ;RESET FULBF=0
0922  154F CDFF15           CALL  DSCHK    ;CHECK SEQ BYTE AND CHECKSUM
```

```
0923   1552 3A5143          LDA   FMBAD     ;MESSAGE GOOD ?
0924   1555 D600            SUI   00H
0925   1557 CA6715          JZ    CDDEC     ;JMP IF MESSAGE IS GOOD
0926                ;
0927                ; MESSAGE SEQ BYTE OR CHECKSUM NO GOOD. SEND DAS LAST PSC MESSAGE(WITH
0928                ; UNTOGGLED SEQ BYTE).
0929   155A 3E00            MVI   A,00              ;A= 0
0930   155C 325143          STA   FMBAD             ;RESET FMBAD
0931   155F 3EFF            MVI   A,0FFH    ;A=FFH
0932   1561 CD7719          CALL  DSOUT     ;RESEND TO DAS THE LAST PSC MESSAGE
0933   1564 C36A15          JMP   IEVNT     ;JMP TO IEVNT
0934                ;
0935                ; MESSAGE SEQ BYTE AND CHECKSUM IS GOOD. PROCESS THE MESSAGE AND SEND
0936                ; REPLY
0937                ;
0938   1567 CD8116  CDDEC:  CALL  DSDEC
0939                ;
0940                ; PROCESS EVENTS DUE OR OVERDUE IN ALL AVL TREES
0941                ;     TREE # 0 IS FOR RTC, TREES #1 TO NPPI ARE FOR PPI'S
0942                ;     (BEWARE OF CRITICAL SECTIONS);
0943                ;     RTC IS USED FOR LOW PRIORITY DELAYED TASKS
0944                ;
0945   156A 3A023D  IEVNT:  LDA   NPPI      ;A=# PPI'S = #TREES-1
0946   156D 32AD42          STA   NTRE0     ;COUNT DOWN # EMPTY TREES
0947   1570 2A4542          LHLD  PPIDB     ;POINT TO CTIME IN PPIDB
0948   1573 22D842  INWTR:  SHLD  PPIDP     ;SAVE PPIDB POINTER
0949   1576 E5              PUSH  H         ;SAVE AGAIN(GTREP MAY CHANGE PPIDP)
0950   1577 F5              PUSH  PSW       ;SAVE TREE COUNT
0951   1578 11B642          LXI   D,CTIME   ;D,E POINTS TO CTIME
0952   157B 0610            MVI   B,10H     ;GET TREE
0953   157D F3              DI              ;   PARAMETERS
0954   157E CDCF10          CALL  MOVEB     ;   (CRITICAL SECTION WITH
0955   1581 FB              EI              ;   RTCIN AND PPIIN)
0956   1582 2ABA42          LHLD  RLINK     ;IF THE TREE
0957   1585 7D              MOV   A,L       ;   IS NOT
0958   1586 B4              ORA   H         ;   EMPTY
0959   1587 CABC15          JZ    ITREE
0960   158A 2ABD42          LHLD  NXTIM     ;H,L=NXTIM
0961   158D 3AB642          LDA   CTIME     ;IF CTIME-NXTIME>=0,
0962   1590 95              SUB   L         ;   I.E.,
0963   1591 3AB742          LDA   CTIME+1   ;   IF EVENT
0964   1594 9C              SBB   H         ;IS OVERDUE
0965   1595 FAC015          JM    INXTR     ;OR DUE
0966                ;
0967                ; REMOVE EVENT FROM TREE AND EXECUTE IT
0968                ;
0969   1598 CDAE27          CALL  DEQUE     ;H;L POINTS TO EVENT TCR
0970   159B 22B442          SHLD  TCBP      ;TCBP POINTS TO TCB
0971   159E 2AD842          LHLD  PPIDP     ;POINT DIE TO
0972   15A1 EB              XCHG            ;   TREE HEADER IN PPIDB
0973   15A2 13              INX   D         ;POINT TO TREE PARAMETERS
0974   15A3 13              INX   D         ;   IN PPIDB
0975   15A4 21B842          LXI   H,HEAD    ;POINT H,L TO CTIME
0976   15A7 060E            MVI   B,14      ;UPDATE
0977   15A9 CDCF10          CALL  MOVEB     ;   PPIDB
0978   15AC 21C015          LXI   H,INXTR   ;SIMULATE A SUBROUTINE
0979   15AF E5              PUSH  H         ;   RETURN ADDRESS INXTR
0980   15B0 2AB442          LHLD  TCBP      ;POINT H,L TO
0981   15B3 110700          LXI   D,7       ;   BYTE 7
0982   15B6 19              DAD   D         ;   IN TCB
0983   15B7 5E              MOV   E,M       ;POINT
0984   15B8 23              INX   H         ;   DIE TO
0985   15B9 56              MOV   D,M       ;EVENT
0986   15BA EB              XCHG            ;H,L=ENTRY,D,E POINTS TO
0987                                        ;   BYTE 8 OF TCB
0988   15BB E9              PCHL            ;XQ EVENT WITH SIMULATED CALL
0989                ;
0990                ; EXAMINE NEXT EVENT TREE
0991                ;
0992   15BC 21AD42  ITREE:  LXI   H,NTRE0   ;COUNT DOWN
0993   15BF 35              DCR   M         ;   # EMPTY TREES
0994   15C0 F1      INXTR:  POP   PSW       ;A= #TREES LEFT TO BE EXAMINED
0995   15C1 E1              POP   H         ;POIN TO PPIDB
0996   15C2 011000          LXI   B,10H     ;POINT TO NEXT CTIME
0997   15C5 09              DAD   B         ;   IN PPIDB
0998   15C6 3D              DCR   A         ;DO FOR ALL
0999   15C7 F27315          JP    INWTR     ;   TREES
1000                ;
1001                ; TEST TO SEE IF FS IS OUT. IF IT IS, COLDSTART(WARMSTART) AFTER A DELAY
1002                ; AND MESSAGES IN FS ARE SENT TO DAS(A WAIT LOOP COUNT NNTR0 IS USED
1003                ; RATHER THAN THE RTC SO THAT THE LARGER THE SYSTEM, THE LONGER THE WAIT
1004                ;
1005   15CA 2AAC42          LHLD  FSQ0      ;L=FSQ0,H=NTRE0
1006   15CD 2D              DCR   L         ;IF FSQ0 SET,I.E.
1007   15CE FACF14          JM    ICEX1     ;   FREE STORAGE OUT
1008   15D1 24              INR   H         ;IF ALL TREES
1009   15D2 C2F615          JNZ   IENNT     ;   EMPTY
```

```
1010   15D5 2AAE42            LHLD  NNTR0     ;WAIT UNTIL
1011   15D8 2B                DCX   H         ;   ALL TREES
1012   15D9 22AE42            SHLD  NNTR0     ;   HAVE BEEN
1013   15DC 7D                MOV   A,L       ;   EMPTY FOR
1014   15DD B4                ORA   H         ;   SOME
1015   15DE C2CF14            JNZ   ICEX1     ;   TIME
1016                  ; SEE IF MESSAGES IN FS ARE ALL SENT TO DAS
1017   15E1 2AAE42            LHLD  NNTR0      ;SET NNTR0= 1 AGAIN
1018   15E4 23                INX   H
1019   15E5 22AE42            SHLD  NNTR0
1020   15E8 2A5443            LHLD  LMP       ;H,L=LMP VALUE
1021   15EB 7D                MOV   A,L       ;A=L
1022   15EC B4                ORA   H
1023   15ED C2CF14            JNZ   ICEX1     ;JMP IF LMP2 NOT EQUAL ZERO,INDICATING
1024                                           ;MESSAGES STILL IN FS
1025   15F0 CD6609            CALL  COLDS     ;COLDSTART(WARMSTART) AND RESTART
1026   15F3 C3C214            JMP   ICEXC     ;I/C RUN MODE
1027   15F6 210020   IENNT:   LXI   H,2000H   ;IF ANY TREE NONEMPTY THEN
1028   15F9 22AE42            SHLD  NNTR0     ;   INITIALIZE WAIT COUNT
1029   15FC C3CF14            JMP   ICEX1     ;DO EXEC FOREVER
1030                  ;I/C SUBROUTINE DSCHK
1031                  ;
1032                  ;CHECKS DAS MESSAGES FOR CORRECT SEQ BYTE AND CHECKSUM. IF SEQ BYTE
1033                  ;OR CHECKSUM IS INVALID, DON'T TOGGLE SEQ BYTE AND RETURN TO ICEXC. IF
1034                  ;VALID, TOGGLE SEQ BYTE AND RETURN TO ICEXC.
1035                  ;
1036                  ;INPUTS: DAS MESSAGE RESIDES IN BUFFER DASDB(16 BYTES)
1037                  ;
1038                  ;PROCESS:SEE ABOVE
1039                  ;
1040                  ;OUTPUTS:IF MESSAGE IS GOOD, TOGGLE RS,LMR AND SET FMBAD=0 AND RETURN
1041                  ;        TO ICEXC. IF MESSAGE IS BAD, DON'T TOGGLE LMR,RS, SET FMBAD=1
1042                  ;        AND RETURN TO ICEXC. ALSO SET COUNTERS FOR DEBUGGING IF SEQ
1043                  ;        BYTE NOT A '5' OR 'J' OR IF CHECKSUM NO GOOD.
1044                  ;TEST TO SEE IF IT'S MESSAGE L
1045                  ;
1046   15FF 212843    DSCHK:   LXI   H,DASDB   ;H,L= MESSAGE BYTE 0 LOC
1047   1602 23                 INX   H         ;H,L= MESSAGE BYTE 1 LOC
1048   1603 46                 MOV   B,M       ;B= MESSAGE IDENTIFIER
1049   1604 3E4C               MVI   A,'L'     ;A= ASCII L
1050   1606 B8                 CMP   B         ;COMPARE
1051   1607 C21316             JNZ   NOL       ;JMP IF NOT MESSAGE L
1052                  ;ITS MESSAGE L. TEST IF 1ST MESSAGE L
1053   160A 3A5043    YESL:    LDA   FLINK     ;A= FLAG INDICATING L COUNT
1054   160D D600               SUI   0         ;
1055   160F CA1C16             JZ    CDSCH     ;JMP IF THIS IS THE 1ST L MESSAGE
1056                  ;ITS 2ND OR MORE MESSAGE L. THEREFORE IGNORE IT AND DO NOTHING SO
1057                  ;THAT DAS TIMES OUT.  THE PSC LINK CAN ONLY BE OPENED AGAIN WHEN THE
1058                  ;HOST SWITCH IS OPERATED FOLLOWED BY A MESSAGE L FROM DAS
1059   1612 C9                 RET             ;RET TO ICEXC AND TAKE MULTIPLE L PATH IN DSDEC
1060                  ;
1061                  ;ITS NOT MESSAGE L. TEST TO SEE IF L HAS BEEN SENT YET
1062                  ;
1063   1613 3A5043    NOL:     LDA   FLINK     ;A=L COUNT
1064   1616 D600               SUI   0         ;
1065   1618 C21C16             JNZ   CDSCH     ;JMP IF L HAS BEEN SENT AT LEAST ONCE ALREADY
1066                  ;MESSAGE L HAS NOT YET BEEN SENT. THEREFORE DO NOTHING
1067   161B C9                 RET             ;RET TO ICEXC AND TAKE 'DO NOTHING' PATH IN
1068                  ;                        ;DSDEC
1069                  ;TEST IF SEQ BYTE= COMPLEMENT OF LMR
1070   161C 3A2A42    CDSCH:   LDA   LMR       ;A= LMR
1071   161F 2F                 CMA             ;A=COMPLEMENT OF LMR
1072   1620 E67F               ANI   7FH       ;MASK OFF BIT 7 FOR SAFETY
1073   1622 47                 MOV   B,A       ;B= COMPL OF LMR
1074   1623 212843             LXI   H,DASDB   ;H,L=DASDB LOC
1075   1626 7E                 MOV   A,M       ;A= SEQ BYTE
1076   1627 90                 SUB   B         ;A=0 IF SEQ BYTE=COMPLEMENT OF LMR
1077   1628 CA5816             JZ    YSEQ      ;JMP IF SEQ GOOD
1078                  ;SEQUENCE NO GOOD.CHECK TO SEE IF ITS A '5' OR 'J'
1079   162B 212843             LXI   H,DASDB   ;H,L=POINT TO SEQ BYTE
1080   162E 46                 MOV   B,M       ;B=SEQ BYTE
1081   162F 3E35               MVI   A,'5'     ;A= ASCII 5
1082   1631 90                 SUB   B
1083   1632 CA3F16             JZ    NMESG     ;JMP IF SEQ IS A '5'
1084   1635 3E4A               MVI   A,'J'     ;A= ASCII J
1085   1637 90                 SUB   B
1086   1638 CA3F16             JZ    NMESG     ;JMP IF SEQ IS A 'J'
1087                  ; BUMP COUNTER TO INDICATE SEQ NEITHER A '5' NOR A 'J'
1088   163B 214843             LXI   H,CSEQ    ;CSEQ=CSEQ+1
1089   163E 34                 INR   M
1090                  ;
1091                  ;MESSAGE SEQ OR CHECKSUM NO GOOD. LEAVE LMR,RS UNCHANGED, SET FMBAD=1
1092                  ;AND RETURN TO ICEXC
1093   163F 3E01      NMESG:   MVI   A,01H     ;SET FMBAD=1
1094   1641 325143             STA   FMBAD
1095   1644 212843             LXI   H,DASDB   ;H,L=MESSAGE BYTE 0 LOC
1096   1647 23                 INX   H         ;H,L=BYTE 1 LOC
1097   1648 46                 MOV   B,M       ;B=MESSAGE IDENTIFIER
```

```
1098  1649 3E4C          MVI   A,'L'      ;A=ASCII L
1099  164B B8            CMP   B          ;COMPARE
1100  164C C0            RNZ              ;RET IF NO MESSAGE L (TO ICEXC)WITH FMBAD=1
1101  164D 3E00          MVI   A,0        ;SET FMBAD=0 FOR USE ON RETURN TO ICEXC
1102  164F 325143        STA   FMBAD      ;SO THAT DSDEC WILL BE CALLED
1103  1652 3EFF          MVI   A,-1       ;SET FLINK=-1 SO THAT WHEN DSDEC IS CALLED
1104  1654 325043        STA   FLINK      ;ON RETURN TO ICEXC, THE MULTIPLE L PATH
1105  1657 C9            RET              ;WILL BE TAKEN SO DAS TIMES OUT AND FUTURE
1106                                              ;L WILL BE TREATED AS 1ST L.
1107                ;
1108                ;SEQ OK. CHECK FOR CORRECT CHECKSUM
1109  1658 212843  YSEQ:    LXI   H,DASDB   ;H,L=DASDB LOC
1110  165B CD211A          CALL  CHKSM     ;GET PRINTABLE ASCII CHECKSUM IN REG A
1111  165E 212843          LXI   H,DASDB   ;H,L=DASDB BYTE 0 LOC
1112  1661 110D00          LXI   D,13
1113  1664 19              DAD   D         ;H,L=CHECKSUM BYTE 13 LOC
1114  1665 46              MOV   B,M       ;B=CHECKSUM FROM DAS
1115  1666 B8              CMP   B         ;COMPARE CALL WITH DAS CHECKSUM
1116  1667 CA7116          JZ    YMESG     ;JMP IF CHECKSUM GOOD
1117                ;
1118                ;CHECKSUM NO GOOD. DON'T TOGGLE RS,LMR, AND SET FMBAD=1. RET TO ICEXC
1119                ;AFTER BUMPING COUNTER CSUM FOR DEBUGGING PURPOSES
1120  166A 214943          LXI   H,CSUM    ;H,L= CSUM LOC
1121  166D 34              INR   M         ;CSUM = CSUM+1
1122  166E C33F16          JMP   NMESG
1123                ;
1124                ;SEQ BYTE AND CHECKSUM BOTH GOOD. SET FMBAD=0, TOGGLE LMR,RS AND RETURN
1125  1671 212843  YMESG:   LXI   H,DASDB   ;H,L= SEQ BYTE LOC
1126  1674 7E              MOV   A,M       ;A=SEQ BYTE
1127  1675 322A42          STA   LMR       ;LMR= SEQ BYTE JUST RECEIVED FROM DAS
1128  1678 322942          STA   RS        ;RS= SAME
1129  167B 3E00            MVI   A,0
1130  167D 325143          STA   FMBAD     ;FMBAD=0
1131  1680 C9              RET             ;RET TO ICEXC
1132                ;       SUBROUTINE DSDEC
1133                ;
1134                ;  THIS SUBROUTINE DECODES THE VALID(SEQ BYTE AND CHECKSUM ARE OK) DAS
1135                ;  MESSAGES AND PROCESSES THEM IF THEY ARE MASKED FOR PROCESSING.
1136                ;
1137                ;  INPUTS: DAS MESSAGE RESIDING IN BUFFER DASDB(16 BYTES)
1138                ;
1139                ;  PROCESS: NO MESSAGES ARE PROCESSED UNTIL MESSAGE L IS RECEIVED. WHEN
1140                ;           MESSAGE L IS RECEIVED FOR THE 1ST TIME, DSOUT IS CALLED TO
1141                ;           SEND DAS THE 1ST MESSAGE WAITING IN FS TO BE SENT TO DAS IF
1142                ;           THE MESSAGE IN MOBUF IS NUMBER 8 OR 9(LAST MESSAGE SENT TO
1143                ;           DAS IS MESSAGE 8 OR 9). IF THE LAST MESSAGE SENT TO DAS WAS
1144                ;           NOT 8 OR 9, THEN THE MESSAGE IN MOBUF WILL BE RESENT TO DAS
1145                ;           WHEN MESSAGE L IS RECEIVED FOR THE 1ST TIME.MULTIPLE MESSAGE
1146                ;           L'S WILL CAUSE DAS TO TIMEOUT(NO PSC REPLY WILL BE SENT) AND
1147                ;           A FLAG FLINK INCREMENTED FOR DEBUGGING PURPOSES.IF MESSAGE
1148                ;           I IS SENT,DSOUT WILL BE CALLED TO REPLY WITH 1ST WAITING PSC
1149                ;           MESSAGE. IF MESSAGE D IS SENT,IT WILL BE PROCESSED BY
1150                ;           CALLING DASMD AND THEN A REPLY SENT TO DAS BY CALLING DSOUT.
1151                ;           UNINTERPRETABLE MESSAGES WILL CAUSE A MESSAGE 8 REPLY WITH
1152                ;           UNTOGGLED SEQ BYTE SO THAT DAS TIMES OUT.
1153                ;
1154                ;  OUTPUTS: DAS MESSAGE DECODED AND REPLY SENT BACK (BY CALL DSOUT)
1155                ;  CHECK FOR MESSAGE L
1156                ;
1157  1681 212943  DSDEC:   LXI   H,DASDB+1        ;H,L= LOC OF MESSAGE BYTE 1
1158  1684 3E4C            MVI   A,'L'           ;A=ASCII L
1159  1686 96              SUB   M               ;A=ASCII L - MESSAGE BYTE 1
1160  1687 C2D316          JNZ   LPAST           ;IF NOT L, JMP TO LPAST
1161                ;  ITS MESSAGE L
1162                ;  IS IT 1ST MESSAGE L?
1163  168A 3A5043          LDA   FLINK           ;A=FLINK
1164  168D A7              ANA   A               ;SET ZERO FLAG AS PER VALUE OF FLINK
1165  168E CA9616          JZ    LPROS           ;JMP TO LPROS IF FLINK=0
1166                ;  MULTIPLE MESSAGE L.
1167  1691 215043  LMULT:   LXI   H,FLINK         ;H,L= FLINK LOC IN MEMORY
1168  1694 34              INR   M               ;FLINK=FLINK+1
1169  1695 C9              RET                   ;RETURN TO ICEXC WITHOUT REPLY TO DAS SO
1170                ;                            ;DAS WILL TIMEOUT AND REQUIRE I/C
1171                ;                            ;SWITCH TO BE RESET TO RESYNC RS,LMR,ETC
1172                ;                            ;DON'T REFRESH I/C LIGHT EITHER
1173                ;  1ST TIME MESSAGE L IS SENT
1174  1696 3E01    LPROS:   MVI   A,01H          ;A=1
1175  1698 325043           STA   FLINK          ;FLINK=1
1176  169B 3A7542           LDA   ILCHN          ;A=I/C LIGHT CHANNEL
1177  169E 4F               MOV   C,A            ;C=SAME
1178  169F 0601             MVI   B,1            ;B=1 FOR TURN ON
1179  16A1 CD0F11           CALL  OPUT           ;TURN ON I/C LIGHT
1180  16A4 3A2E42           LDA   VICLG          ;A= VICLG
1181  16A7 325343           STA   ICLGT          ;SET I/C LIGHT REFRESH TIMER
1182  16AA 213943           LXI   H,MOBUF+1      ;H,L=LOC OF BYTE 1 OF MOBUF
1183  16AD 3E20             MVI   A,20H          ;A=ASCII SPACE
1184  16AF 96               SUB   M              ;A= 0 - BYTE 1 OF MOBUF
1185  16B0 CACD16           JZ    CDSOT          ;JMP IF BYTE 1 = 0 (IF NO MESS IN MOBUF)
```

```
1186  16B3 3E38          MVI   A,'8'          ;A=ASCII 8
1187  16B5 96            SUB   M              ;A=ASCII 8 - BYTE 1 OF MOBUF
1188  16B6 CACD16        JZ    CDSOT          ;JMP IF BYTE 1 = 8 (LAST MESSAGE SENT=8)
1189  16B9 3E39          MVI   A,'9'          ;DO SAME FOR MESSAGE 9
1190  16BB 96            SUB   M
1191  16BC CACD16        JZ    CDSOT          ;JMP IF MESSAGE 9 IN MOBUF
1192                ;
1193                ;   RESEND MOBUF TO DAS WITH NEW SEQ BYTE
1194  16BF 212843        LXI   H,DASDB        ;
1195  16C2 7E            MOV   A,M            ;A= SEQ BYTE  FROM DAS
1196  16C3 213843        LXI   H,MOBUF
1197  16C6 77            MOV   M,A            ;MOBUF BYTE 0= SEQ BYTE FROM DAS
1198  16C7 3EFF          MVI   A,0FFH         ;A=FF
1199  16C9 CD7719        CALL  DSOUT          ;SEND MOBUF TO DAS
1200  16CC C9            RET                  ;RET TO ICEXC
1201                ;  SEND DAS 1ST MESSAGE IN FS (DEFAULTS TO IDLE MESSAGE)
1202  16CD 3E00     CDSOT: MVI   A,00H          ;A=00
1203  16CF CD7719        CALL  DSOUT
1204  16D2 C9            RET                  ;RETURN TO ICEXC
1205                ;
1206                ;  NOT MESSAGE L
1207                ;
1208                ;  HAS MESSAGE L BEEN SENT YET
1209  16D3 3A5043   LPAST: LDA   FLINK          ;A=FLINK
1210  16D6 D601          SUI   01H            ;A=FLINK-1
1211  16D8 CADC16        JZ    ITEST          ;JMP TO ITEST IF FLINK=1(L HAS BEEN SENT
1212  16DB C9            RET                  ;DO NOTHING.RETURN TO ICEXC AND WAIT FOR
1213                ;                         ;MESSAGE L OTHERWISE
1214                ;  REFRESH I/C LIGHT
1215  16DC 3A2E42   ITEST: LDA   VICLG          ;A= VICLG
1216  16DF 325343        STA   ICLGT          ;ICLGT=VICLG
1217                ;  TEST FOR MESSAGE I
1218  16E2 212943        LXI   H,DASDB+1      ;H,L= ADDR OF BYTE 1
1219  16E5 3E49          MVI   A,'I'          ;A= ASCII I
1220  16E7 96            SUB   M              ;A= ASCII I - BYTE 1
1221  16E8 C2FA16        JNZ   DTEST          ;JMP TO DTEST IF NOT MESSAGE I
1222                ;  ITS MESSAGE I IS I MASKED?
1223  16EB 21253D        LXI   H,ERMKI+3      ;
1224  16EE 7E            MOV   A,M            ;A= BYTE 4 OF ERMKI
1225  16EF E601          ANI   01H            ;LOOK AT MESSAGE I BIT
1226  16F1 CAFA16         JZ    DTEST          ;JMP IF NOT MASKED
1227  16F4 3E00     IPROS: MVI   A,00H          ;A=00H
1228  16F6 CD7719        CALL  DSOUT          ;CALL DSOUT TO SEND DAS 1ST WAITING
1229                ;                         ;MESSAGE IN FS.DEFAULT TO I MESSAGE
1230  16F9 C9            RET                  ;RET TO ICEXC
1231                ;
1232                ;  TEST FOR MESSAGE D
1233                ;
1234  16FA 212943   DTEST: LXI   H,DASDB+1      ;H,L= LOC OF BYTE 1 OF MESSAGE
1235  16FD 3E44          MVI   A,'D'
1236  16FF 96            SUB   M              ;A=ASCII D - BYTE 1
1237  1700 C26517        JNZ   DFALT          ;JMP IF NOT D
1238                ; ITS MESSAGE D. IS D MASKED?
1239  1703 21243D        LXI   H,ERMKI+2      ;
1240  1706 7E            MOV   A,M            ;A= BYTE 3 OF ERMKI
1241  1707 E608          ANI   08H            ;LOOK AT MESSAGE D BIT
1242  1709 CA6517        JZ    DFALT          ;JMP IF D NOT MASKED
1243                ;  TEST TO SEE IF FS RUNNOT IS IN PROGRESS
1244  170C 3AAC42        LDA   FSQ0           ;A=FSC0 FLAG
1245  170F A7            ANA   A              ;SET ZERO FLAG ACCORDINGLY
1246  1710 C2F416        JNZ   IPROS          ;JMP IF FS RUNNOUT IS IN PROGRESS
1247                ;TEST TO SEE LAQ OVERFLOW HAS OCCURRED. IF SO, DON'T PROCESS MESSAGE
1248                ;D TILL ERROR 6 IS CLEARED OFF KB.
1249                ;
1250                ;ACCESS ER FLAG IN KBDB TO SEE IF ERROR 6 (LAQ OVERFLOW) IS ON KB
1251  1713 212843        LXI   H,DASDB   ;H,L= MESSAGE BYTE 0
1252  1716 110800        LXI   D,8       ;
1253  1719 19            DAD   D         ;H,L= INDUCT BYTE 1 LOC
1254  171A 7E            MOV   A,M       ;A= INDUCT MSB
1255  171B 0E0A          MVI   C,10      ;C= WEIGHT OF 10
1256  171D CD8F18        CALL  ASCII     ;CONVERT TO BINARY
1257  1720 5F            MOV   E,A       ;E= CURRENT SUM
1258  1721 23            INX   H         ;H,L= INDUCT BYTE 2 LOC
1259  1722 7E            MOV   A,M       ;A= INDUCT LSB
1260  1723 0E01          MVI   C,1       ;C= WEIGHT OF 1
1261  1725 CD8F18        CALL  ASCII     ;A= BINARY 1'S
1262  1728 83            ADD   E         ;A= BINARY INDUCT $ = KB$
1263  1729 CD170D        CALL  CPKBP     ;POINT TO KBI1 IN KBDB FOR KB IN REG A
1264  172C 110400        LXI   D,4       ;
1265  172F 19            DAD   D         ;H,L= ER LOC
1266  1730 7E            MOV   A,M       ;A= ER VALUE
1267  1731 D601          SUI   1         ;SUB 1
1268  1733 C24117        JNZ   THDAS     ;JMP IF NO ERROR ON KB
1269                ;ER= 1. THEREFORE AN ERROR IS ON THIS KB. ACCESS ERROR CODE(TEMPORARILY
1270                ;STORED IN SND LOC) TO SEE IF ITS ERROR $6.
1271  1736 23            INX   H         ;
1272  1737 23            INX   H         ;H,L= SND LOC
1273  1738 7E            MOV   A,M       ;A= SND VALUE = ERROR CODE
```

```
1274  1739 D606            SUI   6          ;SUB 6
1275  173B C24117          JNZ   THDAS      ;JMP IF NOT ERROR 6(LAQ OVERFLOW)
1276                ;LAQ OVERFLOW HAS OCCURRED ON THIS KB. THEREFORE DON'T PROCESS ANYMORE
1277                ;MESSAGE D'S UNTIL KB IS CLEARED(AND LAQ) BY HITTING 'CLEAR' KEY
1278                ;ON KB. THIS IS NECESSARY TO INSURE PACKAGES DON'T GET OUT OF SEQUENCE.
1279  173E C3F416          JMP   IPROS      ;JMP TO IPROS
1280                ;
1281                ;TEST IF IN HALF DAS MODE. IF ARE, DON'T PROSS MESSAGE D.
1282                ;
1283  1741 3A0640   THDAS:  LDA   HDSEN      ;A= HALF-DAS FLAG
1284  1744 D601            SUI   1          ;SUB 1
1285  1746 CAF416          JZ    IPROS      ;JMP IF HALF-DAS
1286                ;
1287                ;PROCESS MESSAGE D
1288  1749 CD9617          CALL  DASMD            ;PROCESS MESSAGE D
1289  174C 3A4F43          LDA   FLG9             ;FLG9=1?
1290  174F D601            SUI   01H
1291  1751 CA5A17          JZ    REGA             ;JMP IF FLG9=1
1292  1754 3E00            MVI   A,0              ;REG A = 0
1293  1756 CD7719          CALL  DSOUT            ;CALL DSOUT WITH REGA=0
1294  1759 C9              RET                    ;RET TO ICEXC
1295  175A 3E00     REGA:  MVI   A,0              ;A=0
1296  175C 324F43          STA   FLG9             ;FLG9=0
1297  175F 3EFF            MVI   A,0FFH           ;A= FFH
1298  1761 CD7719          CALL  DSOUT            ;CALL DSOUT WITH REG A=FF
1299  1764 C9              RET                    ;RET TO ICEXC
1300                ;  DEFAULT TO MESSAGE 8. DON'T TOGGLE RS,LMR AS WANT LINK TO GO DOWN.
1301                ;  SINCE WE HAVE PREVIOUSLY TOGGLED THEM AT YMESG,WE MUST RETOGGLE
1302                ;  THEM HERE TO GET BACK TO ORIGINAL DAS STATE.
1303  1765 3A2843   DFALT:  LDA   DASDB            ;A= SEQ BYTE FROM DAS
1304  1768 2F              CMA                    ;A= COMPLEMENT OF SEQ BYTE
1305  1769 E67F            ANI   7FH              ;MASK BIT 7
1306  176B 322A42          STA   LMR              ;SET LMR,RS & SEQ BYTE 0 IN MOBUF = TO
1307  176E 322942          STA   RS               ;COMPLEMENT OF SEQ BYTE SENT BY DAS AS
1308  1771 323843          STA   MOBUF            ;DON'T WANT TO TOGGLE SEQ BYTE
1309  1774 212843          LXI   H,DASDB
1310  1777 224B43          SHLD  DASIP
1311                ;
1312                ; LOAD REST OF MESSAGE 8 INTO MOBUF
1313  177A 212943          LXI   H,DASDB+1        ;H,L= BYTE 1 LOC OF INPUT MESSAGE
1314  177D 113943          LXI   D,MOBUF+1        ;D,E= BYTE 1 LOC OF OUTPUT MESSAGE
1315  1780 010F00          LXI   B,15             ;B,C= 15 BYTES TO MOVE
1316  1783 CDC410          CALL  MOVE             ;MOVE 15 BYTES FROM DASDB TO MOBUF
1317  1786 3E38            MVI   A,'8'            ;A=ASCII 8
1318  1788 323943          STA   MOBUF+1          ;STORE IN MOBUF
1319  178B 3E08            MVI   A,8              ;SEND ERROR 8 TO MASTER
1320  178D CD080F          CALL  ERPC             ;
1321  1790 3EFF            MVI   A,0FFH           ;A=FFFF
1322  1792 CD7719          CALL  DSOUT            ;SEND MOBUF TO DAS
1323  1795 C9              RET                    ;RET TO ICEXC
1324                ; SUBROUTINE DASMD
1325                ;
1326                ; INPUTS: MESSAGE D RESIDING IN DASDB
1327                ;
1328                ; PROCESS: PUT PACKAGE IDENT AND DVT INTO LAQ. OTHER FUNCTIONS SIMILAR
1329                ;          TO SNDPC
1330                ;
1331                ; OUTPUTS: SIMILAR TO SNDPC SUBROUTINE
1332                ;
1333                ; CONVERT ASCII DVT # AND INDUCT # IN MESSAGE D TO BINARY AND PUT ON
1334                ; STACK FOR FUTURE USE
1335                ;
1336  1796 212843   DASMD:  LXI   H,DASDB          ;H,L= LOC MESSAGE D BYTE 0
1337  1799 110A00          LXI   D,10
1338  179C 19              DAD   D                ;H,L= LOC OF DVT BYTE 1
1339  179D 7E              MOV   A,M              ;A= DVT BYTE 1 (100'S DIGIT)
1340  179E 0E64            MVI   C,100            ;C= WEIGHT OF 100
1341  17A0 CD8F18          CALL  ASCII            ;CONVERT TO BINARY
1342  17A3 5F              MOV   E,A              ;E= CURRENT SUM
1343  17A4 23              INX   H                ;H,L= DVT BYTE 2 (10'S DIGIT) LOC
1344  17A5 7E              MOV   A,M              ;A= 10'S DIGIT
1345  17A6 0E0A            MVI   C,10             ;C= WEIGHT OF 10
1346  17A8 CD8F18          CALL  ASCII            ;A= BINARY EQUIVALENT
1347  17AB 83              ADD   E                ;ADD 100'S
1348  17AC 5F              MOV   E,A              ;E= 100'S + 10'S
1349  17AD 23              INX   H                ;H,L= DVT BYTE 3 (1'S DIGIT) LOC
1350  17AE 7E              MOV   A,M              ;A= 1'S DIGIT
1351  17AF 0E01            MVI   C,1              ;C= WEIGHT OF 1
1352  17B1 CD8F18          CALL  ASCII            ;A= BINARY 1'S
1353  17B4 83              ADD   E                ;A= BINARY DIVERT #
1354  17B5 47              MOV   B,A              ;B= SAME
1355  17B6 C5              PUSH  B                ;STACK = SAME
1356  17B7 212843          LXI   H,DASDB          ;H,L= LOC OF MESSAGE D
1357  17BA 110800          LXI   D,8
1358  17BD 19              DAD   D                ;H,L= INDUCT BYTE 1 LOC
1359  17BE 7E              MOV   A,M              ;A= INDUCT BYTE 1 LOC (10'S DIGIT)
1360  17BF 0E0A            MVI   C,10             ;C= WEIGHT OF 10
1361  17C1 CD8F18          CALL  ASCII            ;CONVERT TO BINARY
```

```
1362  17C4 5F          MOV   E,A             ;E=CURRENT SUM
1363  17C5 23          INX   H               ;H,L= INDUCT BYTE 2 LOC (1'S DIGIT)
1364  17C6 7E          MOV   A,M             ;A= 1'S DIGIT
1365  17C7 0E01        MVI   C,1             ;C= WEIGHT OF 1
1366  17C9 CD8F18      CALL  ASCII           ;A= BINARY 1'S
1367  17CC 83          ADD   E               ;A= BINARY INDUCT #
1368  17CD C1          POP   B               ;B= JDV IN BINARY
1369  17CE 4F          MOV   C,A             ;C= JINDU IN BINARY
1370  17CF C5          PUSH  B               ;STACK= JDV,JINDU IN BINARY
1371                ;
1372                ; CHECK FOR LEGAL INDUCT # (>0 AND < NINDU)
1373  17D0 79          MOV   A,C             ;A= INDUCT #
1374  17D1 D600        SUI   0
1375  17D3 CA1E18      JZ    SNDM9           ;JMP IF IND = 0
1376  17D6 3A033D      LDA   NINDU           ;A=NINDU
1377  17D9 91          SUB   C               ;A=NINDU-JINDU
1378  17DA FA1E18      JM    SNDM9           ;JMP IF JINDU> NINDU (# OF INDUCTS)
1379                ;
1380                ; INDUCT IS LEGAL. CHECK FOR LEGAL PACK CODE (EACH BYTE A NUMBER)
1381  17DD 0630        MVI   B,30H           ;B= ASCII 0
1382  17DF 0E39        MVI   C,39H           ;C= ASCII 9
1383  17E1 212A43      LXI   H,DASDB+2       ;H,L= PACKAGE CODE BYTE 1
1384  17E4 1606        MVI   D,6             ;D=LOOP COUNTER
1385  17E6 79    LEGPC: MOV   A,C             ;A= UPPER LIMIT ASCII 9
1386  17E7 96          SUB   M               ;A= ASCII 9 - BYTE
1387  17E8 FA1E18      JM    SNDM9           ;JMP IF BYTE > 9
1388  17EB 7E          MOV   A,M             ;A= PACK CODE BYTE
1389  17EC 90          SUB   B               ;A-BYTE - ASCII 0
1390  17ED FA1E18      JM    SNDM9           ;JMP IF BYTE < 0
1391  17F0 23          INX   H               ;GO TO NEXT BYTE
1392  17F1 15          DCR   D               ;DEC COUNTER
1393  17F2 C2E617      JNZ   LEGPC           ;DO FOR ALL 6 BYTES
1394                ; LEGAL PACKAGE CODE. CHECK FOR LEGAL DV # (>0 AND <NDVT)
1395  17F5 C1          POP   B               ;B,C= JDV,JINDU
1396  17F6 C5          PUSH  B               ;STACK=SAME
1397  17F7 78          MOV   A,B             ;A= JDV
1398  17F8 D600        SUI   00H
1399  17FA CA0A18      JZ    DVEO            ;JMP IF DV#=0
1400  17FD 3A053D      LDA   NDVT            ;A= NDVT
1401  1800 57          MOV   D,A             ;0= SAME
1402  1801 78          MOV   A,B             ;A= DV#
1403  1802 3D          DCR   A               ;BIAS FOR 'JP' INSTRUCTION
1404  1803 92          SUB   D               ;A= DV#-NDVT
1405  1804 F20A18      JP    DVEO            ;JMP IF DV#>NDVT
1406  1807 C34A18      JMP   INHIB           ;LEGAL DV#.JMP TO INHIB
1407                ; DV # IS ILLEGAL. CHECK IF ZERO OR NON-ZERO
1408                ;
1409  180A 78    DVEO: MOV   A,B             ;A= BINARY DV #
1410  180B D600        SUI   00H
1411  180D C21E18      JNZ   SNDM9           ;JMP IF DV # > 0
1412                ; DV# IS 0. DEFAULT JDV TO SERDV. SET FDV FOR USE IN DPLAQ
1413  1810 3A3A3D      LDA   SERDV
1414  1813 329642      STA   JDV             ;JDV=SERDV
1415  1816 3E01        MVI   A,01H
1416  1818 324E43      STA   FDV             ;FDV=1
1417  181B C34A18      JMP   INHIB
1418                ; SEND MESSAGE 9 TO DAS WITH SEQUENCE BYTE UNTOGGLED SO DAS TIMESOUT
1419  181E 3A2843 SNDM9: LDA   DASDB          ;A= SEQ BYTE RECEIVED FROM DAS
1420  1821 2F          CMA                   ;A= COMPLEMENT OF SEQ BYTE FROM DAS
1421  1822 E67F        ANI   7FH             ;MASK BIT 7
1422  1824 322A42      STA   LMR             ;SET LMF,RS AND SEQ BYTE IN MOBUF
1423  1827 322942      STA   RS              ;EQUAL TO ORIGINAL STATE (UNTOGGLED)
1424  182A 323843      STA   MOBUF
1425                ; ECHO INPUT MESSAGE BY LOADING INTO MOBUF AND SENDING TO DAS BY
1426                ; CALL DSOUT WITH REG A = FFH
1427  182D 212943      LXI   H,DASDB+1       ;H,L= BYTE 1 OF DASDB
1428  1830 113943      LXI   D,MOBUF+1       ;D,E= BYTE 1 OF MOBUF
1429  1833 010F00      LXI   B,15            ;B= 15 BYTES TO MOVE
1430  1836 CDC410      CALL  MOVE            ;MOVE 15 BYTES FROM DASDB TO MOBUF
1431  1839 3E39        MVI   A,'9'           ;A= ASCII 9
1432  183B 323943      STA   MOBUF+1         ;MOBUF BYTE 1 = MESSAGE 9
1433  183E 3E09        MVI   A,09H           ;A=09H
1434  1840 CD080F      CALL  ERPC            ;PUT ERROR 9 ON KB
1435  1843 3E01        MVI   A,01H           ;FLG9=1 SO WHEN DSOUT IS CALLED ON RET
1436  1845 324F43      STA   FLG9            ;TODSDEC MOBUF WILL BE SENT TO DAS
1437  1848 C1          POP   B               ;ADJUST STACK
1438  1849 C9          RET                   ;RET TO DSDEC
1439                ;
1440                ; CHECK FOR LANE FULL KB INHIBIT DESIRED
1441  184A C1    INHIB: POP   B               ;B,C=JDV, JINDU
1442  184B 3A3C3D      LDA   LFO             ;CHECK TO SEE IF LANE-FULL KB INHIBIT
1443  184E E601        ANI   01H             ;    IS DESIRED
1444  1850 CA7018      JZ    CDPLQ           ;JMP IF NOT DESIRED
1445                ;
1446                ; GET LANE FULL MASK
1447  1853 78          MOV   A,B             ;A=JDV
1448  1854 C5          PUSH  B               ;STACK= JDV, JINDU
```

```
1449  1855 CD250D            CALL  CPLFM             ;CP LF MASK IN A. H,L=MASK POINTER
1450  1858 C1                POP   B                 ;B,C= JDV. JINDU
1451  1859 A6                ANA   M                 ;IF LANE FULL JMP TO CALL 7
1452  185A C26018            JNZ   CALL7             ;IF NOT FULL JMP TO CDPLQ
1453  185D C37018            JMP   CDPLQ
1454  1860 3E07      CALL7:  MVI   A,7               ;LANE FULL. PUT ERROR 7 ON KB. IF AUTO
1455  1862 CD130F            CALL  ERRPC             ;REINDUCT AFTER LANE CLEARS IS DESIRED,
1456  1865 3A3C3D            LDA   LFO               ;SET AUTO REINDUCT FLAG ER=10
1457  1868 E610              ANI   10H
1458  186A C8                RZ                      ;RET TO DSDEC IF NO AUTO REINDUCT
1459  186B 2ADA42            LHLD  KBDBP
1460  186E 77                MOV   M,A               ;SET ER=10
1461  186F C9                RET                     ;RET TO DSDEC
1462                 ;
1463                 ; LOAD LAQ WITH JDV AND IDENT(IDENT= LOC IN TRANSACTION TABLE OF
1464                 ; PACKAGE CODE) AND ATTEMPT TO SEND PACKAGE
1465  1870 C5        CDPLQ:  PUSH  B                 ;STACK= JDV. JINDU
1466  1871 CD9D18            CALL  DPLAQ             ;LOAD LAW WITH JDV,IDENT
1467  1874 C1                POP   B                 ;B,C= JDV,JINDU
1468  1875 CA0B22            JZ    SLAQF             ;IF LAQ FULL,SEND ERROR 6 TO KB
1469  1878 2AEB42            LHLD  PPDBP             ;POINT H,L TO BYTE 1 IN PPDB
1470  187B 7E                MOV   A,M               ;A=PWING
1471  187C 3D                DCR   A                 ;IF PACKAGE AT PP
1472  187D C28618            JNZ   CRBSO             ;   WAITING TO BE CODED
1473                 ;
1474                 ; SEND OUT WAITING PACKAGE IF NO MERGE CONFLICT
1475  1880 C5                PUSH  B                 ;STACK= JDV,JINDU
1476  1881 23                INX   H                 ;H,L= BYTE 2 OF PPDB
1477  1882 CD7022            CALL  PSEQR             ;QUEUE PACKAGE UP FOR RELEASE
1478  1885 C1                POP   B                 ;B,C= JDV,JINDU .
1479                 ;
1480                 ; CONSIDER REPEAT-BY-SEND OPTION
1481  1886 3A323D    CRBSO:  LDA   SENDO             ;IF REPEAT NOT DESIRED
1482  1889 E620              ANI   20H               ;   THEN CLEAR
1483  188B CCC90C            CZ    CLRKB             ;   KB AND KBDB
1484  188E C9                RET                     ;RET TO DSDEC
1485                 ; SUBROUTINE ASCII
1486                 ;
1487                 ; INPUTS: REG A = ASCII CHARACTER, REG C = VALUE ASSIGNED TO EACH
1488                 ;         DIGIT(1,10 OR 100)
1489                 ;
1490                 ; PROCESS: CONVERTS ASCII CHAR IN REG A TO BINARY VALUE WEIGHTED
1491                 ;          BY REG C
1492                 ;
1493                 ; OUTPUTS: WEIGHTED BINARY VALUE IN REG A
1494                 ;
1495  188F E60F      ASCII:  ANI   0FH               ;LOOK AT 4 LSB
1496  1891 CA9C18            JZ    RETRN             ;JMP IF ZERO
1497  1894 47                MOV   B,A               ;B= LOOP COUNT
1498  1895 3E00              MVI   A,0               ;INITIALIZE A
1499  1897 81        ALOOP:  ADD   C                 ;A= WEIGHT
1500  1898 05                DCR   B                 ;DEC COUNTER
1501  1899 C29718            JNZ   ALOOP             ;DO FOR ALL COUNTS
1502  189C C9        RETRN:  RET                     ;RET WITH WEIGHTED BINARY $ IN REG A
1503                 ; SUBROUTINE DPLAQ
1504                 ;
1505                 ; INPUTS: MESSAGE D RESIDING IN DASDB. STACK= JDV,JINDU IN BINARY
1506                 ;
1507                 ; PROCESS: LOAD THE LAQ FOR THE INDUCT IN MESSAGE D WITH THE DIVERT
1508                 ;          NUMBER IN MESSAGE D(DVT$ IN BINARY). LOAD THE IDENT IN THE
1509                 ;          LAQ WITH THE ADDRESS IN THE TRANSACTION TABLE WHERE THE
1510                 ;          6 BYTE ASCII PACKAGE CODE HAS BEEN STORED
1511                 ;
1512                 ;OUTPUTS: STACK=JDV,JINDU. LAQ LOADED WITH JDV,IDENT
1513                 ;
1514                 ; TEST TO SEE IF LAQ IS FULL
1515  189D 79        DPLAQ:  MOV   A,C               ;A=INDUCT $
1516  189E CD9927            CALL  CPPPP             ;POINT H,L TO BYTE 1
1517  18A1 23                INX   H                 ;   IN PPDB FOR THIS INDUCT
1518  18A2 22EB42            SHLD  PPDBP             ;OUTPUT POINTER
1519  18A5 110500            LXI   D,5
1520  18A8 19                DAD   D
1521  18A9 3A343D            LDA   DQLIM
1522  18AC 96                SUB   M
1523  18AD C8                RZ                      ;RET IF LAQ FULL
1524                 ;
1525                 ; LAQ NOT FULL. TEST TO SEE IF JDV HAS ALREADY BEEN SET IN DASMD FOR
1526                 ; THE CASE WHEN JDV=0.
1527  18AE 3A4E43            LDA   FDV               ;FDV=1?
1528  18B1 A7                ANA   A
1529  18B2 CABE18            JZ    CNVJD             ;JMP IF FDV=0
1530  18B5 3E00              MVI   A,0               ;SET FDV=0
1531  18B7 324E43            STA   FDV
1532  18BA C5                PUSH  B                 ;STACK= JDV,JINDU
1533  18BB C3C318            JMP   PCODE             ;JDV ALREADY LOADED IN MEMORY,JUMP
1534                 ; LOAD JDV FROM MESSAGE D AS BINARY INTO MEMORY LOC JDV
1535  18BE C5        CNVJD:  PUSH  B                 ;STACK=JDV,JINDU
```

```
1536  18BF 78              MOV   A,B              ;A=JDV
1537  18C0 329642          STA   JDV              ;MEMORY JDV = JDV IN BINARY
1538                       ;
1539                       ; SEE IF PACKAGE CODE=000000 MEANING TO RECIRCULATE PACKAGE. IF SO,
1540                       ; BYPASS PUTTING CODE INTO TRANSACTION TABLE AND SET IDENT=0000 TO
1541                       ; IDENTIFY RECIRCULATING PACKAGE
1542  18C3 212A43  PCODE:  LXI   H,DASDB+2        ;H,L= PACKAGE CODE BYTE 1
1543  18C6 0606            MVI   B,6              ;B= COUNTER
1544  18C8 7E      PCDCK:  MOV   A,M              ;A= PACK CODE BYTE
1545  18C9 D630            SUI   '0'              ;SUBTRACT ASCII 0
1546  18CB C2DE18          JNZ   SCODE            ;JMP IF EACH BYTE NOT A ASCII 0
1547  18CE 23              INX   H                ;LOOK AT NEXT BYTE
1548  18CF 05              DCR   B                ;DEC COUNTER
1549  18D0 C2C818          JNZ   PCDCK            ;DO ALL 6 BYTES
1550                       ;
1551                       ; PACKAGE CODE ='S 000000. THEREFORE SET IDENT=0000 SO WHEN PACKAGE
1552                       ; LEAVES SYSTEM, IT WON'T BE NECESSARY TO CLEAR IT FROM TRANSACTION
1553                       ; TABLE AS WE ARE ABOUT TO BYPASS PUTTING IT INTO TABLE.
1554  18D3 3E00            MVI   A,00H            ;A=0
1555  18D5 329742          STA   IDENT
1556  18D8 329842          STA   IDENT+1          ;IDENT = 0000H
1557  18DB C3E118          JMP   LDLAQ            ;JMP LDLAQ
1558                       ;
1559                       ; LOAD 6 ASCII BYTES OF PACKAGE CODE INTO 1ST AVAILABLE 6 LOC IN
1560                       ; TRANSACTION TABLE BY CALLING DSPAK. RET WITH IDENT= ADDRESS IN
1561                       ; TRANSACTION TABLE WHERE CODE IS STORED.
1562  18DE CD0619  SCODE:  CALL  DSPAK
1563                       ;
1564                       ; LOAD MEMORY JDV AND IDENT INTO LAQ FOR THIS INDUCT
1565  18E1 C1      LDLAQ:  POP   B                ;B,C= JDV,JINDU
1566  18E2 79              MOV   A,C              ;A=INDUCT #
1567  18E3 CD9927          CALL  CPPPP            ;POINT H,L TO BYTE 1 IN PPDB
1568  18E6 23              INX   H
1569  18E7 22EB42          SHLD  PPDBP            ;OUTPUT POINTER
1570  18EA 110500          LXI   D,5              ;POINT H,L TO LAQSZ IN PPDB
1571  18ED 19              DAD   D                ;H,L=LAQSZ=# OF PACKAGES IN LAQ
1572  18EE 7E              MOV   A,M              ;A=LAQSZ VALUE
1573  18EF 34              INR   M                ;INCREMENT LAQSZ
1574  18F0 23              INX   H                ;H,L=LAQ1 LOC
1575  18F1 86              ADD   M                ;A=RELATINE NUMBER OF LOC USED UP IN LAQ
1576  18F2 E607            ANI   07H              ;MASK TO 3 BITS FOR WRAPAROUND
1577  18F4 5F              MOV   E,A
1578  18F5 1600            MVI   D,0              ;D,E= RELATIVE LOC OF AVAIL SPACE IN LAQ
1579  18F7 23              INX   H                ;H,L= START LOC OF LAQ IN PPDB
1580  18F8 19              DAD   D                ;POINT H,L TO ABSOLUTE LOC OF AVAILABLE
1581  18F9 19              DAD   D                ;    SPACE IN
1582  18FA 19              DAD   D                ;    LAQ
1583  18FB EB              XCHG                   ;D,E = AVAIL LOC IN LAQ
1584  18FC 219642          LXI   H,JDV            ;POINT H,L TO MEMORY JDV,IDENT
1585  18FF 0603            MVI   B,3              ;3 BYTES TO MOVE
1586  1901 CDCF10          CALL  MOVEB            ;COPY MEMORY JDV, IDENT INTO LAQ
1587  1904 04              INR   B                ;POSITIVE RETURN
1588  1905 C9              RET
1589                       ;        SUBROUTINE DSPAK
1590                       ;
1591                       ;        INPUTS: POINTERS TRSDB,TRSDP,TLAST. 6 BYTE ASCII PACKAGE CODE
1592                       ;                IN DASDB BYTES 2-7. STACK= JDV,JINDU
1593                       ;
1594                       ;        PROCESS: LOAD 6 BYTES OF ASCII CODE INTO TRANSACTION TABLE IN
1595                       ;                 1ST AVAILABLE 6 BYTES. TRANSACTION TABLE STARTS AT
1596                       ;                 LOC TRSDB, ENDS AT LOC TLAST, WITH TRSDP POINTING TO
1597                       ;                 LOC IN TABLE WHERE PREVIOUS 6 BYTES ENDED.
1598                       ;
1599                       ;        OUTPUTS: PACKAGE CODE IN ASCII RESIDES IN TRANSACTION TABLE
1600                       ;                 STARTING AT LOC POINTED TO BY H,L. TRSDP POINTS TO
1601                       ;                 START OF NEXT 6 BYTE BLOCK IN TRANSACTION TABLE,STACK
1602                       ;                 =JDV,JINDU
1603                       ; TEST TO SEE IF VALUE IN LOC TRSDP = 0
1604  1906 2A5D43  DSPAK:  LHLD  TRSDP    ;H,L= LOC IN TRANS TAB WHERE LEFT OFF
1605  1909 7E              MOV   A,M      ;A= VALUE IN LOC TRSDP
1606  190A D600            SUI   0
1607  190C CA1719          JZ    BCONT    ;JMP IF VALUE=0
1608  190F 3E00            MVI   A,0      ;OTHERWISE, RESET COUNTER
1609  1911 325A43          STA   TCONT    ;TCONT=0
1610  1914 C31E19          JMP   BTRDP    ;JMP TO BTRDP
1611  1917 3A5A43  BCONT:  LDA   TCONT    ;SET TCONT = TCONT+1
1612  191A 3C              INR   A
1613  191B 325A43          STA   TCONT
1614  191E 2A5D43  BTRDP:  LHLD  TRSDP    ;INCREMENT TRSDP POINTER
1615  1921 23              INX   H        ;TRSDP=TRSDP+1
1616  1922 225D43          SHLD  TRSDP
1617                       ;
1618                       ; TEST FOR WRAPAROUND OF TRANSACTION TABLE
1619  1925 3AF23D          LDA   TLAST    ;A= TLAST LSB
1620  1928 57              MOV   D,A      ;B= SAME
1621  1929 3A5D43          LDA   TRSDP    ;A= TRSDP LSB
1622  192C 92              SUB   D        ;A= TRSDP LSB - TLAST LSB
```

```
1623   192D C24919            JNZ   TTCNT     ;JMP IF NOT EQUAL
1624   1930 3AF33D            LDA   TLAST+1   ;A= TLAST MSB
1625   1933 57                MOV   D,A       ;D= SAME
1626   1934 3A5E43            LDA   TRSDP+1   ;A= TRSDP MSB
1627   1937 92                SUB   D         ;A= TRSDP MSB - TLAST MSB
1628   1938 C24919            JNZ   TTCNT     ;JMP IF NOT EQUAL
1629                        ;
1630                        ; WRAPAROUND BECAUSE TRSDP = TLAST
1631   193B 3E00              MVI   A,0       ;A=0
1632   193D 325A43            STA   TCONT     ;TCONT=0
1633   1940 2AF03D            LHLD  TRSDB     ;H,L= START LOC OF TRANSACTION TAB
1634   1943 225D43            SHLD  TRSDP     ;RESET TRSDP POINTER TO START
1635   1946 C30619            JMP   DSPAK     ;START AT TOP OF TABLE TO LOOK FOR 6 BYTE
1636                        ;
1637                        ; NO WRAP-AROUND. CHECK TO SEE IF 6 BYTES HAS BEEN FOUND
1638                        ;
1639   1949 3A5A43          TTCNT:  LDA   TCONT     ;A= TCONT
1640   194C D606                    SUI   6         ;TCONT=6?
1641   194E C20619                  JNZ   DSPAK     ;JMP IF TCONT NOT=6
1642   1951 3E00                    MVI   A,0       ;A=0
1643   1953 325A43                  STA   TCONT     ;TCONT=0 (RESET COUNTER)
1644                        ; SIX CONSECUTIVE BYTES FOUND. POINT H,L TO START OF 6 BYTES
1645   1956 2A5D43                  LHLD  TRSDP
1646   1959 2B                      DCX   H
1647   195A 2B                      DCX   H
1648   195B 2B                      DCX   H
1649   195C 2B                      DCX   H
1650   195D 2B                      DCX   H
1651   195E 2B                      DCX   H         ;H,L= START OF 6 AVAILABLE BYTES IN TABLE
1652   195F E5                      PUSH  H         ;STACK= SAME
1653   1960 EB                      XCHG            ;D,E=SAME
1654   1961 212843                  LXI   H,DASDB   ;H,L=DASDB LOC
1655   1964 010200                  LXI   B,2       ;B,C=2
1656   1967 09                      DAD   B         ;H,L=BYTE 2 OF DASDB= PACKAGE CODE START
1657   1968 010600                  LXI   B,6       ;B,C= 6
1658   196B CDC410                  CALL  MOVE      ;MOVE 6 BYTES FROM H,L TO DIE
1659   196E E1                      POP   H         ;H,L= LOC IN TRANSACTION TABLE OF CODE
1660   196F EB                      XCHG            ;D,E=SAME
1661                        ; PUT LOC OF STORED PAK CODE INTO MEMORY IDENT
1662   1970 219742                  LXI   H,IDENT   ;H,L= IDENT LSB LOC
1663   1973 73                      MOV   M,E       ;IDENT LSB= REG E
1664   1974 23                      INX   H         ;H,L= IDENT MSB LOC
1665   1975 72                      MOV   M,D       ;IDENT MSB= REG D
1666   1976 C9                      RET             ;RET TO DPLAQ WITH IDENT=LOC IN TRANS TAB
1667                        ;       SUBROUTINE DSOUT
1668                        ;
1669                        ;   WILL SEND DAS THE 1ST MESSAGE POINTED TO BY SMP IN FS. DEFAULTS TO
1670                        ;   IDLE MESSAGE IF NO MESSAGES IN FS. WILL RESEND LAST PSC MESSAGE IF
1671                        ;   CALLED FROM ICEXC WITH REG A=FFH, OR IF CALLED FROM DSDEC WITH
1672                        ;   REG A= FFH WILL SEND MESSAGE IN MOBUF(COULD BE SAME AS LAST PSC
1673                        ;   MESSAGE SENT)
1674                        ;
1675                        ;   INPUTS: IF REG A=FFH, MOBUF WILL CONTAIN DESIRED MESSAGE TO DAS.
1676                        ;           OTHERWISE, SMP WILL BE POINTING TO 1ST MESSAGE IN FS WAIT-
1677                        ;           ING TO BE SENT TO DAS (DEFAULT TO IDLE MESSAGE)
1678                        ;
1679                        ;   PROCESS: IF REG A=FFH ON ENTRY, SEND MOBUF TO DAS.  IF THERE ARE NO
1680                        ;            MESSAGES IN FS, LOAD IDLE MESSAGE INTO MOBUF AND SEND TO
1681                        ;            DAS. IF THERE IS A MESSAGE IN FS, LOAD IT INTO MOBUF AND
1682                        ;            SEND TO DAS AND THEN UPDATE SMP POINTER TO NEXT MESSAGE.
1683                        ;
1684                        ;   OUTPUTS: DAS MESSAGE OUTPUT TO 5080 BOARD
1685                        ;
1686                        ;
1687   1977 06FF            DSOUT:  MVI   B,0FFH          ;JMP TO SMOBF IF REG A=FFH IN ORDER
1688   1979 90                      SUB   B               ;TO RESEND MOBUF
1689   197A CAF119                  JZ    SMOBF
1690                        ; REG A MUST=00.
1691                        ; CHECK TO SEE IF ANY MESSAGES IN FS.
1692   197D 2A5843                  LHLD  SMP             ;H,L= SMP VALUE
1693   1980 7C                      MOV   A,H
1694   1981 B5                      ORA   L               ;SMP=0000H?
1695   1982 C2A019                  JNZ   YFS             ;JMP IF MESSAGES IN FS
1696                        ;
1697                        ; NO MESSAGES IN FS. DEFAULT TO IDLE MESSAGE.
1698   1985 3A2942                  LDA   RS              ;SET SEQ BYTE= TOGGLED RS
1699   1988 323843                  STA   MOBUF
1700   198B 3E49                    MVI   A,'I'           ;A= ASCII 'I'
1701   198D 323943                  STA   MOBUF+1
1702   1990 3E20                    MVI   A,20H           ;A= ASCII SPACE
1703   1992 213A43                  LXI   H,MOBUF+2       ;STORE INTO MOBUF BYTES 2-12 AS SPACES
1704   1995 060B                    MVI   B,11
1705   1997 77              LPMOB:  MOV   M,A
1706   1998 23                      INX   H
1707   1999 05                      DCR   B
1708   199A C29719                  JNZ   LPMOB
1709   199D C3F119                  JMP   SMOBF           ;JMP WHEN ALL BYTES 2-12 LOADED AS SPACE
1710                        ;
```

```
1711                    ; LOAD MESSAGE IN FS POINTED TO BY SMP INTO MOBUF
1712    19A0 3A2942     YFS:    LDA     RS              ;SET SEQ BYTE = TOGGLED RS
1713    19A3 323843             STA     MOBUF
1714    19A6 2A5843             LHLD    SMP             ;H,L= BYTE 1 OF MESSAGE IN FS
1715    19A9 23                 INX     H
1716    19AA 113943             LXI     D,MOBUF+1               ;D,E= MOBUF BYTE 1 LOC
1717    19AD 010C00             LXI     B,12            ;B,C= 12 BYTES TO MOVE
1718    19B0 CDC410             CALL    MOVE            ;MOVE MESSAGE FROM FS TO MOBUF
1719                    ;
1720                    ; TEST TO SEE IF THIS IS THE LAST MESSAGE WAITING IN FS. IT'S LAST
1721                    ; MESSAGE IF SMP=LMP(AND WE ALREADY KNOW LMP AND SMP ARE NON-ZERO)
1722    19B3 2A5843             LHLD    SMP             ;H,L= SMP VALUE
1723    19B6 EB                 XCHG                    ;D,E= SAME
1724    19B7 2A5443             LHLD    LMP             ;H,L= LMP VALUE
1725    19BA 7B                 MOV     A,E             ;A= SMP LSB
1726    19BB 95                 SUB     L               ;A= SMP LSB- LMP LSB
1727    19BC C2D919             JNZ     OTHFS           ;JMP IF SMP LSB NOT= SMP LSB
1728    19BF 7A                 MOV     A,D             ;A= SMP MSB
1729    19C0 94                 SUB     H               ;A= SMP MSB- LSB MSB
1730    19C1 C2D919             JNZ     OTHFS           ;JUMP IF NOT=
1731                    ;
1732                    ; THIS WAS LAST MESSAGE IN FS. RESET THE LOAD MESSAGE POINTER LMP AND
1733                    ; SEND MESSAGE POINTER SMP.
1734    19C4 2A5843             LHLD    SMP             ;H,L= SMP VALUE
1735    19C7 EB                 XCHG                    ;D,E= ORIG SMP VALUE
1736    19C8 3E00               MVI     A,0             ;A=0
1737    19CA 325843             STA     SMP             ;SMP=0000H
1738    19CD 325943             STA     SMP+1
1739    19D0 325443             STA     LMP             ;LMP=0000H
1740    19D3 325543             STA     LMP+1
1741    19D6 C3ED19             JMP     RTTCB           ;JMP TO RTTCB
1742                    ;
1743                    ; THIS IS NOT THE ONLY MESSAGE IN FS. POINT SMP POINTER TO NEXT ONE
1744    19D9 2A5843     OTHFS:  LHLD    SMP             ;H,L= SMP VALUE
1745    19DC EB                 XCHG                    ;D,E= ORIG SMP
1746    19DD 2A5843             LHLD    SMP             ;POINT H,L TO BYTE 14 OF MESSAGE TCB
1747    19E0 010E00             LXI     B,14
1748    19E3 09                 DAD     B               ;H,L= BYTE 14 LOC
1749    19E4 7E                 MOV     A,M             ;A= BYTE 14
1750    19E5 325843             STA     SMP             ;SMP= LSB OF LINK
1751    19E8 23                 INX     H               ;H,L= BYTE 15 LOC
1752    19E9 7E                 MOV     A,M             ;A= BYTE 15
1753    19EA 325943             STA     SMP+1           ;SMP+1 = MSB OF LINK
1754                    ;
1755                    ; RETURN MESSAGE TCB TO FS
1756    19ED EB         RTTCB:  XCHG                    ;H,L= ORIG SMP
1757    19EE CD4F11             CALL    PUTFS           ;RETURN TCB POINTED TO BY ORIG SMP TO FS
1758                    ;
1759                    ;LOAD CR,LF INTO MOBUF,CALC CHECKSUM OF MOBUF AND LOAD INTO MOBUF.
1760                    ; SEND MOBUF
1761                    ; TO DAS BY CALLING PUTDS. RESET DASIP
1762    19F1 213843     SMOBF:  LXI     H,MOBUF         ;POINT TO MOBUF
1763    19F4 CD211A             CALL    CHKSM           ;A= PRINTABLE ASCII CHECKSUM
1764    19F7 324543             STA     MOBUF+13        ;PUT CHKSM INTO MOBUF
1765    19FA 212843             LXI     H,DASDB         ;H,L= LOC OF DAS MESSAGE INPUT BUFFER
1766    19FD 224B43             SHLD    DASIP           ;RESET POINTER TO START OF BUFFER
1767    1A00 214643             LXI     H,MOBUF+14      ;H,L= BYTE 14 LOC
1768    1A03 360A               MVI     M,0AH           ;SET ASCII LF IN MOBUF
1769    1A05 23                 INX     H               ;H,L= BYTE 15 LOC
1770    1A06 360D               MVI     M,0DH           ;SET ASCII CR IN MOBUF
1771    1A08 CD0C1A             CALL    PUTDS           ;PUTDS STARTS TRANSMISSION OF MOBUF
1772    1A0B C9                 RET                     ;RET TO CALLER
1773                    ;       SUBROUTINE PUTDS
1774                    ;
1775                    ;       INPUTS: PSC TO DAS MESSAGE RESIDING IN BUFFER MOBUF. POINTER
1776                    ;               MBUFP POINTS TO BYTE 0 OF MOBUF.DTYBS=0
1777                    ;
1778                    ;       PROCESS: DOUT BYTE 0 OF MOBUF TO I/C 5080 BOARD.
1779                    ;
1780                    ;       OUTPUT: MOBUF BYTE 0 OUTPUT TO I/C 5080 BOARD.MBUFP POINTS TO
1781                    ;               BYTE 1 OF MOBUF. REST OF MESSAGE IN MOBUF WILL BE
1782                    ;               SENT BY INTERRUPT DRIVERS IN SUBROUTINE ICINT.ALSO
1783                    ;               SET DTYBS=1 FOR USE IN ICINT.
1784                    ;
1785    1A0C 3E01       PUTDS:  MVI     A,01H
1786    1A0E 324D43             STA     DTYBS   ;DTYBS=1 FOR USE IN ICINT
1787    1A11 2A5643             LHLD    MBUFP
1788    1A14 23                 INX     H
1789    1A15 225643             SHLD    MBUFP    ;MBUFP=MBUFP+1 (MOBUF BYTE 1)
1790    1A18 2A063D             LHLD    P5080   ;H,L= SIN ADDR FOR I/C 5080 BOARD
1791    1A1B 25                 DCR     H       ;H=FE FOR DOUT OPERATION
1792    1A1C 3A3843             LDA     MOBUF   ;A= MOBUF BYTE 0
1793    1A1F 77                 MOV     M,A     ;DOUT MOBUF BYTE 0 TO REG FEXX
1794    1A20 C9                 RET             ;RETURN TO DSOUT
1795                    ;       SUBROUTINE CHKSM
1796                    ;
1797                    ;       INPUT: H,L POINTS TO 1ST OF 13 CONSECUTIVE BYTES TO CALCULATE
```

```
1798                    ;               A PRINTABLE ASCII CHECKSUM FROM
1799                    ;
1800                    ;       PROCESS: SUM ALL 12 BINARY BYTES. MAKE BIT 7 OF RESULT=0 AND
1801                    ;                BIT 6 = COMPLEMENT OF BIT 5.
1802                    ;
1803                    ;       OUTPUT: PRINTABLE ASCII CHECKSUM IN REG A
1804                    ;
1805                    ; MASK OFF BIT 7 OF EACH BYTE BEFORE CHECKSUMMING
1806  1A21 060D         CHKSM:  MVI   B,13      ;SET LOOP COUNTER
1807  1A23 0E00                 MVI   C,0       ;INITIALIZE C REG TO ZERO
1808  1A25 7E           MCKSM:  MOV   A,M       ;A=BYTE
1809  1A26 E67F                 ANI   7FH       ;MASK BIT 7
1810  1A28 81                   ADD   C
1811  1A29 4F                   MOV   C,A       ;C=CURRENT CHKSM
1812  1A2A 23                   INX   H         ;POINT TO NEXT BYTE
1813  1A2B 05                   DCR   B         ;DEC COUNTER
1814  1A2C C2251A               JNZ   MCKSM     ;DO ALL 12 BYTES
1815  1A2F 79                   MOV   A,C       ;A=CHECKSUM
1816                    ;
1817                    ; SET BIT 7 = 0
1818  1A30 E67F                 ANI   7FH       ;BIT 7=0
1819  1A32 47                   MOV   B,A       ;STORE RESULT
1820                    ;
1821                    ; SET BIT 6 = COMPLEMENT OF BIT 5
1822  1A33 E620                 ANI   20H       ;LOOK AT BIT 5
1823  1A35 CA3C1A               JZ    BITO      ;JMP IF BIT 5 = 0
1824                    ; BIT 5 = 1. THEREFORE MAKE BIT 6 = 0.
1825  1A38 78                   MOV   A,B       ;A= CHECKSUM WITH BIT 7=0
1826  1A39 E6BF                 ANI   0BFH      ;ZERO OUT BIT 6
1827  1A3B C9                   RET             ;RET WITH PRINTABLE ASCII CHECKSUM IN A
1828                    ; BIT 5 = 0. THEREFORE MAKE BIT 6 = 1.
1829  1A3C 78           BITO:   MOV   A,B       ;A = CHECKSUM WITH BIT 7 = 0.
1830  1A3D F640                 ORI   40H       ;SET BIT 6=1
1831  1A3F C9                   RET             ;RET WITH PRINTABLE ASCII CHKSM IN REG A
1832                    ;       SUBROUTINE ZPACK
1833                    ;
1834                    ;       INPUTS: H,L POINTS TO LOC WHERE IT IS DESIRED TO CLEAR 6 BYTES
1835                    ;               IN TRANSACTION TABLE
1836                    ;       PROCESS: IF H,L IS WITHIN TRANSACTION TABLE, CLEAR THE 6 BYTES
1837                    ;                STARTING AT H,L. IF H,L NOT WITHIN TABLE, DO FATAL
1838                    ;                ERROR # 1
1839                    ;
1840                    ;       OUTPUT: 6 BYTES OF TRANSACTION TABLE CLEARED STARTING AT INPUT
1841                    ;               H,L LOCATIONS.
1842                    ;
1843                    ; CHECK TO SEE IF H,L WITHIN TRANSACTION TABLE LIMITS
1844  1A40 E5           ZPACK:  PUSH  H         ;STACK= H,L INPUT
1845  1A41 2AF03D               LHLD  TRSDB
1846  1A44 44                   MOV   B,H
1847  1A45 4D                   MOV   C,L       ;B,C=START ADDR OF TRANS TABLE
1848  1A46 2AF23D               LHLD  TLAST     ;H,L= END ADDR OF TRANS TAB
1849  1A49 EB                   XCHG            ;D,E=SAME
1850  1A4A E1                   POP   H         ;H,L= ADDR OF DESIRED DELETE
1851  1A4B E5                   PUSH  H         ;STACK= SAME
1852  1A4C 7D                   MOV   A,L       ;A= LSB OF DESIRED DELETE ADDR
1853  1A4D 91                   SUB   C         ;16-BIT
1854  1A4E 7C                   MOV   A,H       ;COMPARE
1855  1A4F 98                   SBB   B         ;HERE
1856  1A50 FA661A               JM    FATLE     ;JMP IF PASSED ADDR < TRANSLATION TABLE
1857                    ;
1858                    ; IF GET TO HERE, DESIRED DELETE ADDR IS GREATER THAN START OF TRANS TAB
1859                    ; NOW SEE IF DELETE ADDR < TLAST OF TRANS TABLE
1860  1A53 7B           TTOP:   MOV   A,E       ;A=LSB OF TLAST
1861  1A54 95                   SUB   L         ;16-BIT
1862  1A55 7A                   MOV   A,D       ;COMPARE
1863  1A56 9C                   SBB   H         ;HERE
1864  1A57 FA661A               JM    FATLE     ;JMP IF PASSED ADDR > TLAST
1865                    ;
1866                    ; H,L IS WITHIN TRANS TABLE. THEREFORE ZERO THE 6 LOC'S.
1867  1A5A E1           ZLOOP:  POP   H         ;H,L= ADDR OF DESIRED DELETE
1868  1A5B 3E00                 MVI   A,0       ;A=0
1869  1A5D 0606                 MVI   B,6       ;COUNT=6
1870  1A5F 77           ZTRAN:  MOV   M,A       ;ZERO THE LOC
1871  1A60 23                   INX   H         ;INCREMENT LOC
1872  1A61 05                   DCR   B         ;DEC COUNTER
1873  1A62 C25F1A               JNZ   ZTRAN     ;DO ALL 6 LOC
1874  1A65 C9                   RET             ;RET TO CALLER
1875                    ;
1876                    ; H,L NOT WITHIN TRANSACTION TABLE. DO FATAL ERROR #1
1877  1A66 3E01         FATLE:  MVI   A,01H
1878  1A68 CD0C2C               CALL  FATAL
1879  1A6B C9                   RET
1880                    ;       SUBROUTINE TRANP
1881                    ;
1882                    ;       INPUTS:H,L POINTS TO 1ST OF 6 BYTES DESIRED TO BE MOVED
1883                    ;
1884                    ;       PROCESS: MOVE 6 BYTES POINTED TO BY H,L INTO BUFFER BYT16
1885                    ;                IN BYTES 2-7 LOC
```

```
1886                    ;
1887                    ;       OUTPUTS: BYTES 2-7 OF BUFFER BYT16 CONTAIN THE 6 BYTES
1888                    ;                ORIGINALLY POINTED TO BY H,L
1889                    ;
1890    1A6C E5         TRANP:  PUSH  H         ;STACK = LOC OF 6 BYTES TO MOVE
1891    1A6D 211843             LXI   H,BYT16   ;H,L= BYT16 LOC
1892    1A70 110200             LXI   D,2
1893    1A73 19                 DAD   D         ;H,L= BYTE 2 OF BYT16
1894    1A74 EB                 XCHG            ;D,E= SAME
1895    1A75 010600             LXI   B,6       ;B,C= 6 COUNT
1896    1A78 E1                 POP   H         ;H,L= LOC OF 6 BYTES
1897    1A79 CDC410             CALL  MOVE      ;MOVE 6 BYTES FROM H,L TO D1E LOC
1898    1A7C C9                 RET             ;RETURN
1899                    ;       SUBROUTINE DSLD
1900                    ;
1901                    ;       INPUTS: MESSAGE TO BE LOADED INTO FS RESIDES IN BUFFER BYT16.
1902                    ;               LMP POINTS TO FS LOC WHERE LAST MESSAGE WAS LOADED.
1903                    ;
1904                    ;       OUTPUTS: MESSAGE IS IN FS POINTED TO BY LMP. MESSAGE IS LINKED
1905                    ;                TO OTHER MESSAGES IN FS.
1906                    ;
1907                    ;       PROCESS: GET 1ST AVAIL FS BLOCK AND LOAD MESSAGE BUFFER BYT16
1908                    ;                INTO IT. LINK THIS FS BLOCK TO PREVIOUS MESSAGE
1909                    ;                LOADED.
1910                    ;
1911    1A7D CDCB0F     DSLD:   CALL  GETFS             ;H,L= LOC OF AVAIL FS
1912    1A80 EB                 XCHG                    ;D,E= LOC OF AVAIL FS
1913    1A81 211843             LXI   H,BYT16           ;H,L= BYTE 0 LOC OF BYT16 BUFFER
1914    1A84 011000             LXI   B,16              ;B,C= 16 BYTES TO MOVE
1915    1A87 D5                 PUSH  D                 ;STACK= LOC OF AVAIL FS
1916    1A88 CDC410             CALL  MOVE              ;COPY BYT16 INTO FS
1917                    ;
1918                    ;       TEST TO SEE IF THIS IS THE 1ST MESSAGE IN FS
1919    1A8B 2A5443             LHLD  LMP               ;H,L= LMP VALUE
1920    1A8E 7C                 MOV   A,H               ;A= LMP MSB
1921    1A8F B5                 ORA   L                 ;OR WITH LMP LSB
1922    1A90 C29B1A             JNZ   MLINK             ;JMP IF LMP NOT ZERO
1923                    ; THIS IS THE 1ST MESSAGE IN FS. SET LMP AND SMP TO THIS FS LOC
1924    1A93 E1                 POP   H                 ;H,L= LOC IN FS WHERE MESSAGE WAS LOADED
1925    1A94 225843             SHLD  SMP               ;SMP= LOC IN FS OF MESSAGE
1926    1A97 225443             SHLD  LMP               ;LMP= LOC IN FS OF MESSAGE
1927    1A9A C9                 RET                     ;RET TO CALLER
1928                    ;  THIS IS NOT THE ONLY MESSAGE IN FS. LINK IT TO PREVIOUS MESSAGE IN FS
1929    1A9B 2A5443     MLINK:  LHLD  LMP               ;H,L= LMP LOC = LOC IN FS WHERE LAST
1930                                                    ;   MESSAGE WAS STORED
1931    1A9E EB                 XCHG                    ;D,E= LMP LOC OF LAST MESSAGE
1932    1A9F 210E00             LXI   H,14
1933    1AA2 19                 DAD   D                 ;H,L= BYTE 14 OF LAST MESSAGE
1934    1AA3 D1                 POP   D                 ;D,E= LOC WHERE NEW MESSAGE JUST STORED
1935    1AA4 73                 MOV   M,E               ;BYTE 14 = LSB OF NEW LOC
1936    1AA5 23                 INX   H                 ;POINT TO BYTE 15
1937    1AA6 72                 MOV   M,D               ;BYTE 15 OF PREVIOUS MESG=MSB OF NEW LOC
1938                    ;  BUMP LMP TO LOC IN FS WHERE NEW MESSAGE WAS JUST LOADED
1939    1AA7 EB                 XCHG                    ;H,L= ADDR IN FS WHERE NEW MESSAGE PUT
1940    1AA8 225443             SHLD  LMP               ;LMP= ADDR IN FS OF NEW MESSAGE
1941    1AAB C9                 RET                     ;RET TO CALLER
1942                    ;
1943                    ;RUNNING MODE INITIALIZATION.
1944                    ;
1945                    ;INPUT: INTERRUPTS DISABLED.
1946                    ;ALL REGISTERS EFFECTED.
1947                    ;
1948    1AAC 3E3C       RINI:   MVI   A,'<'     ;MODE
1949    1AAE 322542             STA   MODE      ;  = "RUN"
1950    1AB1 CD9710             CALL  LNKFS     ;LINK FREE STORAGE
1951    1AB4 3A0B3D             LDA   NXLTB     ;IF TRANSLATE OPTION SPECIFIED
1952    1AB7 B7                 ORA   A         ;  THEN INITIALIZE
1953    1AB8 C4002F             CNZ   XLINI     ;  TRANSLATION DATA
1954                    ;
1955                    ;POLL PP AND TE PHOTOS TO SEE IF PACKAGE IS PRESENT
1956                    ;
1957    1ABB 3EFF               MVI   A,0FFH    ;FOR HIGH-TRUE
1958    1ABD D301               OUT   1         ;  PHOTOS
1959    1ABF 3A6F42             LDA   PPCHN     ;A= PM5008 COMPOSITE CH # FOR PP # 1
1960    1AC2 47                 MOV   B,A       ;B= COMPOSITE CH #
1961    1AC3 0E01               MVI   C,1       ;C= INDUCT # = 1
1962                    ;
1963    1AC5 C5         RPLPP:  PUSH  B         ;SAVE COMPOSITE CH #.  INDUCT #
1964    1AC6 CD171B             CALL  POLL      ;POLL PP
1965    1AC9 04                 INR   B         ;B= TE CH #
1966    1ACA DAD31A             JC    RPP1      ;IF PP NOT BLOCKED
1967    1ACD CD171B             CALL  POLL      ;POLL TE
1968    1AD0 D2E01A             JNC   RPP0      ;IF BLOCKED
1969                    ;
1970    1AD3 C1         RPP1:   POP   B         ;B=COMPOSITE CH #, C= INDUCT #
1971    1AD4 79                 MOV   A,C       ;A= INDUCT #
1972    1AD5 CD9927             CALL  CPPPP     ;POINT H,L TO PPDB
```

```
1973   1AD8 3601            MVI   M,1        ;SET PACKAGE-PRESENT &
1974   1ADA 23              INX   H          ;  PACKAGE-WAITING
1975   1ADB 3601            MVI   M,1        ;  FLAGS
1976   1ADD C3E81A          JMP   RPPLP
1977                ;
1978   1AE0 C1      RPP0:   POP   B          ;B= COMPOSITE CH #.
1979   1AE1 C5              PUSH  B          ;  C= INDUCT #
1980   1AE2 0601            MVI   B,1        ;START RELEASE
1981   1AE4 CD0F11          CALL  OPUT       ;  BELT
1982   1AE7 C1              POP   B          ;B= COMPOSITE CH #, C= INDUCT #
1983                ;
1984   1AE8 04      RPPLP:  INR   B          ;B= NEXT PP CH #
1985   1AE9 04              INR   B          ;
1986   1AEA 0C              INR   C          ;INC INDUCT #
1987   1AEB 3A033D          LDA   NINDU      ;DOUNTIL
1988   1AEE B9              CMP   C          ;  INDUCT #
1989   1AEF D2C51A          JNC   RPLPP      ;  > NINDU
1990                ;
1991                ;POLL LF TO SET LFDB FLAGS
1992                ;
1993   1AF2 3A3C3D  RPLLF:  LDA   LFO        ;IF
1994   1AF5 B7              ORA   A          ;  THERE ARE
1995   1AF6 CA131B          JZ    REINI      ;  LF'S
1996   1AF9 3A7442          LDA   TRCHN      ;A= PM5008 COMPOSITE CH #
1997   1AFC DE02            SBI   2          ;  FOR LAST LF
1998   1AFE 47              MOV   B,A        ;B= COMPOSITE CH #
1999   1AFF 3A053D          LDA   NDVT       ;A= # LF'S = COUNT
2000   1B02 4F              MOV   C,A        ;C= COUNT
2001   1B03 C5      RPLF:   PUSH  B          ;SAVE CH #, COUNT
2002   1B04 CD171B          CALL  POLL       ;POLL LF # C
2003   1B07 C1              POP   B          ;B= CH #,
2004   1B08 C5              PUSH  B          ;  C= DIVERT #
2005   1B09 DCE720          CC    LFBPC      ;IF LF BLOCKED THEN SET FLAG IN LFDB
2006   1B0C C1              POP   B          ;B= COMPOSITE CH #, C= DIVERT #
2007   1B0D 05              DCR   B          ;B= COMPOSITE CH #
2008   1B0E 05              DCR   B          ;  FOR PREVIOUS LF
2009   1B0F 0D              DCR   C          ;DECREMENT DIVERT # (COUNT)
2010   1B10 C2031B          JNZ   RPLF       ;  UNTIL =0
2011                ;
2012                ;REINITIALIZE PM5008'S
2013                ;
2014   1B13 CD190C  REINI:  CALL  DEVII      ;REINITIALIZE
2015   1B16 C9              RET
2016                ;;
2017                ;POLL PM5008 COMPOSITE CHANNEL # B FOR HIGH-TRUE SIGNAL LEVEL.
2018                ;  B>=0.
2019                ;
2020                ;INPUT: A=0 MEANS LOW-TRUE, A=0FFH MEANS HIGH-TRUE.
2021                ;OUTPUT: FLAG C SET IF SIGNAL TRUE.
2022                ;ALL REGISTERS EXCEPT B EFFECTED.
2023                ;
2024   1B17 3EFF    POLL:   MVI   A,0FFH     ;FOR HIGH-TRUE PHOTOS
2025                ;
2026   1B19 D301    POLL1:  OUT   1          ;SET HIGH-TRUE SELECT BITS
2027   1B1B 3A6C42          LDA   P5008      ;H,L
2028   1B1E 6F              MOV   L,A        ;  = COUT AD FOR
2029   1B1F 26FF            MVI   H,0FFH     ;  1ST PM5008 BOARD
2030   1B21 78              MOV   A,B        ;A= COMPOSITE CH #
2031   1B22 CD280D          CALL  CPMSK      ;CP COUT AD & BIT MASK
2032   1B25 3606            MVI   M,06H      ;ENABLE BOARD & CLEAR ITS FLAGS
2033   1B27 25              DCR   H          ;H,L= DOUT AD
2034   1B28 3600            MVI   M,0        ;FOR SPURIOUS INT'S (SEE PM5008 MANUAL)
2035   1B2A 77              MOV   M,A        ;DOUT TO ENABLE CH
2036   1B2B 24              INR   H          ;H,L= SIN AD
2037   1B2C 7E              MOV   A,M        ;SIN
2038   1B2D 1F              RAR              ;GROUP SELECT BIT TO FLAG C
2039   1B2E C9              RET
2040                ;;
2041                ;RUNNING MODE PHOTO SENSOR CELL PROCESSOR.
2042                ;
2043                ;INPUT: B=DEVIC, C=INDEX.
2044                ;ALL REGISTERS EFFECTED.
2045                ;
2046   1B2F 78      PHPC:   MOV   A,B        ;CASENTRY DEVIC= 2 TO 11
2047   1B30 3D              DCR   A          ;
2048   1B31 1E0A            MVI   E,10       ;CASE SWITCH
2049   1B33 CDA20C          CALL  CASSW      ;  ON 10 CASES
2050   1B36 651C            DW    PPBPC      ;DEVIC=2: PP-BLOCKING
2051   1B38 C51C            DW    TECPC      ;DEVIC=3: TE-CLEARING
2052   1B3A 2E1D            DW    IDBPC      ;DEVIC=4: ID-BLOCKING
2053   1B3C 4F1B            DW    PSRET      ;FOR FUTURE EXPANSION
2054   1B3E 6E1D            DW    UDBPC      ;DEVIC=6: UD-BLOCKING
2055   1B40 3120            DW    UDCPC      ;DEVIC=7: UD-CLEARING
2056   1B42 0043            DW    DCBPC      ;DEVIC=8: DC-BLOCKING
2057   1B44 4F1B            DW    PSRET      ;FOR FUTURE EXPANSION
2058   1B46 E720            DW    LFBPC      ;DEVIC=10: LF-BLOCKING
2059   1B48 EE20            DW    LFCPC      ;DEVIC=11: LF-CLEARING
```

```
2060                    ;
2061                    ;ERROR EXIT AND NO-OPERATION EXIT
2062                    ;
2063    1B4A 3E0A               MVI   A,10      ;DEFAULT CASE:
2064    1B4C CD080F             CALL  ERPC      ;  ERROR 10
2065    1B4F C9         PSRET:  RET             ;NO OPERATION
2066                    ;;
2067                    ;RUNNING MODE KEYBOARD INPUT CHARACTER PROCESSOR.
2068                    ;
2069                    ;INPUT: C= KEYBOARD NUMBER, A= INPUT CHARACTER,
2070                    ;  KBDBP=HL POINTS TO BYTE 4 IN KBDB.
2071                    ;OUTPUT: CHAR.
2072                    ;ALL REGISTERS EFFECTED.
2073                    ;
2074    1B50 32DE42     KBPC:   STA   CHAR              ;SAVE INPUT CHAR
2075    1B53 47                 MOV   B,A               ;B=CHAR
2076    1B54 7E                 MOV   A,M               ;IF ER IN KBDB=0 THEN
2077    1B55 B7                 ORA   A                 ;   SET FLAG Z
2078    1B56 C2731B             JNZ   IKERZ             ;IF KB NOT IN ER MODE
2079                    ; TEST IF IN I/C MODE
2080    1B59 3A2B42             LDA   ICMO
2081    1B5C D601               SUI   01H               ;ICMO=1?
2082    1B5E C2EF1B             JNZ   NUMER             ;JMP IF NOT I/C
2083                    ; TEST FOR HALF DAS
2084    1B61 3A0640             LDA   HDSEN
2085    1B64 D601               SUI   01H
2086    1B66 CAEF1B             JZ    NUMER             ;JMP IF HALF DAS
2087                    ;TEST IF IN 'PROGRAM' MODE, I.E., IS IT DESIRED TO PROGRAM
2088                    ;TRANSLATION TABLES WHILE IN HOST MODE?
2089    1B69 3E2E               MVI   A,2EH             ;A=ASCII + = 'SEND' KEY IN PROGRAM MODE
2090    1B6B B8                 CMP   B                 ;COMPARE '+' WITH INPUT CHAR
2091    1B6C CAEF1B             JZ    NUMER             ;JMP IF 'SEND' KEY WAS HIT WHILE IN
2092                                                    ;  PROGRAM MODE
2093    1B6F D2EF1B             JNC   NUMER             ;JMP IF A DIGIT WAS HIT WHILE IN
2094                                                    ;  PROGRAM MODE
2095    1B72 C9                 RET                     ;RET-THEREFORE WON'T PROCESS ANY KB
2096                                                    ;INPUTS EXCEPT 'SEND' OR A DIGIT WHILE
2097                                                    ;IN PROGRAM MODE IF NO ERRORS ARE ON KB
2098                                                    ; (WILL ALSO PROCESS '<' AND '>')
2099                                                    ;AND SYSTEM IS IN HOST MODE BUT NOT IN
2100                                                    ;HALF DAS MODE
2101                    ;
2102                    ;
2103                    ; TEST IF IN I/C MODE
2104    1B73 3A2B42     IKERZ:  LDA   ICMO
2105    1B76 D601               SUI   01H
2106    1B78 C2411C             JNZ   KERNZ             ;JMP IF NOT I/C
2107                    ;
2108                    ; TEST FOR HALF DAS
2109    1B7B 3A0640             LDA   HDSEN
2110    1B7E D601               SUI   01H
2111    1B80 CA411C             JZ    KERNZ             ;JMP IF HALF DAS
2112                    ;
2113                    ; TEST IF INPUT WAS A 'CLEAR'
2114    1B83 3ADE42             LDA   CHAR              ;PICK UP INPUT CHAR
2115    1B86 F630               ORI   30H               ;NORMAL ILED INPUT CHAR
2116    1B88 FE3A               CPI   ':'               ;COMPARE WITH 'CLEAR' CHAR
2117    1B8A CA8E1B             JZ    PRCLR             ;JMP IF 'CLEAR' WAS HIT
2118    1B8D C9                 RET                     ;RETURN AS SOME KEY OTHER THAN CLEAR WAS
2119                                                    ;HIT WHILE IN I/C MODE(AND NOT HALF DAS)
2120                    ;
2121                    ; PROCESS 'CLEAR' KEY
2122                    ;
2123                    ; IF ERROR ON KB IS A 1,2,6,11,OR 12, DO SAME THING NORMALLY DONE FOR
2124                    ; 'CLEAR' KEY. IF ANY OTHER ERROR, ONLY CLEAR ERROR OFF KB AND TURN RUN
2125                    ; LIGHT ON KB BACK ON.
2126    1B8E 79         PRCLR:  MOV   A,C               ;A=KB#
2127    1B8F CD170D             CALL  CPKBP             ;H,L POINTS TO KBI1 IN KBDB
2128    1B92 110600             LXI   D,6               ;POINT TO SND LOC(CODE IS TEMP STORED
2129    1B95 19                 DAD   D                 ;$ HERE IN ERRPC)
2130    1B96 7E                 MOV   A,M               ;A= ERROR CODE
2131    1B97 57                 MOV   D,A               ;STORE ERROR CODE IN D
2132    1B98 D601               SUI   1
2133    1B9A CAB91B             JZ    KRLAQ             ;JMP IF ERROR #1
2134    1B9D 7A                 MOV   A,D
2135    1B9E D602               SUI   2
2136    1BA0 CAB91B             JZ    KRLAQ             ;JMP IF ERROR #2
2137    1BA3 7A                 MOV   A,D
2138    1BA4 D606               SUI   6                 ;JMP IF ERROR #6
2139    1BA6 CAB91B             JZ    KRLAQ
2140    1BA9 7A                 MOV   A,D
2141    1BAA D60B               SUI   11
2142    1BAC CAB91B             JZ    KRLAQ             ;JMP IF ERROR #11
2143    1BAF 7A                 MOV   A,D
2144    1BB0 D60C               SUI   12
2145    1BB2 CAB91B             JZ    KRLAQ             ;JMP IF ERROR #12
2146                    ;
2147                    ; THEREFORE JUST WANT TO CLEAR ERROR FROM KB AND TURN RUN LIGHT ON KB
```

```
2148                    ; BACK ON(FOR NON-INDUCT KB'S AND ERRORS 0,3,4,5,7,8,9,10,13,14)
2149    1BB5 CDC90C     KLRKB:  CALL    CLRKB           ;CLEAR KBDB, KB DISPLAY, TURN BACK ON
2150                                                    ;MODE LIGHT
2151    1BB8 C9                 RET                     ;RETURN
2152                    ;
2153                    ; BEFORE CLEARING LAQ DURING CALL MCLPC, CLEAR THE PACKAGES CURRENTLY
2154                    ; IN LAW FROM TRANSACTION TABLE
2155    1BB9 3A033D     KRLAQ:  LDA     NINDU           ;A=NUMBER OF INDUCTS
2156    1BBC B9                 CMP     C               ;COMPARE WITH KB# IN REG C
2157    1BBD DAB51B             JC      KLRKB           ;JUMP IF IT'S A NON-INDUCT KB
2158    1BC0 79                 MOV     A,C             ;A= INDUCT #
2159    1BC1 CD9927             CALL    CPPPP           ;POINT H,L TO PPDB FOR THIS INDUCT
2160    1BC4 C5                 PUSH    B               ;C= KB# NOW ON STACK
2161    1BC5 110600             LXI     D,6             ;D,E= 6
2162    1BC8 19                 DAD     D               ;H,L= LOC OF LAQSZ IN PPDB
2163    1BC9 46                 MOV     B,M             ;B= LAQSZ = LOOP COUNTER
2164    1BCA 23                 INX     H               ;H,L= LOC OF LAQ1 IN PPDB
2165    1BCB 5E                 MOV     E,M             ;E= LAQ1
2166    1BCC 1600               MVI     D,0             ;D,E= LAQ1
2167    1BCE 23                 INX     H               ;H,L= LOC OF LAQ IN PPDB
2168    1BCF 19                 DAD     D               ;POINT H,L TO LOC IN PPDB
2169    1BD0 19                 DAD     D               ;   OF 1ST PACKAGE THAT
2170    1BD1 19                 DAD     D               ;   HASN'T YET BEEN INDUCTED
2171    1BD2 05         LGLAQ:  DCR     B               ;COUNT = COUNT -1
2172    1BD3 FAEB1B             JM      MGLAQ           ;JMP IF NO PACKAGES LEFT TO CLEAR IN LAQ
2173    1BD6 23                 INX     H               ;POINT TO IDENT LSB
2174    1BD7 5E                 MOV     E,M             ;E= IDENT LSB
2175    1BD8 23                 INX     H               ;POINT TO IDENT MSB
2176    1BD9 56                 MOV     D,M             ;D,E= IDENT
2177    1BDA E5                 PUSH    H               ;STACK= LOC OF IDENT MSB
2178    1BDB EB                 XCHG                    ;H,L= IDENT VALUE
2179    1BDC 7C                 MOV     A,H             ;A= IDENT MSB
2180    1BDD B5                 ORA     L               ;OR WITH IDENT LSB
2181    1BDE CAE61B             JZ      NGLAQ           ;JMP IF IDENT = 0
2182    1BE1 C5                 PUSH    B               ;SAVE COUNTER
2183    1BE2 CD401A             CALL    ZPACK           ;REMOVE PACK CODE FROM TRANSACTION TABLE
2184    1BE5 C1                 POP     B               ;B= COUNT AGAIN
2185    1BE6 E1         NGLAQ:  POP     H               ;H,L= LOC OF IDENT MSB
2186    1BE7 23                 INX     H               ;H,L= LOC OF NEXT PACK JDV VALUE
2187    1BE8 C3D21B             JMP     LGLAQ           ;JMP TO LGLAQ
2188    1BEB C1         MGLAQ:  POP     B               ;B= CB# AGAIN
2189    1BEC C3611C             JMP     KCLR            ;JMP TO KCLR TO CLEAR LAQ,ETC.
2190    1BEF 3ADE42     NUMER:  LDA     CHAR            ;A= CHAR
2191    1BF2 F630               ORI     30H         ;NORMALIZED INPUT CHAR
2192    1BF4 FE3A               CPI     ':'         ;IF CHAR IS
2193    1BF6 D2111C             JNC     KNDIG       ;   '0' TO '9'
2194    1BF9 CD9E0D             CALL    DIGPC       ;PROCESS DIGIT
2195                    ;
2196                    ;   ;AUTOMATIC "SEND" BY FIXED LENGTH CODE
2197                    ;
2198    1BFC 3ADE42             LDA     CHAR        ;"SEND" PACKAGE
2199    1BFF E6F0               ANI     0F0H        ;   ONLY IF
2200    1C01 FE30               CPI     30H         ;   KEYSWITCH
2201    1C03 C0                 RNZ                 ;   IS IN "RUN" POSITION
2202    1C04 3A323D             LDA     SENDO       ;A= LENGTH OF FIXED-LENGTH CODE
2203    1C07 E60F               ANI     0FH         ;   WITHOUT "SEND"
2204    1C09 C8                 RZ                  ;IF LENGTH SPECIFIED ( > 0 )
2205    1C0A 23                 INX     H           ;POINT H,L TO BYTE 5 IN KBDB
2206    1C0B BE                 CMP     M           ;IF CODE ENTERED HAS REACHED
2207    1C0C C0                 RNZ                 ;   THAT LENGTH
2208    1C0D CD5B21             CALL    SNPC        ;"SEND" AUTOMATICALLY
2209    1C10 C9                 RET
2210                    ;
2211                    ;NON-NUMERIC CHAR
2212                    ;
2213    1C11 CA611C     KNDIG:  JZ      KCLR        ;IF NOT= ':'
2214    1C14 FE3C               CPI     '<'         ;IF < '<' THEN
2215    1C16 D21D1C             JNC     KNSND       ;   MUST BE ';'
2216    1C19 CD3D21             CALL    SNDPC       ;NORMAL "SEND"
2217    1C1C C9                 RET
2218                    ;
2219    1C1D C22B1C     KNSND:  JNZ     KNLT        ;IF = '<'
2220    1C20 3A0B3D             LDA     NXLTB       ;USED ONLY FOR
2221    1C23 B7                 ORA     A           ;   TRANSLATION
2222    1C24 CA3B1C             JZ      KBERR       ;   OPTION
2223    1C27 CDB22F             CALL    RLTPC       ;TRANSLATE FROM DV # TO CODE
2224    1C2A C9                 RET
2225                    ;
2226    1C2B FE3E       KNLT:   CPI     '>'         ;IF IT IS
2227    1C2D C23B1C             JNZ     KBERR       ;   ">" KEY
2228    1C30 3A0B3D             LDA     NXLTB       ;USED ONLY FOR
2229    1C33 B7                 ORA     A           ;   TRANSLATION
2230    1C34 CA3B1C             JZ      KBERR       ;   OPTION
2231    1C37 CD6830             CALL    RGTPC       ;EXAMINE OR CHANGE DV #
2232    1C3A C9                 RET
2233                    ;
2234                    ;ERRONEOUS CHAR
2235                    ;
```

```
2236  1C3B 3E08     KBERR:   MVI   A,8        ;ERROR
2237  1C3D CD130F            CALL  ERRPC      ;  8
2238  1C40 C9                RET
2239                 ;
2240                 ;
2241                 ;KEYBOARD PROCESSOR WHEN FLAG ER NOT=0, I.E., KEYBOARD SPECIAL MODES
2242                 ;
2243                 ;INPUT: C= KEYBOARD #, B=CHAR= INPUT CHARACTER.
2244                 ;
2245  1C41 3ADE42    KERNZ:   LDA   CHAR       ;PICK UP INPUT CHARACTER
2246  1C44 F630              ORI   30H        ;NORMALIZED CHAR
2247  1C46 FE3A              CPI   ':'        ;IF NOT
2248  1C48 CA5D1C            JZ    KBCLR      ;  "CLEAR"
2249  1C4B 7E                MOV   A,M        ;A= SPECIAL KB MODE FLAG ER IN KBDB
2250  1C4C D60A              SUI   10         ;ER=10 IS FOR MANIPULATING
2251  1C4E CABA30            JZ    RM0PC      ;  TRANSLATE TABLE
2252  1C51 3D                DCR   A          ;ER=11 IS FOR MANIPULATING
2253  1C52 CA0000            JZ    0000H      ;  COUNT TABLE
2254  1C55 3D                DCR   A          ;ER=12 IS FOR MANIPULATING
2255  1C56 CA0000            JZ    0000H      ;  PACKAGE COUNTS
2256  1C59 CD1C0E            CALL  ERMPC      ;CALL SPECIAL KB MODE PROCESSOR
2257  1C5C C9                RET
2258                 ;
2259                 ;KB "CLEAR" PROCESSING
2260                 ;
2261  1C5D 7E        KBCLR:   MOV   A,M        ;IGNORE ALL INPUTS IF
2262  1C5E FE0F              CPI   0FH        ;  KB IS IN
2263  1C60 C8                RZ               ;  LOCK-OUT MODE
2264                 ;
2265  1C61 CD1422    KCLR:    CALL  MCLPC      ;"MASTER CLEAR"
2266  1C64 C9                RET
2267                 ;;
2268                 ;PACKAGE-PRESENT-PHOTO-BLOCKING PROCESSOR.
2269                 ;
2270                 ;INPUT: C= INDUCT NUMBER, PPDB.
2271                 ;OUTPUT: PPDB.
2272                 ;ALL REGISTERS & PPDB EFFECTED.
2273                 ;
2274  1C65 79        PPBPC:   MOV   A,C        ;A = INDUCT #
2275  1C66 CD9927            CALL  CPPPP      ;POINT H,L TO PPDB
2276  1C69 7E                MOV   A,M        ;PICK UP FLAG PP IN PPDB
2277  1C6A 3601              MVI   M,1        ;PP IN PPDB = 1
2278  1C6C B7                ORA   A          ;IF ORIGINAL PP IN PPDB
2279  1C6D C2B21C            JNZ   PPBES      ;  = 0
2280                 ;
2281  1C70 E5                PUSH  H          ;SAVE PPDB POINTER
2282  1C71 C5                PUSH  B          ;SAVE INDUCT #
2283  1C72 79                MOV   A,C        ;A = INDUCT #
2284  1C73 2A5942            LHLD  OUTDB      ;IF RELEASE IS STOPPED
2285  1C76 CD280D            CALL  CPMSK      ;  THEN SET
2286  1C79 A6                ANA   M          ;  FLAG Z
2287  1C7A C1                POP   B          ;RESTORE C= INDUCT #
2288  1C7B E1                POP   H          ;RESTORE POINTER TO PPDB
2289  1C7C CAB91C            JZ    PPBPE      ;IF RLS STOPPED THEN ERROR
2290                 ;
2291                 ;LEGITIMATE PACKAGE BLOCKING PP
2292                 ;
2293  1C7F C5                PUSH  B          ;SAVE C= INDUCT #
2294  1C80 E5                PUSH  H          ;SAVE PPDB POINTER
2295  1C81 CDA20F            CALL  GITRE      ;GET TREE PARAMETERS FOR INDUCT
2296  1C84 2A0A40            LHLD  PPD        ;SET DE = # PPI PULSES
2297  1C87 EB                XCHG             ;  TO TIME OUT
2298  1C88 E1                POP   H          ;POINT TO
2299  1C89 23                INX   H          ;  BYTE 2
2300  1C8A 23                INX   H          ;  IN PPDB
2301  1C8B E5                PUSH  H          ;SAVE POINTER
2302  1C8C 23                INX   H          ;POINT TO BYTE 4
2303  1C8D 23                INX   H          ;  IN PPDB
2304  1C8E 019123            LXI   B,PPDTO    ;SCHEDULE EVENT PPDTO
2305  1C91 CD192D            CALL  RSCHD      ;  TO TIMEOUT PP DWELL
2306  1C94 D1                POP   D          ;POINT DE TO BYTE 2 IN PPDB
2307  1C95 210400            LXI   H,4        ;POINT HL TO BYTE 6
2308  1C98 19                DAD   D          ;  IN PPDB
2309  1C99 7E                MOV   A,M        ;PICK UP LAQ CURRENT SIZE
2310  1C9A EB                XCHG             ;POINT TO PPDB BYTE 2
2311  1C9B C1                POP   B          ;RESTORE C= INDUCT #
2312  1C9C B7                ORA   A          ;IF LAQ NOT
2313  1C9D CAA41C            JZ    PPWTG      ;  EMPTY
2314  1CA0 CD7022            CALL  PSEQR      ;SEQUENCE PACKAGE TO BE INDUCTED
2315  1CA3 C9                RET
2316                 ;
2317                 ;  ;PACKAGE HAS TO WAIT TO BE ENCODED
2318                 ;
2319  1CA4 2B        PPWTG:   DCX   H          ;POINT H,L TO BYTE 1 IN PPDB
2320  1CA5 7E                MOV   A,M        ;A= PWTNG IN PPDB
2321  1CA6 3601              MVI   M,1        ;PWTNG IN PPDB = 1
2322  1CA8 B7                ORA   A          ;IF ORIGINAL PWTNG IN PPDB
2323  1CA9 C2BF1C            JNZ   PPBFX      ;  = 0
```

```
2324   1CAC 0602            MVI   B,2        ;RELEASE
2325   1CAE CD0F11          CALL  OPUT       ;  OFF
2326   1CB1 C9              RET
2327                ;
2328                ;ILLEGITIMATE PP-BLOCKED INTERRUPT
2329                ;
2330   1CB2 C5      PPBES:  PUSH  B          ;STOP RELEASE MECHANISM
2331   1CB3 0602            MVI   B,2        ;  TO AVOID RELEASING
2332   1CB5 CD0F11          CALL  OPUT       ;  MULTIPLE
2333   1CB8 C1              POP   B          ;  PACKAGES
2334   1CB9 3E0C    PPBPE:  MVI   A,12       ;INDUCT ERROR
2335   1CBB CD130F          CALL  ERRPC      ;  12
2336   1CBE C9              RET
2337                ;
2338                ;FLAG PP=0, RELEASE OFF, BUT FLAG PWTNG NOT=0
2339                ;
2340   1CBF 3E0C    PPBFX:  MVI   A,12       ;FATAL ERROR
2341   1CC1 CD0C2C          CALL  FATAL      ;  # 12
2342   1CC4 C9              RET
2343                ;;
2344                ;TRAILING-EDGE-PHOTO-CLEARING PROCESSOR.
2345                ;
2346                ;INPUT: C= INDUCT #, PPDB, RLSDB, OUTDB.
2347                ;OUTPUT: PPDB, RLSDB, EVENT TECMG OR CMG.
2348                ;ALL REGISTERS EFFECTED.
2349                ;
2350   1CC5 79      TECPC:  MOV   A,C        ;A = INDUCT #
2351   1CC6 CD9927          CALL  CPPPP      ;POINT H,L TO PPDB
2352   1CC9 35              DCR   M          ;RESET PP IN PPDB
2353   1CCA C2261D          JNZ   TCPE0      ;IF NOT ALREADY RESET
2354                ;
2355   1CCD C5              PUSH  B          ;SAVE C= INDUCT #
2356   1CCE 79              MOV   A,C        ;A = INDUCT #
2357   1CCF 2A5942          LHLD  OUTDB      ;IF RLS IS STOPPED
2358   1CD2 CD280D          CALL  CPMSK      ;  THEN SET
2359   1CD5 A6              ANA   M          ;  FLAG Z
2360   1CD6 C1              POP   B          ;C = INDUCT #
2361   1CD7 CA281D          JZ    TCPEX      ;IF RLS RUNNING
2362                ;
2363   1CDA CDA227          CALL  CPRLP      ;POINT H,L TO TCBP IN RLSDB
2364   1CDD 5E              MOV   E,M        ;POINT D,E TO
2365   1CDE 23              INX   H          ;  PACKAGE BEING
2366   1CDF 56              MOV   D,M        ;  RELEASED
2367   1CE0 7B              MOV   A,E        ;IF TCB
2368   1CE1 B2              ORA   D          ;  NOT
2369   1CE2 CA201D          JZ    TCPFE      ;  NULL
2370                ;
2371                ;D,E POINTS TO TCB FOR PACKAGE BEING RELEASED
2372                ;
2373   1CE5 AF              XRA   A          ;A=0
2374   1CE6 77              MOV   M,A        ;CLEAR TCBP
2375   1CE7 2B              DCX   H          ;  IN
2376   1CE8 77              MOV   M,A        ;  RLSDB
2377   1CE9 210900          LXI   H,9        ;POINT H,L TO BYTE 9
2378   1CEC 19              DAD   D          ;  IN TCB
2379   1CED 7E              MOV   A,M        ;A= JINDU IN TCB
2380   1CEE B9              CMP   C          ;IF IT MATCHES
2381   1CEF C2201D          JNZ   TCPFE      ;  THE INT
2382                ;
2383                ;  ;JUNK EVENT TEC2, CREATE EVENT TECMG OR CMG
2384                ;
2385   1CF2 2B              DCX   H          ;POINT H,L TO BYTE 8 IN TCB
2386   1CF3 118025          LXI   D,JUNK     ;CHANGE
2387   1CF6 72              MOV   M,D        ;  EVENT
2388   1CF7 2B              DCX   H          ;  TEC2
2389   1CF8 73              MOV   M,E        ;  TO JUNK
2390                ;
2391   1CF9 C5              PUSH  B          ;SAVE JINDU
2392   1CFA CDA20F          CALL  GITRE      ;GET TREE PARAMETERS FOR INDUCT
2393   1CFD C1              POP   B          ;B,C=JINDU
2394   1CFE CDCB0F          CALL  GETFS      ;GET FREE BLOCK
2395   1D01 110900          LXI   D,9        ;POINT H,L TO BYTE 9
2396   1D04 19              DAD   D          ;  IN TCB
2397   1D05 71              MOV   M,C        ;SET JINDU IN TCB
2398   1D06 2B              DCX   H          ;POINT H,L TO BYTE 8 IN TCB
2399   1D07 11CA23          LXI   D,TECMG    ;POINT D,E TO TECMG
2400   1D0A 3A313D          LDA   SIDE0      ;IF
2401   1D0D E610            ANI   010H       ;  SIDE INDUCT
2402   1D0F CA151D          JZ    TSEVN      ;  THEN
2403   1D12 112924          LXI   D,CMG      ;  POINT D,E TO CMG
2404   1D15 72      TSEVN:  MOV   M,D        ;SET
2405   1D16 2B              DCX   H          ;  EVENT
2406   1D17 73              MOV   M,E        ;  IN TCB
2407   1D18 EB              XCHG             ;POINT D,E TO BYTE 7 IN TCB
2408   1D19 2A3142          LHLD  TET        ;SCHEDULE EVENT AT
2409   1D1C CD2F29          CALL  ENTCB      ;  CTIME+ TET TIME
2410   1D1F C9              RET
2411                ;
```

```
2412                    ;BAD TCB, OR NULL TCB
2413                    ;
2414   1D20 3E02        TCPFE:  MVI   A,2       ;FATAL ERROR
2415   1D22 CD0C2C             CALL  FATAL     ;  2
2416   1D25 C9                 RET
2417                    ;
2418                    ;TECPC WITHOUT PPBPC
2419                    ;
2420   1D26 3600        TCPE0:  MVI   M,0       ;ZERO PP IN PPDB
2421                    ;
2422                    ;TECPC WHILE RELEASE NOT RUNNING
2423                    ;
2424   1D28 3E01        TCPEX:  MVI   A,1       ;ERROR
2425   1D2A CD130F             CALL  ERRPC     ;  1
2426   1D2D C9                 RET
2427                    ;;
2428                    ;INDUCT-PHOTO-BLOCKING PROCESSOR.  FOR STRAIGHT INDUCT ONLY
2429                    ;  START TO TRACK PACKAGE.
2430                    ;
2431                    ;INPUT: C= INDUCT #.
2432                    ;ALL REGISTERS EFFECTED.
2433                    ;
2434   1D2E CDA227      IDBPC:  CALL  CPRLP     ;POINT HL TO
2435   1D31 23                 INX   H         ;  BYTE 2
2436   1D32 23                 INX   H         ;  IN RLSDB
2437   1D33 22EF42             SHLD  RLDBP     ;SAVE RLSDB POINTER
2438   1D36 5E                 MOV   E,M       ;POINT DE
2439   1D37 23                 INX   H         ;  TO MERGE
2440   1D38 56                 MOV   D,M       ;  FIFO
2441   1D39 7B                 MOV   A,E       ;IF POINTER
2442   1D3A B2                 ORA   D         ;  NOT
2443   1D3B C8                 RZ              ;  NULL
2444                    ;
2445   1D3C 210900             LXI   H,9       ;POINT H,L TO BYTE 9
2446   1D3F 19                 DAD   D         ;  IN TCB
2447   1D40 7E                 MOV   A,M       ;IF JINDU IN TCB
2448   1D41 B9                 CMP   C         ;  MATCHES
2449   1D42 C0                 RNZ             ;  THIS INTERRUPT
2450                    ;
2451                    ;THE INTERRUPT IS AS EXPECTED; JUNK EVENT IDB2
2452                    ;
2453   1D43 210E00             LXI   H,14      ;POINT H,L TO BYTE 14
2454   1D46 19                 DAD   D         ;  IN TCB
2455   1D47 E5                 PUSH  H         ;SAVE POINTER
2456   1D48 4E                 MOV   C,M       ;POINT B,C TO NEXT
2457   1D49 23                 INX   H         ;  PACKAGE
2458   1D4A 46                 MOV   B,M       ;  IN MERGE FIFO
2459   1D4B 2AEF42             LHLD  RLDBP     ;DELETE TCB
2460   1D4E 71                 MOV   M,C       ;  FROM
2461   1D4F 23                 INX   H         ;  INDUCT MERGE
2462   1D50 70                 MOV   M,B       ;  FIFO
2463   1D51 210700             LXI   H,7       ;POINT H,L TO BYTE 7
2464   1D54 19                 DAD   D         ;  IN TCB
2465   1D55 118025             LXI   D,JUNK    ;CHANGE
2466   1D58 73                 MOV   M,E       ;  EVENT IDB2
2467   1D59 23                 INX   H         ;  TO
2468   1D5A 72                 MOV   M,D       ;  JUNK
2469                    ;
2470                    ;CREATE EVENT UDB1
2471                    ;
2472   1D5B 3E01               MVI   A,1       ;GET TREE PARAMETERS
2473   1D5D CDBA0F             CALL  GTREP     ;  FOR PPI # 1
2474   1D60 C1                 POP   B         ;POINT B,C TO BYTE 13
2475   1D61 0B                 DCX   B         ;  IN OLD TCB
2476   1D62 CD892C             CALL  GUDB1     ;CREATE TCB FOR UDB1
2477   1D65 4F                 MOV   C,A       ;C=JINDU
2478   1D66 EB                 XCHG            ;POINT D,E TO BYTE 7 IN TCB
2479   1D67 2A3342             LHLD  IDT       ;POINT H,L TO IDT TABLE
2480   1D6A CD2F29             CALL  ENTCB     ;SCHEDULE EVENT UDB1
2481   1D6D C9                 RET
2482                    ;;
2483                    ;UPDATE-PHOTO-BLOCKING PROCESSOR.
2484                    ;
2485                    ;INPUT: C= UPDATE PHOTO NUMBER > 0.
2486                    ;OUTPUT: NEW SCHEDULED EVENTS, UDBT FOR SDVRS, LDVP FOR SCDVA,
2487                    ;  JDV FOR SCHDC.
2488                    ;ALL REGISTERS AND JUPDA EFFECTED.
2489                    ;
2490   1D6E 79          UDBPC:  MOV   A,C       ;SAVE UPDATE PHOTO
2491   1D6F 329542             STA   JUPDA     ;  NUMBER
2492   1D72 0600               MVI   B,0       ;B,C=JUPDA
2493   1D74 0B                 DCX   B         ;B,C=JUPDA-1
2494   1D75 2A5142             LHLD  UDDB      ;POINT
2495   1D78 09                 DAD   B         ;  H,L
2496   1D79 09                 DAD   B         ;  TO
2497   1D7A 09                 DAD   B         ;  PMUZ(JUPDA)
2498   1D7B 09                 DAD   B         ;  IN UDDB
2499   1D7C 03                 INX   B         ;B,C=JUPDA
```

```
2500  1D7D 5E            MOV   E,M       ;D,E
2501  1D7E 70            MOV   M,B       ;  = PMUZ,
2502  1D7F 23            INX   H         ;  AND NEW
2503  1D80 56            MOV   D,M       ;  PMUZ IN UDDB
2504  1D81 70            MOV   M,B       ;  = 0
2505  1D82 E5            PUSH  H         ;SAVE POINTER TO BYTE 1 IN UDDB
2506  1D83 7B            MOV   A,E       ;IF OLD
2507  1D84 B2            ORA   D         ;  PMUZ
2508  1D85 C2AA1D        JNZ   UBPN0     ;  NULL
2509                ;
2510                ;UNIDENTIFIED PACKAGE DETECTED AT UD
2511                ;
2512  1D88 CD080F        CALL  ERPC      ;ERROR 0 TO MASTER KB
2513  1D8B CDCB0F        CALL  GETFS     ;GET FREE BLOCK FOR TCB
2514  1D8E E5            PUSH  H         ;SAVE TCB POINTER
2515  1D8F 110D00        LXI   D,13      ;POINT H,L TO BYTE 13
2516  1D92 19            DAD   D         ;  IN TCB
2517  1D93 72            MOV   M,D       ;IDENT
2518  1D94 2B            DCX   H         ;  IN TCB
2519  1D95 72            MOV   M,D       ;  = 0
2520  1D96 2B            DCX   H         ;POINT H,L TO BYTE 11 IN TCB
2521  1D97 3A303D        LDA   ERCHU     ;JDV IN TCB
2522  1D9A 77            MOV   M,A       ;  =ERCHU
2523  1D9B 2B            DCX   H         ;POINT H,L TO BYTE 10 IN TCB
2524  1D9C 3A9542        LDA   JUPDA     ;A=JUPDA
2525  1D9F 77            MOV   M,A       ;SET JUPDA IN TCB
2526  1DA0 2B            DCX   H         ;JINDU IN TCB
2527  1DA1 3601          MVI   M,1       ;  = 1
2528  1DA3 2B            DCX   H         ;POINT H,L TO BYTE 8 IN TCB
2529  1DA4 72            MOV   M,D       ;EVENT IN TCB
2530  1DA5 2B            DCX   H         ;  = 0
2531  1DA6 72            MOV   M,D       ;  FOR UDCPC
2532  1DA7 C3541E        JMP   UBPUD
2533                ;
2534                ;INT IS AS EXPECTED; PACKAGE WITHIN UD ZONE
2535                ;
2536  1DAA 210A00   UBPN0:  LXI   H,10      ;POINT H,L TO BYTE 10
2537  1DAD 19            DAD   D         ;  IN TCB AT PMUZ
2538  1DAE 79            MOV   A,C       ;A=JUPDA
2539  1DAF BE            CMP   M         ;IF JUPDA MATCHES
2540  1DB0 C2A71F        JNZ   UBPFX     ;  JUPDA IN TCB
2541                ;
2542  1DB3 D5            PUSH  D                 ;SAVE TCB POINTER
2543  1DB4 2B            DCX   H                 ;POINT H,L TO BYTE 8
2544  1DB5 2B            DCX   H                 ;    IN TCB
2545                ;
2546                ; TEST TO SEE IF EVENT IS UDB2 OR DCB2
2547  1DB6 2B            DCX   H                 ;H,L= EVENT LSB
2548  1DB7 11D924        LXI   D,UDB2            ;D,E=UDB2
2549  1DBA 7B            MOV   A,E               ;A= LSB
2550  1DBB 96            SUB   M                 ;
2551  1DBC 47            MOV   B,A               ;B= LSB RESULT
2552  1DBD 23            INX   H                 ;
2553  1DBE 7A            MOV   A,D               ;A= MSB
2554  1DBF 96            SUB   M                 ;MSB SUBTRACT
2555  1DC0 90            SUB   B                 ;(UDB2-EVENT)
2556  1DC1 CA481E        JZ    UBPN1             ;JMP IF EVENT UDB2
2557                ;
2558                ; EVENT = DCB2. THEREFORE SEND DAS THE MESSAGE U(IF MASKED) AND SEND
2559                ;PACKAGE TO ERCHU.DON'T CLEAR FROM TRANSACTION TAB AS PACK STILL TRAKIN
2560  1DC4 3A2B42        LDA   ICM0              ;ICM0=1?
2561  1DC7 A7            ANA   A
2562  1DC8 CA391E        JZ    UBPN2             ;JMP IF NOT IN I/C MODE
2563  1DCB 212C3D        LXI   H,ERMK0+4         ;H,L= MASK BYTE 4
2564  1DCE 7E            MOV   A,M
2565  1DCF E610          ANI   10H               ;LOOK AT BIT 4
2566  1DD1 CA3E1E        JZ    UBPN3             ;JMP IF U NOT MASKED
2567                ; SEND MESSAGE U TO DAS
2568                ; LOAD PACKAGE CODE INTO BUFFER BYT16 FROM TRANSACTION TABLE IF IDENT
2569                ;NOT=0.IF IDENT=0,LOAD 6 ASCII 0'S INTO BYT16.DON'T CLEAR PACKAGE
2570                ; CODE FROM TRANSACTION TABLE AS PACKAGE IS STILL BEING TRACKED.
2571  1DD4 E1            POP   H                 ;H,L= TCB LOC
2572  1DD5 E5            PUSH  H                 ;STACK=SAME
2573  1DD6 110C00        LXI   D,12              ;
2574  1DD9 19            DAD   D                 ;H,L= IDENT LSB
2575  1DDA 5E            MOV   E,M               ;
2576  1DDB 23            INX   H                 ;H,L=IDENT MSB
2577  1DDC 56            MOV   D,M               ;D,E=IDENT=LOC OF PACK CODE IN TRANS TAB
2578  1DDD 7B            MOV   A,E               ;A= IDENT LSB
2579  1DDE B2            ORA   D                 ;OR WITH IDENT MSB
2580  1DDF CAE91D        JZ    UPACK             ;JMP IF IDENT = 0
2581  1DE2 EB            XCHG                    ;H,L= TRANS TAB LOC OF PACK CODE
2582  1DE3 CD8C1A        CALL  TRANP             ;LOAD PACK CODE INTO BYT16 BYTES 2-7
2583  1DE6 C3FD1D        JMP   PSTAK             ;JMP TO PSTAK
2584  1DE9 3E30     UPACK:  MVI   A,'0'         ;STORE ASCII 0 INTO BYT16 FOR PACK CODE
2585  1DEB 321A43        STA   BYT16+2           ;
2586  1DEE 321B43        STA   BYT16+3           ;
2587  1DF1 321C43        STA   BYT16+4           ;
```

```
2588   1DF4 321D43          STA   BYT16+5          ;
2589   1DF7 321E43          STA   BYT16+6          ;
2590   1DFA 321F43          STA   BYT16+7          ;
2591   1DFD E1       PSTAK: POP   H                ;H,L+TCB LOC
2592   1DFE E5              PUSH  H                ;STACK= SAME
2593                ; CONVERT BINARY JDV AND JINDU INTO ASCII AND STORE IN BYT16 BUFFER
2594                ;
2595   1DFF 110900          LXI   D,9
2596   1E02 19              DAD   D                ;H,L= JINDU LOC
2597   1E03 7E              MOV   A,M              ;A= JINDU BINARY VALUE
2598   1E04 CD010E          CALL  ECDIG            ;CONVERT TO ASCII
2599   1E07 3A7E42          LDA   XYZ1
2600   1E0A 322043          STA   BYT16+8          ;STORE ASCII 10'S
2601   1E0D 3A7F42          LDA   XYZ0
2602   1E10 322143          STA   BYT16+9          ;STORE ASCII 1'S
2603   1E13 E1              POP   H                ;H,L= TCB LOC
2604   1E14 E5              PUSH  H                ;STACK= SAME
2605   1E15 110B00          LXI   D,11
2606   1E18 19              DAD   D                ;H,L= JDV BINARY VALUE LOC
2607   1E19 7E              MOV   A,M              ;A= JDV BINARY VALUE
2608   1E1A CD010E          CALL  ECDIG            ;CONVERT TO ASCII
2609   1E1D 3A7D42          LDA   XYZ2
2610   1E20 322243          STA   BYT16+10         ;STORE ASCII 100'S
2611   1E23 3A7E42          LDA   XYZ1
2612   1E26 322343          STA   BYT16+11         ;STORE ASCII 10'S
2613   1E29 3A7F42          LDA   XYZ0
2614   1E2C 322443          STA   BYT16+12         ;STORE ASCII 1'S
2615   1E2F 211843          LXI   H,BYT16          ;H,L= BYT16 LOC
2616   1E32 23              INX   H
2617   1E33 3E55            MVI   A,'U'            ;MAKE BYTE 1 = ASCII U
2618   1E35 77              MOV   M,A
2619   1E36 CD7D1A          CALL  DSLD             ;LOAD BYT16 INTO FS
2620                ;
2621                ; PUT ERROR 5 ON KB
2622   1E39 3E05    UBPN2:  MVI   A,5              ;A=5
2623   1E3B CD080F          CALL  ERPC             ;DISPLAY ON KB
2624                ;
2625                ; SEND PACKAGE TO ERCHU
2626   1E3E E1      UBPN3:  POP   H                ;H,L= TCB LOC
2627   1E3F E5              PUSH  H                ;STACK= SAME
2628   1E40 110B00          LXI   D,11
2629   1E43 19              DAD   D                ;H,L= JDV LOC IN TCB
2630   1E44 3A303D          LDA   ERCHU            ;A= ERCHU
2631   1E47 77              MOV   M,A              ;SET JDV=ERCHU
2632                ;
2633                ; CHANGE EVENT DCB2 TO UDB3
2634   1E48 E1      UBPN1:  POP   H                ;H,L= TCB LOC
2635   1E49 E5              PUSH  H                ;STACK= SAME
2636   1E4A 110800          LXI   D,8
2637   1E4D 19              DAD   D                ;H,L= EVENT MSB
2638   1E4E 119925          LXI   D,UDB3           ;D,E=UDB3
2639   1E51 72              MOV   M,D
2640   1E52 2B              DCX   H
2641   1E53 73              MOV   M,E              ;D,E= UDB3= EVENT IN TCB
2642                ;
2643   1E54 D1      UBPUD:  POP   D          ;POINT D,E TO TCB
2644   1E55 E1              POP   H          ;POINT H,L TO BYTE 2
2645   1E56 23              INX   H          ;  IN  UDDB
2646   1E57 7E              MOV   A,M        ;IF
2647   1E58 23              INX   H          ;  PUD
2648   1E59 B6              ORA   M          ;  IN UDDB
2649   1E5A CA681E          JZ    UPPUD      ;  =0, THEN FINE
2650                ;
2651                ;  ;FAKE AN UD-CLEAR INTERRUPT IF IT WAS LOST
2652                ;
2653   1E5D E5              PUSH  H          ;SAVE
2654   1E5E D5              PUSH  D          ;  POINTERS
2655   1E5F 3A9542          LDA   JUPDA      ;PICK UP UPDATE
2656   1E62 4F              MOV   C,A        ;  PHOTO #
2657   1E63 CD3120          CALL  UDCPC      ;FAKE UD-CLEAR INTERRUPT
2658   1E66 D1              POP   D          ;RESTORE
2659   1E67 E1              POP   H          ;  POINTERS
2660   1E68 72      UPPUD:  MOV   M,D        ;POINT PUD
2661   1E69 2B              DCX   H          ;  IN UDDB
2662   1E6A 73              MOV   M,E        ;  TO TCB
2663   1E6B 210B00          LXI   H,11       ;POINT H,L TO BYTE 11
2664   1E6E 19              DAD   D          ;  IN TCB
2665   1E6F E5              PUSH  H          ;SAVE POINTER
2666   1E70 3A9542          LDA   JUPDA      ;A=JUPDA
2667   1E73 CDAD0F          CALL  GUTR1      ;GET TREE PARAMETERS FOR UD # JUPDA
2668                ;
2669                ;SCHEDULE BMG FOR SIDE INDUCT
2670                ;
2671   1E76 3A313D          LDA   SIDEO      ;IF SIDE INDUCT
2672   1E79 E610            ANI   010H       ;  THEN
2673   1E7B C4AE1F          CNZ   SCBMG      ;  SCHEDULE BMG
2674                ;
2675                ;CONTINUE TO TRACK PACKAGE OR SCHEDULE DIVERTS
```

```
2676                    ;
2677   1E7E 2AB642              LHLD  CTIME      ;SET UDBT FOR
2678   1E81 229942              SHLD  UDBT       ;  SDVRS
2679   1E84 3A9542              LDA   JUPDA      ;A
2680   1E87 D602                SUI   2          ;  = UPDATE PHOTO # - 2
2681   1E89 2A3D42              LHLD  LDVDB      ;POINT H,L TO LDV(1)
2682   1E8C 85                  ADD   L          ;POINT H,L
2683   1E8D 6F                  MOV   L,A        ;  TO LDV(JUPDA-1)
2684   1E8E 22F742              SHLD  LDVP       ;SET LDVP FOR SCDVA, UBNXU
2685   1E91 C1                  POP   B          ;POINT B,C TO BYTE 11 IN TCB
2686   1E92 0A                  LDAX  B          ;A=JDV
2687   1E93 329642              STA   JDV        ;SAVE JDV FOR SCHDC AND SCDVA
2688   1E96 03                  INX   B          ;POINT B,C TO BYTE 13
2689   1E97 03                  INX   B          ;  IN TCB
2690   1E98 B7                  ORA   A          ;JDV=0 MEANS TO
2691   1E99 CA4A1F              JZ    UBNXU      ;  RECIRCULATE
2692   1E9C 3D                  DCR   A          ;A=JDV-1
2693   1E9D BE                  CMP   M          ;IF JDV-1 >= LDV(JUPDA-1),
2694   1E9E DA4A1F              JC    UBNXU      ;  I.E., IF JDV > LDV(JUPDA-1)
2695   1EA1 23                  INX   H          ;POINT H,L TO LDV(JUPDA)
2696   1EA2 BE                  CMP   M          ;IF JDV-1 < LDV,
2697   1EA3 D24A1F              JNC   UBNXU      ;  I.E., IF JDV <= LDV
2698                   ;
2699                   ;  ;PACKAGE IS TO BE DIVERTED BEFORE NEXT UD OR END OF CONVEYOR
2700                   ;
2701   1EA6 C5                  PUSH  B                ;STACK = BYTE 13 OF PACKAGE TCB
2702   1EA7 3A2B42              LDA   ICMO             ;
2703   1EAA D601                SUI   1                ;
2704   1EAC CAB41E              JZ    CKDLF            ;JMP IF I/C MODE
2705   1EAF CDD41F    CSCDV:    CALL  SCDVA            ;SCHEDULE DIVERT
2706   1EB2 C1                  POP   B                ;ADJUST STACK
2707   1EB3 C9                  RET
2708                   ;CHECK TO SEE IF DIVERT LANE IS FULL
2709                   ;
2710   1EB4 3A9642    CKDLF:    LDA   JDV              ;A= DVT#
2711   1EB7 47                  MOV   B,A              ;B= DVT#
2712   1EB8 CD250D              CALL  CPLFM            ;CHECK FOR LANE FULL
2713   1EBB A6                  ANA   M                ;IF DVT LANE FULL, CLEAR ZERO FLAG
2714   1EBC C2D11E              JNZ   CKAMK            ;JMP IF LANE FULL
2715                   ;DIVERT LANE NOT FULL. TEST FOR DIVERT COMPLETE OPTION
2716                   ;
2717   1EBF C1                  POP   B                ;B,C= BYTE 13 LOC OF TCB
2718   1EC0 C5                  PUSH  B                ;STACK = SAME
2719   1EC1 3A3B3D              LDA   DCO              ;DCO= 2?
2720   1EC4 D602                SUI   2                ;
2721   1EC6 C2AF1E              JNZ   CSCDV            ;JMP IF DCO NOT= 2
2722   1EC9 CD1E20              CALL  SCHDC            ;SCHEDULE DCB1 IN ORDER TO SEND MESSAGE
2723                                                   ;  S OR U
2724   1ECC C1                  POP   B                ;B,C= BYTE 13 LOC OF TCB
2725   1ECD C5                  PUSH  B                ;STACK = SAME
2726   1ECE C3AF1E              JMP   CSCDV            ;JMP TO SCHEDULE DIVERT
2727                   ;DIVERT LANE IS FULL. SEND MESSAGE A AND SEND PACKAGE TO LFEDV
2728                   ;IF MESSAGE A IS MASKED.
2729   1ED1 212A3D    CKAMK:    LXI   H,ERMKO+2        ;H,L= ERMKO BYTE 2
2730   1ED4 7E                  MOV   A,M              ;A= BYTE 2
2731   1ED5 E601                ANI   01H              ;LOOK AT BIT 0
2732   1ED7 CA3F1F              JZ    SLFDV            ;JMP IF A NOT MASKED
2733   1EDA C1                  POP   B                ;B,C= BYTE 13 OF PACKAGE TCB
2734   1EDB C5                  PUSH B                 ;STACK=SAME
2735   1EDC 69                  MOV   L,C              ;
2736   1EDD 60                  MOV   H,B              ;H,L= BYTE 13 LOC OF TCB
2737   1EDE 56                  MOV   D,M              ;
2738   1EDF 2B                  DCX   H                ;H,L= BYTE 12 LOC
2739   1EE0 5E                  MOV   E,M              ;D,E= IDENT VALUE
2740                   ;ACCESS PACKAGE CODE FROM TRANSACTION TABLE AND LOAD INTO BYTES 2-7 OF
2741                   ;BUFFER BYT16 IF IDENT NOT=0.IF IDENT=0,LOAD 6 ASCII 0'S INTO BYT16 FOR
2742                   ;PACK CODE. IN EITHER CASE, DON'T REMOVE FROM TRANSACTION TABLE AS
2743                   ;PACKAGE IS STILL TRACKING.
2744   1EE1 7B                  MOV   A,E              ;
2745   1EE2 B2                  ORA   D                ;
2746   1EE3 CAEF1E              JZ    APACK            ;JMP IF IDENT=0
2747   1EE6 EB                  XCHG                   ;H,L= IDENT= LOC OF PACK CODE
2748   1EE7 E5                  PUSH  H                ;STACK= SAME
2749   1EE8 CD6C1A              CALL  TRANP            ;TRANSFER PACK CODE TO BYT16 BYTES 2-7
2750   1EEB E1                  POP   H                ;ADJUST STACK
2751   1EEC C3031F              JMP   LOADH            ;JMP TO LOADH
2752   1EEF 3E30      APACK:    MVI   A,30H            ;STORE 6 ASCII 0'S AS PACK CODE
2753   1EF1 321A43              STA   BYT16+2          ;
2754   1EF4 321B43              STA   BYT16+3          ;
2755   1EF7 321C43              STA   BYT16+4          ;
2756   1EFA 321D43              STA   BYT16+5          ;
2757   1EFD 321E43              STA   BYT16+6          ;
2758   1F00 321F43              STA   BYT16+7          ;
2759                   ;CONVERT BINARY INDUCT AND DVT# IN PACKAGE TCB INTO ASCII AND STORE IN
2760                   ;BUFFER BYT16.
2761                   ;
2762   1F03 E1        LOADH:    POP   H                ;H,L= BYTE 13 LOC IN PACK TCB
```

```
2763   1F04 E5              PUSH  H              ;STACK= SAME
2764   1F05 11FCFF          LXI   D,-4           ;
2765   1F08 19              DAD   D              ;H,L= JINDU LOC
2766   1F09 7E              MOV   A,M            ;A= JINDU IN BINARY
2767   1F0A CD010E          CALL  ECDIG          ;CONVERT TO ASCII
2768   1F0D 3A7E42          LDA   XYZ1           ;
2769   1F10 322043          STA   BYT16+8        ;WSTORE ASCII 10'S DIGIT
2770   1F13 3A7F42          LDA   XYZ0           ;
2771   1F16 322143          STA   BYT16+9        ;STORE ASCII 1'S DIGIT
2772   1F19 E1              POP   H              ;H,L= TCB BYTE 13 LOC
2773   1F1A E5              PUSH  H              ;
2774   1F1B 11FEFF          LXI   D,-2           ;H,L= JDV LOC
2775   1F1E 19              DAD   D              ;
2776   1F1F 7E              MOV   A,M            ;A= JDV IN BINARY
2777   1F20 CD010E          CALL  ECDIG          ;CONVERT TO ASCII
2778   1F23 3A7D42          LDA   XYZ2           ;
2779   1F26 322243          STA   BYT16+10       ;STORE 100'S DIGIT
2780   1F29 3A7E42          LDA   XYZ1           ;
2781   1F2C 322343          STA   BYT16+11       ;STORE 10'S DIGIT
2782   1F2F 3A7F42          LDA   XYZ0           ;
2783   1F32 322443          STA   BYT16+12       ;STORE 1'S DIGIT
2784   1F35 211843          LXI   H,BYT16        ;
2785   1F38 23              INX   H              ;H,L= BYT16 BYTE 1
2786   1F39 3E41            MVI   A,'A'          ;
2787   1F3B 77              MOV   M,A            ;STORE ASCII A
2788   1F3C CD7D1A          CALL  DSLD           ;STORE MESSAGE A IN FS
2789                ;
2790                ;SEND PACKAGE TO LFEDV
2791                ;
2792   1F3F C1      SLFDV:  POP   B              ;B,C= BYTE 13 OF TCB
2793   1F40 69              MOV   L,C            ;H,L= BYTE 13 LOC OF TCB
2794   1F41 60              MOV   H,B            ;
2795   1F42 11FEFF          LXI   D,-2           ;
2796   1F45 19              DAD   D              ;H,L= JDV LOC (BYTE 11)
2797   1F46 3A3D3D          LDA   LFEDV          ;
2798   1F49 77              MOV   M,A            ;JDV IN TCB NOW=LFEDV
2799                ;
2800                ;  ;TRANSFER PACKAGE TO NEXT UD
2801                ;
2802   1F4A C5      UBNXU:  PUSH  B          ;SAVE POINTER TO BYTE 13 IN TCB
2803   1F4B 2AF742          LHLD  LDVP       ;POINT H,L TO LDV(JUPDA-1)
2804   1F4E 4E              MOV   C,M        ;C= LDV(JUPDA-1)
2805   1F4F 0C              INR   C          ;C= 1ST DV # FOR THIS UD
2806   1F50 23              INX   H          ;A
2807   1F51 7E              MOV   A,M        ;  = LDV(JUPDA)
2808   1F52 CD7727          CALL  SDVRS      ;SCHEDULE RESETS OF ALL DV'S FOR THIS UD
2809   1F55 C1              POP   B          ;POINT B,C TO BYTE 13 IN TCB
2810                ;
2811                ;  ;  ;SCHEDULE UDB1 IF NECESSARY
2812                ;
2813   1F56 3A9542          LDA   JUPDA      ;IF
2814   1F59 21043D          LXI   H,NUPDA    ;  JUPDA
2815   1F5C BE              CMP   M          ;  <
2816   1F5D D26C1F          JNC   USUD1      ;  NUPDA
2817   1F60 CD892C          CALL  GUDB1      ;CREATE TCB FOR UDB1 FOR NEXT UD
2818   1F63 4B              MOV   C,E        ;C= OLD JUPDA
2819   1F64 EB              XCHG             ;POINT D,E TO BYTE 7 IN TCB
2820   1F65 2A3742          LHLD  UDT        ;POINT H,L TO UDT TABLE
2821   1F68 CD2F29          CALL  ENTCB      ;SCHEDULE UDB1 FOR NEXT UD
2822   1F6B C9              RET
2823                ;
2824                ; PACKAGE HAS JUST PASSED LAST UPDATE PHOTO. IF IN I/C MODE. AND IDENT
2825                ; NOT=0, REMOVE PACK CODE FROM TRANSACTION TABLE.
2826                ;
2827   1F6C 3A2B42  USUD1:  LDA   ICMO           ;TEST TO SEE IF ARE IN I/C MODE
2828   1F6F D600            SUI   0              ;
2829   1F71 CA8F1F          JZ    USUDP          ;JMP IF NOT IN I/C MODE
2830                ;
2831                ; ARE IN I/C MODE. TEST TO SEE IF PACK CODE IS IN TRANS TABLE
2832   1F74 60              MOV   H,B            ;
2833   1F75 69              MOV   L,C            ;H,L= TCB BYTE 13 LOC
2834   1F76 7E              MOV   A,M            ;A= IDENT MSB
2835   1F77 D600            SUI   0
2836   1F79 C2861F          JNZ   USUDJ          ;JMP IF IDENT MSB NOT=0
2837   1F7C 2B              DCX   H              ;H,L= IDENT LSB LOC
2838   1F7D 7E              MOV   A,M
2839   1F7E D600            SUI   0
2840   1F80 C2861F          JNZ   USUDJ          ;JMP IF IDENT LSB NOT = 0
2841   1F83 C38F1F          JMP   USUDP          ;JMP IF IDENT = 0
2842                ;
2843                ; PACKAGE HAS JUST PASSED LAST UPDATE PHOTO WITH IDENT NOT=0. THEREFORE
2844                ; REMOVE PACKAGE CODE FROM TRANSACTION TABLE
2845   1F86 60      USUDJ:  MOV   H,B            ;
2846   1F87 69              MOV   L,C        ;H,L=IDENT MSB LOC
2847   1F88 56              MOV   D,M
2848   1F89 2B              DCX   H
2849   1F8A 5E              MOV   E,M        ;D,E=IDENT
```

```
2850   1F8B EB              XCHG                    ;H,L=SAME
2851   1F8C CD401A          CALL  ZPACK             ;REMOVE PACK CODE FROM TRANS TAB
2852                       ;
2853   1F8F 3A313D  USUDP:  LDA   SIDEO     ;IF SIDE-INDUCT RE-CIRCULATION
2854   1F92 E602            ANI   02H       ;  AUTO-REINDUCT
2855   1F94 C8              RZ              ;  DESIRED
2856   1F95 CD892C          CALL  GUDB1     ;CREATE TCB FOR UDB1, E=JUPDA
2857   1F98 4B              MOV   C,E       ;C=JUPDA, SHOULD = NUPDA
2858   1F99 EB              XCHG            ;POINT D,E TO BYTE 7 IN TCB
2859   1F9A 210300          LXI   H,3       ;POINT H,L TO BYTE 10
2860   1F9D 19              DAD   D         ;  IN TCB
2861   1F9E 3601            MVI   M,1       ;JUPDA IN TCB = 1
2862   1FA0 2A3742          LHLD  UDT       ;POINT H,L TO UDT TABLE
2863   1FA3 CD2F29          CALL  ENTCB     ;SCHEDULE UDB1 FOR UD # 1
2864   1FA6 C9              RET
2865                       ;
2866                       ;OLD TCB FOR UDB2 FOUND TO BE SCRIBBLED; OR
2867                       ;  UPDATE DATA SCRIBBLED (MAY BE DUE TO LOST UD-CLEARING INTERRUPT)
2868                       ;
2869   1FA7 3E0A    UBPFX:  MVI   A,10      ;FATAL ERROR
2870   1FA9 CD0C2C          CALL  FATAL     ;  10
2871   1FAC E1              POP   H         ;RESET SP
2872   1FAD C9              RET
2873                       ;;
2874                       ;CALLED BY UDBPC TO SCHEDULE EVENT BMG FOR SIDE INDUCT.
2875                       ;
2876                       ;INPUT: JUPDA= UPDATE #, TREE PARAMETERS.
2877                       ;OUTPUT: EVENT BMG IN TREE.
2878                       ;ALL REGISTERS EFFECTED.
2879                       ;
2880   1FAE 3A9542  SCBMG:  LDA   JUPDA     ;PICK UP
2881   1FB1 4F              MOV   C,A       ;  UPDATE
2882   1FB2 0600            MVI   B,0       ;  PHOTO #
2883   1FB4 2A3F42          LHLD  FIDDB     ;GET NUMBER OF INDUCT
2884   1FB7 09              DAD   B         ;  FOLLOWING
2885   1FB8 2B              DCX   H         ;  IF
2886   1FB9 4E              MOV   C,M       ;  ANY
2887   1FBA 0D              DCR   C         ;DO NOTHING IF
2888   1FBB 0C              INR   C         ;  NO INDUCT
2889   1FBC C8              RZ              ;  FOLLOWS
2890                       ;
2891   1FBD CDCB0F          CALL  GETFS     ;GET FREE BLOCK
2892   1FC0 110900          LXI   D,9       ;POINT H,L TO BYTE 9
2893   1FC3 19              DAD   D         ;  IN TCB
2894   1FC4 71              MOV   M,C       ;SET INDUCT # IN TCB
2895   1FC5 2B              DCX   H         ;POINT H,L TO BYTE 8 IN TCB
2896   1FC6 111A24          LXI   D,BMG     ;POINT EVENT
2897   1FC9 72              MOV   M,D       ;  IN TCB
2898   1FCA 2B              DCX   H         ;  TO
2899   1FCB 73              MOV   M,E       ;  BMG
2900   1FCC EB              XCHG            ;POINT D,E TO BYTE 7 IN TCB
2901   1FCD 2A2F42          LHLD  PRET      ;POINT H,L TO PRET TABLE
2902   1FD0 CD2F29          CALL  ENTCB     ;SCHEDULE EVENT BMG
2903   1FD3 C9              RET
2904                       ;;
2905                       ;SCHEDULE DIVERT ACTUATION EVENTS AFTER
2906                       ;  PACKAGE HAS HIT ITS LAST UPDATE PHOTO.
2907                       ;  CALLED BY UDBPC.
2908                       ;
2909                       ;INPUT: B,C POINTS TO BYTE 13 IN OLD TCB, TREE PARAMETERS,
2910                       ;  UDBT, JDV, LDVP POINTS TO LDV(JUPDA-1).
2911                       ;ALL REGISTERS EFFECTED.
2912                       ;
2913   1FD4 2AF742  SCDVA:  LHLD  LDVP      ;POINT H,L TO LDV(JUPDA-1)
2914   1FD7 C5              PUSH  B         ;STACK = BYTE 13 OF OLD TCB
2915   1FD8 4E              MOV   C,M       ;C= LDV(JUPDA-1)
2916   1FD9 0C              INR   C         ;C= 1ST DV FOR THIS UD
2917   1FDA 3A9642          LDA   JDV       ;A
2918   1FDD 3D              DCR   A         ;  =JDV-1
2919   1FDE CD7727          CALL  SDVRS     ;SCHEDULE RESETS OF DV'S BEFORE # JDV
2920                       ;
2921                       ;SCHEDULE ACTUATION OF DV # JDV
2922                       ;
2923   1FE1 CDCB0F          CALL  GETFS     ;GET FREE STORAGE
2924   1FE4 D1              POP   D         ;D,E= BYTE 13 OF OLD TCB LOC
2925   1FE5 E5              PUSH  H         ;STACK= LOC OF NEW TCB
2926   1FE6 EB              XCHG            ;H,L= BYTE 13 OF OLD TCB LOC
2927   1FE7 7E              MOV   A,M       ;A= IDENT MSB
2928   1FE8 329842          STA   IDENT+1   ;STORE IN WORKING MEMORY
2929   1FEB 2B              DCX   H         ;H,L= BYTE 12 OF OLD TCB LOC
2930   1FEC 7E              MOV   A,M       ;A= IDENT LSB
2931   1FED 329742          STA   IDENT     ;STORE IN WORKING MEMORY
2932   1FF0 2B              DCX   H
2933   1FF1 2B              DCX   H
2934   1FF2 2B              DCX   H         ;H,L= BYTE 9 OF OLD TCB
2935   1FF3 7E              MOV   A,M       ;A= JINDU
2936   1FF4 329442          STA   JINDU     ;STORE IN WORKING MEMORY
```

```
2937   1FF7 E1              POP   H          ;H,L= LOC OF NEW TCB
2938   1FF8 110D00          LXI   D,13       ;D,E= 13
2939   1FFB 19              DAD   D          ;H,L= LOC OF BYTE 13 IN NEW TCB
2940   1FFC 3A9842          LDA   IDENT+1    ;A= IDENT MSB
2941   1FFF 77              MOV   M,A        ;SET IN NEW TCB
2942   2000 3A9742          LDA   IDENT      ;A= IDENT LSB
2943   2003 2B              DCX   H          ;H,L= BYTE12 LOC IN NEW TCB
2944   2004 77              MOV   M,A        ;SET IN NEW TCB
2945   2005 2B              DCX   H          ;H,L= BYTE 11 LOC IN NEW TCB
2946   2006 3A9442          LDA   JINDU      ;A= JINDU
2947   2009 77              MOV   M,A        ;SET IN JDV LOC FOR DVACT TCB
2948   200A 2B              DCX   H          ;H,L= BYTE 10 LOC
2949   200B 3601            MVI   M,1        ;SET DIVERT FLAG = 1 FOR SETTING DV
2950   200D 2B              DCX   H          ;SET JDV
2951   200E 3A9642          LDA   JDV        ;  IN
2952   2011 77              MOV   M,A        ;  TCB
2953   2012 2B              DCX   H          ;POINT H,L TO BYTE 8 IN TCB
2954   2013 11A526          LXI   D,DVACT    ;LET
2955   2016 72              MOV   M,D        ;  EVENT
2956   2017 2B              DCX   H          ;  BE
2957   2018 73              MOV   M,E        ;  DVACT
2958   2019 4F              MOV   C,A        ;C=JDV
2959   201A CD1D29          CALL  ENDVA      ;SCHEDULE DVACT
2960   201D C9              RET
2961                ;
2962                ; SUBROUTINE SCHDC
2963                ;
2964                ;         INPUTS: B,C POINTS TO BYTE 13 OF PACKAGE TCB,TREE PARAMETERS,
2965                ;                 UDBT,JDV,LDVP POINTS TO LDV(JUPDA-1)(INPUTS SAME AS
2966                ;                 FOR SCDVA).
2967                ;
2968                ;         OUTPUTS: EVENT DCB1 CREATED AND SCHEDULED. DCB1 IS LOWER BOUND
2969                ;                 ON MUZ FOR NEXT UPDATE PHOTO
2970                ;
2971                ;         PROCESS: CREATE EVENT DCB1 BY CALLING GUDB1 AND THEN CHANGING
2972                ;                 EVENT TO DCB1. SCHEDULE DCB1 BY CALLING ENTCB
2973                ;
2974   201E CD892C  SCHDC:   CALL  GUDB1          ;CREATE TCB FOR EVENT UDB1. RETURN WITH
2975                                              ;H,L AT BYTE 7 OF NEW TCB,E=OLD JUPDA
2976   2021 01A525          LXI   B,DCB1        ;B,C= DCB1 EVENT LOC
2977   2024 23              INX   H             ;H,L= BYTE 8 OF NEW TCB
2978   2025 70              MOV   M,B           ;MAKE EVENT IN NEW TCB= DCB1
2979   2026 2B              DCX   H             ;H,L= BYTE 7 OF NEW TCB
2980   2027 71              MOV   M,C           ;EVENT IN NEW TCB NOW = DCB1
2981   2028 4B              MOV   C,E           ;C= OLD JUPDA
2982   2029 EB              XCHG                ;POINT D,E TO BYTE 7 OF NEW TCB=EVENT
2983   202A 2A3742          LHLD  UDT           ;POINT H,L TO UDT TABLE
2984   202D CD2F29          CALL  ENTCB         ;SCHEDULE DCB1 FOR NEXT UD
2985   2030 C9              RET                 ;RET TO UDBPC
2986                ;
2987                ;UPDATE-PHOTO-CLEARING PROCESSOR.
2988                ;
2989                ;INPUT: C=JUPDA (P
2990                ;UPDATE-PHOTO-CLEARING PROCESSOR.
2991                ;
2992                ;INPUT: C=JUPDA (PHOTO #).
2993                ;ALL REGISTERS EFFECTED.
2994                ;
2995   2031 0600    UDCPC:   MVI   B,0          ;B,C=JUPDA
2996   2033 2A5142           LHLD  UDDB         ;POINT H,L TO UDDB TABLE
2997   2036 09               DAD   B            ;POINT
2998   2037 09               DAD   B            ;  H,L TO
2999   2038 09               DAD   B            ;  PMUZ IN UDDB
3000   2039 09               DAD   B            ;  FOR UD # JUPDA+1
3001   203A 2B               DCX   H            ;POINT H,L TO PUD MS BYTE IN UDDB
3002   203B 56               MOV   D,M          ;SET
3003   203C 70               MOV   M,B          ;  D,E
3004   203D 2B               DCX   H            ;  = PUD IN UDDB
3005   203E 5E               MOV   E,M          ;  POINTING TO TCB
3006   203F 70               MOV   M,B          ;  & ZERO PUD IN UDDB
3007   2040 7B               MOV   A,E          ;IF
3008   2041 B2               ORA   D            ;  TCB
3009   2042 CAA420           JZ    UCPCE        ;  NONNULL
3010                ;
3011                ;D,E POINTS TO NONNULL PACKAGE TCB
3012                ;
3013   2045 D5               PUSH  D            ;SAVE TCB POINTER
3014   2046 C5               PUSH  B            ;SAVE JUPDA
3015   2047 CDAC0F           CALL  GUTRE        ;GET TREE PARAMETERS
3016   204A C1               POP   B            ;B,C=JUPDA
3017   204B C5               PUSH  B            ;SAVE UPDATE PHOTO #
3018   204C 3A313D           LDA   SIDEO        ;IF SIDE INDUCT
3019   204F E610             ANI   010H         ;  THEN
3020   2051 C4AA20           CNZ   SCCMG        ;  SCHEDULE CMG
3021                ;
3022   2054 C1               POP   B            ;RESTORE BC = UPDATE PHOTO #
3023   2055 2A3D42           LHLD  LDVDB        ;POINT H,L TO LDV TABLE
3024   2058 09               DAD   B            ;POINT H,L TO LDV
```

```
3025  2059 2B            DCX   H         ;  FOR THIS UD
3026  205A 7E            MOV   A,M       ;A= LDV FOR THIS UD
3027  205B D1            POP   D         ;POINT D,E TO TCB
3028  205C 210B00        LXI   H,11      ;POINT H,L TO BYTE 11
3029  205F 19            DAD   D         ;  IN TCB
3030  2060 BE            CMP   M         ;IF LDV FOR THIS UD
3031  2061 DA8E20        JC    UCUSE     ;  >= JDV
3032                ;
3033                ;SCHEDULE PROGRAM-MEASURED DIVERT RESET IF DESIRED
3034                ;
3035  2064 3A383D        LDA   DVTRS     ;IF
3036  2067 3D            DCR   A         ;  MEASURED
3037  2068 C28E20        JNZ   UCUSE     ;  RESET
3038                ;
3039  206B D5            PUSH  D         ;SAVE TCB POINTER
3040  206C 4E            MOV   C,M       ;C=JDV
3041  206D 2AB642        LHLD  CTIME     ;D,E
3042  2070 EB            XCHG            ;  = CTIME
3043  2071 2A0840        LHLD  DVH       ;ADD CONSTANT
3044  2074 19            DAD   D         ;  DVH
3045  2075 229942        SHLD  UDBT      ;SET UDBT FOR ENDVA
3046  2078 CDCB0F        CALL  GETFS     ;GET FREE BLOCK
3047  207B 110A00        LXI   D,10      ;POINT H,L TO BYTE 10
3048  207E 19            DAD   D         ;  IN TCB
3049  207F 3602          MVI   M,2       ;SET DV MODE = 2 FOR RESETTING DV
3050  2081 2B            DCX   H         ;SET DIVERT $
3051  2082 71            MOV   M,C       ;  IN TCB
3052  2083 2B            DCX   H         ;POINT H,L TO BYTE 8 IN TCB
3053  2084 11A526        LXI   D,DVACT   ;LET
3054  2087 72            MOV   M,D       ;  EVENT
3055  2088 2B            DCX   H         ;  BE
3056  2089 73            MOV   M,E       ;  DVACT
3057  208A CD1D29        CALL  ENDVA     ;SCHEDULE DV RESET
3058  208D D1            POP   D         ;POINT D,E TO TCB
3059                ;
3060                ;TCB HAS TO BE RETURNED TO FREE STORAGE AFTER HAVING BEEN USED
3061                ;  BY BOTH UDB3 AND UDCPC, OR UDCPC AND JUNK.
3062                ;
3063  208E 210700   UCUSE:  LXI   H,7       ;POINT H,L TO
3064  2091 19            DAD   D         ;  BYTE 7 IN TCB
3065  2092 7E            MOV   A,M       ;IF TCB IS FOR UNIDENTIFIED PACKAGE
3066  2093 23            INX   H         ;  FROM UDBPC, OR HAS
3067  2094 B6            ORA   M         ;  NOT YET BEEN USED
3068  2095 CA9F20        JZ    UCRFS     ;  BY UDB3
3069                ;
3070  2098 118025        LXI   D,JUNK    ;CHANGE
3071  209B 72            MOV   M,D       ;  EVENT
3072  209C 2B            DCX   H         ;  TO
3073  209D 73            MOV   M,E       ;  JUNK
3074  209E C9            RET
3075                ;
3076  209F EB       UCRFS:  XCHG            ;H,L POINTS TO TCB
3077  20A0 CD4F11        CALL  PUTFS     ;RETURN TO FREE STORAGE
3078  20A3 C9            RET
3079                ;
3080                ;SPURIOUS INTERRUPT
3081                ;
3082  20A4 3E0D     UCPCE:  MVI   A,13      ;ERROR 13
3083  20A6 CD080F        CALL  ERPC      ;  TO MASTER KB
3084  20A9 C9            RET
3085                ;
3086                ;SCHEDULE EVENT CMG FOR SIDE INDUCT.
3087                ;  CALLED BY UDCPC.
3088                ;
3089                ;INPUT: B,C= UPDATE $, TREE PARAMETERS.
3090                ;OUTPUT: EVENT CMG IN TREE.
3091                ;ALL REGISTERS EFFECTED.
3092                ;
3093  20AA 2A3F42   SCCMG:  LHLD  FIDDB     ;GET NUMBER OF INDUCT
3094  20AD 09            DAD   B         ;  FOLLOWING
3095  20AE 2B            DCX   H         ;  IF
3096  20AF 4E            MOV   C,M       ;  ANY
3097  20B0 0D            DCR   C         ;DO NOTHING IF
3098  20B1 0C            INR   C         ;  NO INDUCT
3099  20B2 C8            RZ              ;  FOLLOWS
3100                ;
3101  20B3 CDCB0F        CALL  GETFS     ;GET FREE BLOCK
3102  20B6 110900        LXI   D,9       ;POINT H,L TO BYTE 9
3103  20B9 19            DAD   D         ;  IN TCB
3104  20BA 71            MOV   M,C       ;SET JINDU IN TCB
3105  20BB 2B            DCX   H         ;POINT H,L TO BYTE 8 IN TCB
3106  20BC 112924        LXI   D,CMG     ;POINT EVENT
3107  20BF 72            MOV   M,D       ;  IN TCB
3108  20C0 2B            DCX   H         ;  TO
3109  20C1 73            MOV   M,E       ;  CMG
3110  20C2 E5            PUSH  H         ;SAVE POINTER TO BYTE 7 IN TCB
3111  20C3 0B            DCX   B         ;B,C=JINDU-1
```

```
3112   20C4 2A2F42            LHLD  PRET       ;POINT H,L TO
3113   20C7 09                DAD   B          ;  PRET
3114   20C8 09                DAD   B          ;  TIME
3115   20C9 5E                MOV   E,M        ;D,E
3116   20CA 23                INX   H          ;  =
3117   20CB 56                MOV   D,M        ;  PRET TIME
3118   20CC 2A3142            LHLD  TET        ;POINT H,L TO
3119   20CF 09                DAD   B          ;  TET
3120   20D0 09                DAD   B          ;  TIME
3121   20D1 4E                MOV   C,M        ;B,C
3122   20D2 23                INX   H          ;  =
3123   20D3 46                MOV   B,M        ;  TET TIME
3124   20D4 2AB642            LHLD  CTIME      ;H,L= EVTIM
3125   20D7 19                DAD   D          ;  =CTIME+ PRET TIME
3126   20D8 09                DAD   B          ;  + TET TIME
3127   20D9 EB                XCHG             ;D,E= EVTIM
3128   20DA E1                POP   H          ;POINT H,L TO BYTE 6
3129   20DB 2B                DCX   H          ;  IN TCB
3130   20DC 72                MOV   M,D        ;SET EVTIM
3131   20DD 2B                DCX   H          ;  IN
3132   20DE 73                MOV   M,E        ;  TCB
3133   20DF 11FBFF            LXI   D,0FFFBH   ;POINT H,L TO
3134   20E2 19                DAD   D          ;  TCB
3135   20E3 CD4429            CALL  ENQU2      ;SCHEDULE CMG
3136   20E6 C9                RET
3137                  ;;
3138                  ;LANE-FULL-PHOTO-BLOCKED PROCESSOR.
3139                  ;  CALLED BY PHPC OR RINI.
3140                  ;
3141                  ;INPUT: C= DIVERT #.
3142                  ;ALL REGISTERS EXCEPT B EFFECTED.
3143                  ;
3144   20E7 79        LFBPC:   MOV   A,C        ;A = DIVERT #
3145   20E8 CD250D              CALL  CPLFM      ;H,L POINTS TO BYTE LF, A= MASK
3146   20EB B6                  ORA   M          ;SET LF BIT IN LFDB
3147   20EC 77                  MOV   M,A        ;
3148   20ED C9                  RET
3149                  ;
3150                  ;
3151                  ;LANE-FULL-PHOTO-CLEARED PROCESSOR.
3152                  ;
3153                  ;INPUT: C= DIVERT #.
3154                  ;ALL REGISTERS EFFECTED.
3155                  ;
3156   20EE 79        LFCPC:   MOV   A,C        ;A = DIVERT #
3157   20EF 32FB42              STA   LFDV       ;SAVE DV #
3158   20F2 CD250D              CALL  CPLFM      ;A= LF BYTE MASK, H,L POINTS TO
3159                                             ;  BYTE LF
3160   20F5 2F                  CMA              ;RESET LF BIT IN LFDB
3161   20F6 A6                  ANA   M          ;
3162   20F7 77                  MOV   M,A        ;
3163                  ;
3164                  ;CONSIDER AUTO-INDUCTION AFTER LANE CLEARS
3165                  ;
3166   20F8 111000              LXI   D,10H      ;D,E=E=10H
3167   20FB 3A3C3D              LDA   LFO        ;IF LANE-CLEAR-AUTO-REINDUCT
3168   20FE A3                  ANA   E          ;  OPTION
3169   20FF C8                  RZ               ;  SPECIFIED
3170                  ;
3171   2100 2A4742              LHLD  KBDB       ;POINT H,L TO BYTE 4
3172   2103 010400              LXI   B,4        ;  IN KBDB
3173   2106 09                  DAD   B          ;  INITIALLY FOR KB # 1
3174   2107 0E01                MVI   C,1        ;C = INDUCT #, = 1 INITIALLY
3175   2109 7E        LFCAI:    MOV   A,M        ;A= AUTO-REINDUCT FLAG
3176   210A 93                  SUB   E          ;IF FLAG IS
3177   210B C23321              JNZ   LNXKB      ;  SET
3178                  ;
3179                  ;  ;CHECK DIVERT NUMBER
3180                  ;
3181   210E 22DA42              SHLD  KBDBP      ;SAVE POINTER FOR SNPC
3182   2111 23                  INX   H          ;POINT H,L TO BYTE 6
3183   2112 23                  INX   H          ;  IN KBDB
3184   2113 5E                  MOV   E,M        ;PICK UP
3185   2114 23                  INX   H          ;  PACKAGE IDENT
3186   2115 56                  MOV   D,M        ;  FROM
3187   2116 2B                  DCX   H          ;  KBDB
3188   2117 B2                  ORA   D          ;MS BYTE NONZERO MEANS
3189   2118 CA1C21              JZ    LFNXL      ;  CODE HAS BEEN TRANSLATED
3190   211B EB                  XCHG             ;IDENT POINTS TO DIVERT #
3191   211C 3AFB42    LFNXL:    LDA   LFDV       ;IF THIS DV LANE IS THE ONE
3192   211F 96                  SUB   M          ;  DESIRED THEN SET FLAG Z
3193   2120 C22D21              JNZ   LKBDP      ;IF THIS IS THE DV WANTED
3194                  ;
3195                  ;  ;AUTOMATIC INDUCTION
3196                  ;
3197   2123 2ADA42              LHLD  KBDBP      ;POINT H,L TO BYTE 4 IN KBDB
3198   2126 77                  MOV   M,A        ;RESET FLAG ER IN KBDB
```

```
3199  2127 CDDF0C            CALL  CLDP      ;CLEAR KB DISPLAY
3200  212A CD5B21            CALL  SNPC      ;AUTO-REINDUCT PACKAGE
3201                 ;
3202                 ;  ;CONSIDER NEXT INDUCT
3203                 ;
3204  212D 2ADA42    LKBDP:  LHLD  KBDBP     ;POINT H,L TO BYTE 4 IN KBDB
3205  2130 111000            LXI   D,10H     ;SET D,E=E=10H
3206  2133 19        LNXKB:  DAD   D         ;POINT TO NEXT INDUCT KBDB
3207  2134 0C                INR   C         ;NEXT JINDU-1
3208  2135 3A033D            LDA   NINDU     ;DO FOR
3209  2138 B9                CMP   C         ;  ALL
3210  2139 D20921            JNC   LFCAI     ;  INDUCTS
3211  213C C9                RET
3212                 ;;
3213                 ;RUN MODE "SEND" KEY PROCESSOR IN NORMAL KEYBOARD MODE.
3214                 ;  ';' TO ENTER OR REPEAT CODE FOR PACKAGE INDUCTION;
3215                 ;  '+' TO INSERT, CHANGE OR DELETE ENTRY IN TRANSLATE TABLE.
3216                 ;
3217                 ;INPUT: CHAR, C= KB #, KBDBP=H,L POINTS TO KBDB BYTE 4.
3218                 ;ALL REGISTERS EXCEPT C EFFECTED, JDV, IDENT, XYZ4 TO XYZ0,
3219                 ;  STACK, XLTBP, CODE, DELTA, TCBP, TREE EFFECTED.
3220                 ;
3221  213D 3ADE42    SNDPC:  LDA   CHAR      ;IF CHARACTER ';' THEN
3222  2140 FE3B              CPI   ';'       ;  SET FLAG Z
3223  2142 23                INX   H         ;PICK UP
3224  2143 7E                MOV   A,M       ;  NDIG IN KBDB
3225  2144 C2532F            JNZ   PSNPC     ;IF CHAR WAS INDEED ';'
3226                 ;
3227                 ;"SEND" PACKAGE
3228                 ;
3229  2147 B7                ORA   A         ;IF NDIG IN KBDB
3230  2148 C25521            JNZ   SENTR     ;  =0
3231                 ;
3232                 ;  ;PREVIOUS CHARACTER ENTERED WAS NOT DIGIT
3233                 ;
3234  214B 23                INX   H         ;IF
3235  214C 7E                MOV   A,M       ;  SND
3236  214D 23                INX   H         ;  IN KBDB
3237  214E B6                ORA   M         ;  NOT
3238  214F CA0D22            JZ    SILCM     ;  0
3239  2152 C35B21            JMP   SNPC      ;REPEAT PACKAGE CODE
3240                 ;
3241                 ;. ;NEW DIGIT STRING HAS BEEN ENTERED
3242                 ;
3243  2155 3A323D    SENTR:  LDA   SENDO     ;IF
3244  2158 E610              ANI   10H       ;  ENTER-BY-"SEND"
3245  215A C8                RZ              ;  ENABLED
3246                 ;
3247                 ;
3248                 ;AUTOMATIC "SEND" BY FIXED-LENGTH CODE, OR "REPEAT",
3249                 ;  OR CALLED BY LFCPC FOR LANE-CLEAR-AUTO-REINDUCT,
3250                 ;  OR CALLED BY RLTPC FOR DIRECT PACKAGE CODING BY DIVERT #.
3251                 ;
3252                 ;INPUT: C= KB #, KBDBP PONINTS TO BYTE 4 IN KBDB,
3253                 ;  FLAG XL ACTIVATES CODE TRANSLATION.
3254                 ;ALL REGISTERS EXCEPT C EFFECTED, KBDBP, JDV,IDENT EFFECTED.
3255                 ;
3256  215B 3A033D    SNPC:   LDA   NINDU     ;IF NINDU - KB # >= 0,
3257  215E B9                CMP   C         ;  I.E., IT IS AN
3258  215F 3E00              MVI   A,0       ;  (FOR ERRPC)
3259  2161 DA0D22            JC    SILCM     ;  INDUCTION KB
3260                 ;
3261                 ;DECODE DIVERT # OR PACKAGE CODE ENTERED
3262                 ;
3263  2164 2ADA42            LHLD  KBDBP     ;POINT TO
3264  2167 23                INX   H         ;  BYTE 6
3265  2168 23                INX   H         ;  IN KBDB
3266  2169 3AE042            LDA   XL        ;IF TRANSLATION
3267  216C B7                ORA   A         ;  IS
3268  216D CA8521            JZ    SNNXL     ;  NECESSARY
3269                 ;
3270                 ;  ;TRANSLATE PACKAGE CODE; DEFAULT IF NECESSARY
3271                 ;
3272  2170 CD9732            CALL  XLATE     ;TRANSLATE CODE
3273  2173 DA7D21            JC    SXLDF     ;IF CODE IS
3274  2176 3A9642            LDA   JDV       ;  OUT OF BOUND
3275  2179 B7                ORA   A         ;  OR NOT IN
3276  217A C29F21            JNZ   SNPC1     ;  TRANSLATE TABLE
3277  217D 3A393D    SXLDF:  LDA   SCNRO     ;TEST DEFAULT
3278  2180 E602              ANI   02H       ;  OPTION
3279  2182 C39021            JMP   SDFDV     ;DEFAULT IF NECESSARY
3280                 ;
3281                 ;  ;DECODE DIVERT NUMBER; DEFAULT IF NECESSARY
3282                 ;
3283  2185 CD810D    SNNXL:  CALL  DCDV      ;DECODE DIVERT #
3284  2188 D29F21            JNC   SNPC1     ;IF NOT LEGAL,
3285  218B 3A393D            LDA   SCNRO     ;IF DEFAULT
```

```
3286   218E E601            ANI   01H        ;  OPTION
3287   2190 CA0E22  SDFDV:   JZ    SILDV      ;  SPECIFIED
3288   2193 3A3A3D          LDA   SERDV      ;DEFAULT DIVERT # = SERDV
3289   2196 6F              MOV   L,A        ;  AND PACKAGE
3290   2197 2600            MVI   H,0        ;  IDENTIFICATION = SERDV
3291   2199 329642          STA   JDV        ;SET DIVERT # AND
3292   219C 229742          SHLD  IDENT      ;  PACKAGE IDENTIFICATION
3293                ;
3294                ;SET KEYBOARD DATA IN KBDB
3295                ;
3296   219F 2ADA42  SNPC1:   LHLD  KBDBP      ;CLEAR NDIG IN KBDB
3297   21A2 23              INX   H          ;  TO DENOTE
3298   21A3 3600            MVI   M,0        ;  NON-DIGIT CHAR
3299   21A5 23              INX   H          ;POINT TO BYTE 6 IN KBDB
3300   21A6 3A9742          LDA   IDENT      ;PUT IDENT INTO
3301   21A9 77              MOV   M,A        ;  SECONDARY
3302   21AA 23              INX   H          ;  BUFFER
3303   21AB 3A9842          LDA   IDENT+1    ;  SND IN
3304   21AE 77              MOV   M,A        ;  KBDB
3305                ;
3306                ;B= LEGAL DV #, CHECK LANE FULL
3307                ;
3308   21AF 3A3C3D  SNPC2:   LDA   LFO        ;IF LANE-FULL
3309   21B2 E601            ANI   01H        ;  KB-INHIBIT
3310   21B4 CAC121          JZ    SPLAQ      ;  IS DESIRED
3311   21B7 78              MOV   A,B        ;A= JDV
3312   21B8 C5              PUSH  B          ;SAVE JDV,JINDU
3313   21B9 CD250D          CALL  CPLFM      ;CP LF MASK IN A, H,L MASK POINTER
3314   21BC C1              POP   B          ;B = DIVERT #, C = INDUCT #
3315   21BD A6              ANA   M          ;IF LANE
3316   21BE C2FB21          JNZ   SDVLF      ;  NOT FULL
3317                ;
3318                ;PUT CODE INTO LAQ & ATTEMPT TO SEND PACKAGE
3319                ;
3320                ; IF IN I/C AND HALF-DAS MODE, SET IDENT=0 BEFORE LOADING IT INTO LAQ
3321                ;
3322   21C1 3A2B42  SPLAQ:   LDA   ICMO       ;TEST FOR I/C MODE
3323   21C4 D600            SUI   0          ;
3324   21C6 CAD921          JZ    CSPLA      ;JMP IF NOT I/C MODE
3325   21C9 3A0640          LDA   HDSEN      ;TEST FOR HALF-DAS MODE
3326   21CC D600            SUI   0          ;
3327   21CE CAD921          JZ    CSPLA      ;JMP IF NOT HALF-DAS
3328                ; ARE IN HALF-DAS MODE
3329   21D1 3E00            MVI   A,0        ;ZERO IDENT BEFORE LOADING INTO LAQ
3330   21D3 329742          STA   IDENT      ;
3331   21D6 329842          STA   IDENT+1    ;
3332   21D9 CDF02C  CSPLA:   CALL  PLAQ       ;PUT JDV,IDENT INTO LAQ,POINT PPDBP
3333   21DC CA0B22          JZ    SLAQF      ;  TO BYTE 1 IN PPDB
3334   21DF 2AEB42          LHLD  PPDBP      ;POINT H,L TO BYTE 1 IN PPDB
3335   21E2 7E              MOV   A,M        ;A= PWTNG IN PPDB
3336   21E3 3D              DCR   A          ;IF PACKAGE AT PP
3337   21E4 C2ED21          JNZ   SBEEP      ;  WAITING TO BE CODED
3338                ;
3339                ;SEND OUT WAITING PACKAGE IF NO MERGE CONFLICT
3340                ;
3341   21E7 C5              PUSH  B          ;SAVE INDUCT #
3342   21E8 23              INX   H          ;POINT H,L TO BYTE 2 IN PPDB
3343   21E9 CD7022          CALL  PSEQR      ;QUEUE PACKAGE UP FOR RELEASE
3344                ;
3345   21EC C1              POP   B          ;C= INDUCT #
3346   21ED 063B    SBEEP:   MVI   B,';'     ;BEEP
3347   21EF CD6C11          CALL  PUTKB      ;  KB
3348                ;
3349                ;CONSIDER REPEAT-BY-"SEND" OPTION
3350                ;
3351   21F2 3A323D          LDA   SENDO      ;IF REPEAT NOT DESIRED
3352   21F5 E620            ANI   20H        ;  THEN CLEAR
3353   21F7 CCC90C          CZ    CLRKB      ;  KB & KBDB
3354   21FA C9              RET
3355                ;
3356                ;DV LANE FULL
3357                ;
3358   21FB 3E07    SDVLF:   MVI   A,7        ;ERROR
3359   21FD CD130F          CALL  ERRPC      ;  7
3360   2200 3A3C3D          LDA   LFO        ;IF AUTO-REINDUCT-
3361   2203 E610            ANI   10H        ;  AFTER-LANE-CLEARS
3362   2205 C8              RZ               ;  DESIRED
3363   2206 2ADA42          LHLD  KBDBP      ;POINT H,L TO BYTE 4 IN KBDB
3364   2209 77              MOV   M,A        ;SET AUTO-REINDUCT FLAG IN KBDB
3365   220A C9              RET
3366                ;
3367                ;ERROR HANDLING
3368                ;
3369   220B 3D      SLAQF:   DCR   A          ;ERROR 06H MEANS LAQ FULL
3370   220C 3D              DCR   A          ;ERROR 07H MEANS DV LANE FULL
3371   220D 3D      SILCM:   DCR   A          ;ERROR 08H MEANS ILLEGAL COMMAND
3372   220E C609    SILDV:   ADI   09H        ;ERROR 09H MEANS ILLEGAL DV #
```

```
3373   2210 CD130F            CALL  ERRPC     ;PROCESS RUN ERROR
3374   2213 C9                RET
3375                  ;;
3376                  ;MASTER-CLEAR PROCESSOR FOR KEYBOARD # C.
3377                  ;  CLEARS KBDB, KB, PURGE LAQ IN PPDB,
3378                  ;  AND IF RELEASE IS OFF THEN REINITIALIZE INDUCT.
3379                  ;  CALLED BY KBPC.
3380                  ;
3381                  ;INPUT: C= KEYBOARD #.
3382                  ;ALL REGISTERS EXCEPT C, AND KBDB, PPDB EFFECTED.
3383                  ;
3384   2214 CDC90C    MCLPC:  CALL  CLRKB     ;CLEAR KBDB, KB
3385   2217 3A033D            LDA   NINDU     ;IF IT IS A NON-INDUCT KB
3386   221A B9                CMP   C         ;  THEN WE
3387   221B D8                RC              ;  ARE DONE
3388                  ;
3389                  ;CLEAR INDUCT LAQ
3390                  ;
3391   221C 79                MOV   A,C       ;A = INDUCT #
3392   221D CD9927            CALL  CPPPP     ;POINT H,L TO PPDB(JINDU)
3393   2220 E5                PUSH  H         ;SAVE POINTER
3394   2221 110600            LXI   D,6       ;POINT H,L TO BYTE 6
3395   2224 19                DAD   D         ;  IN PPDB
3396   2225 3600              MVI   M,0       ;PURGE LAQ
3397                  ;
3398                  ;IF RELEASE IS OFF THEN RE-INITIALIZE INDUCT
3399                  ;
3400   2227 C5                PUSH  B         ;SAVE C = INDUCT #
3401   2228 79                MOV   A,C       ;A = INDUCT #
3402   2229 2A5942            LHLD  OUTDB     ;POINT H,L TO OUTIM
3403   222C CD280D            CALL  CPMSK     ;CP A= MASK POINTED TO BY H,L
3404   222F A6                ANA   M         ;IF RELEASE OFF THEN SET FLAG Z
3405   2230 C1                POP   B         ;C=JINDU= INDUCT #
3406   2231 E1                POP   H         ;POINT H,L TO PPDB(JINDU)
3407   2232 C0                RNZ             ;IF RELEASE OFF
3408                  ;
3409   2233 23                INX   H         ;RETURN IF
3410   2234 23                INX   H         ;  RELEASE
3411   2235 23                INX   H         ;  IS
3412   2236 B6                ORA   M         ;  BEING
3413   2237 C0                RNZ             ;  DELAYED
3414                  ;
3415                  ;  ;CLEAN UP PPDB MERGE DATA
3416                  ;
3417   2238 2B                DCX   H         ;POINT TO
3418   2239 2B                DCX   H         ;  BYTE 0
3419   223A 2B                DCX   H         ;  IN PPDB
3420   223B 7E                MOV   A,M       ;A = FLAG PP IN PPDB
3421   223C 23                INX   H         ;POINT H,L TO BYTE 1 IN PPDB
3422   223D 77                MOV   M,A       ;PWTNG IN PPDB = PP IN PPDB
3423   223E B7                ORA   A         ;IF PP IN PPDB
3424   223F C24B22            JNZ   MICNT     ;  = 0
3425   2242 E5                PUSH  H         ;SAVE POINTER
3426   2243 C5                PUSH  B         ;SAVE C= INDUCT #
3427   2244 0601              MVI   B,1       ;SET RELEASE
3428   2246 CD0F11            CALL  OPUT      ;  ON
3429   2249 C1                POP   B         ;C= INDUCT #
3430   224A E1                POP   H         ;POINT H,L TO BYTE 1 IN PPDB
3431                  ;
3432                  ;  ;  ;IF STRAIGHT INDUCTS THEN ADJUST COUNTS IN PPDB
3433                  ;
3434   224B 3A313D    MICNT:  LDA   SIDEO     ;IF
3435   224E E610              ANI   010H      ;  STRAIGHT
3436   2250 C0                RNZ             ;  INDUCTION
3437                  ;
3438   2251 C5                PUSH  B         ;SAVE C=JINDU= INDUCT #
3439   2252 23                INX   H         ;B
3440   2253 46                MOV   B,M       ;  = COUNT(JINDU) IN PPDB
3441   2254 2A4342            LHLD  PPDB      ;POINT H,L TO BYTE 2
3442   2257 23                INX   H         ;  IN PPDB FOR
3443   2258 23                INX   H         ;  INDUCT # 1
3444   2259 112000            LXI   D,32      ;SET POINTER INCREMENT
3445   225C 3A033D            LDA   NINDU     ;C IS THE LOOP
3446   225F 4F                MOV   C,A       ;  COUNT
3447   2260 78                MOV   A,B       ;A= COUNT(JINDU)
3448   2261 BE        M9CNT:  CMP   M         ;IF COUNT(C)
3449   2262 D26622            JNC   MNXCN     ;  > COUNT(JINDU)
3450   2265 35                DCR   M         ;DEC COUNT(C)
3451   2266 19        MNXCN:  DAD   D         ;POINT H,L TO NEXT COUNT
3452   2267 0D                DCR   C         ;DO FOR ALL
3453   2268 C26122            JNZ   M9CNT     ;  INDUCTS
3454                  ;
3455   226B C1                POP   B         ;C= INDUCT #
3456   226C CDAC2C            CALL  PEND      ;ADJUST COUNT(JINDU)
3457   226F C9                RET
3458                  ;;
3459                  ;PACKAGE RELEASE SEQUENCER.  MERGE CODED PACKAGES ON A 1ST-COME-
```

```
3460                    ;  1ST-SERVE BASIS; OR SIDE-INDUCT INTO GAP ON CONVEYOR.
3461                    ;  CALLED BY PPBPC OR SNDPC.
3462                    ;
3463                    ;INPUT: C= INDUCT #, PPDB, H,L POINTS TO BYTE 2 IN PPDB.
3464                    ;OUTPUT: PPDB.
3465                    ;ALL REGISTERS EFFECTED.
3466                    ;
3467    2270 7E         PSEQR:  MOV   A,M       ;A= COUNT IN PPDB
3468    2271 2B                 DCX   H         ;POINT H,L TO BYTE 1 IN PPDB
3469    2272 B7                 ORA   A         ;IF COUNT IN PPDB
3470    2273 C29E22             JNZ   PSTOP     ;  =0
3471                    ;
3472                    ;SPACE AVAILABLE FOR PACKAGE TO BE RELEASED
3473                    ;
3474    2276 77                 MOV   M,A       ;PWTNG IN PPDB = 0
3475    2277 23                 INX   H         ;POINT H,L TO BYTE 2 IN PPDB
3476    2278 3A313D             LDA   SIDEO     ;IF IT IS
3477    227B E610               ANI   010H      ;  A STRAIGHT
3478    227D C29922             JNZ   PSIDE     ;  MERGE
3479                    ;
3480                    ;  ;STRAIGHT-MERGE INDUCT
3481                    ;
3482    2280 E5                 PUSH  H         ;SAVE POINTER TO BYTE 2 IN PPDB
3483    2281 2A4342             LHLD  PPDB      ;POINT H,L TO
3484    2284 23                 INX   H         ;  1ST COUNT
3485    2285 23                 INX   H         ;  IN PPDB
3486    2286 3A033D             LDA   NINDU     ;A= LOOP COUNT
3487    2289 112000             LXI   D,32      ;D,E= 32
3488    228C 0601               MVI   B,1       ;B=1
3489    228E 70         P1CNT:  MOV   M,B       ;EVERY COUNT IN PPDB = 1
3490    228F 19                 DAD   D         ;POINT H,L TO NEXT COUNT IN PPDB
3491    2290 3D                 DCR   A         ;DO FOR ALL
3492    2291 C28E22             JNZ   P1CNT     ;  INDUCTS
3493    2294 E1                 POP   H         ;POINT H,L TO BYTE 2 IN PPDB
3494    2295 CDCF22             CALL  RLSPC     ;PHYSICALLY RELEASE PACKAGE
3495    2298 C9                 RET
3496                    ;
3497                    ;  ;SIDE-MERGE INDUCT
3498                    ;
3499    2299 34         PSIDE:  INR   M         ;INC COUNT(JINDU) IN PPDB
3500    229A CDCF22             CALL  RLSPC     ;PHYSICALLY RELEASE PACKAGE
3501    229D C9                 RET
3502                    ;
3503                    ;SPACE NOT AVAILABLE ON MAINLINE FOR PACKAGE RELEASE
3504                    ;
3505    229E 3602       PSTOP:  MVI   M,2       ;PWTNG IN PPDB = 2
3506    22A0 5F                 MOV   E,A       ;E= COUNT(JINDU) IN PPDB
3507    22A1 3A313D             LDA   SIDEO     ;IF
3508    22A4 E610               ANI   010H      ;  STRAIGHT
3509    22A6 C2C922             JNZ   PRSRL     ;  MERGE
3510                    ;
3511    22A9 0601               MVI   B,1       ;B= INDUCT #, = 1 INITIALLY
3512    22AB 2A4342             LHLD  PPDB      ;POINT H,L TO
3513    22AE 23                 INX   H         ;  1ST COUNT
3514    22AF 23                 INX   H         ;  IN PPDB
3515    22B0 78         PICCN:  MOV   A,B       ;A= INDUCT #
3516    22B1 B9                 CMP   C         ;IF INDUCT #
3517    22B2 CABB22             JZ    PID01     ;  NOT= JINDU
3518    22B5 7E                 MOV   A,M       ;A= COUNT IN PPDB
3519    22B6 BB                 CMP   E         ;IF THIS COUNT
3520    22B7 DABB22             JC    PID01     ;  >= COUNT(JINDU)
3521    22BA 34                 INR   M         ;INC COUNT IN PPDB
3522    22BB D5         PID01:  PUSH  D         ;POINT H,L TO
3523    22BC 112000             LXI   D,32      ;  NEXT
3524    22BF 19                 DAD   D         ;  COUNT
3525    22C0 D1                 POP   D         ;  IN PPDB
3526    22C1 04                 INR   B         ;INC INDUCT #
3527    22C2 3A033D             LDA   NINDU     ;DO FOR
3528    22C5 B8                 CMP   B         ;  ALL
3529    22C6 D2B022             JNC   PICCN     ;  INDUCTS
3530                    ;
3531    22C9 0602       PRSRL:  MVI   B,2       ;RESET RELEASE
3532    22CB CD0F11             CALL  OPUT      ;  # JINDU
3533    22CE C9                 RET
3534                    ;;
3535                    ;PACKAGE RELEASE PROCESSOR.
3536                    ;  CALLED BY PSEQR, TECMG OR CMG.
3537                    ;
3538                    ;INPUT: C= INDUCT #, H,L POINTS TO BYTE 2 IN PPDB.
3539                    ;OUTPUT: EVENT TEC2, EVENT IDB2 OR UDB1.
3540                    ;ALL REGISTERS EFFECTED, JINDU, PPDBP, TCBP, RLDBP EFFECTED.
3541                    ;
3542    22CF 22EB42     RLSPC:  SHLD  PPDBP     ;SAVE POINTER
3543    22D2 CD622C             CALL  GLAQ      ;GET B=JDV, D,E=IDENT FROM LAQ
3544    22D5 CA8B23             JZ    RLSFX     ;  IN PPDB
3545    22D8 D5                 PUSH  D         ;SAVE IDENT
3546    22D9 C5                 PUSH  B         ;SAVE INDUCT #, DIVERT #
```

```
3547                    ;
3548                    ;DELAY PACKAGE IF IT HAS NOT BEEN STOPPED LONG ENOUGH
3549                    ;
3550    22DA CDA20F             CALL    GITRE       ;GET TREE PARAMETERS FOR INDUCT
3551    22DD 2AEB42             LHLD    PPDBP       ;POINT TO
3552    22E0 23                 INX     H           ;  BYTE 4
3553    22E1 23                 INX     H           ;  IN PPDB
3554    22E2 7E                 MOV     A,M         ;TEST IF PP-DWELL HAS
3555    22E3 23                 INX     H           ;  TIMED OUT
3556    22E4 B6                 ORA     M           ;  YET
3557    22E5 C1                 POP     B           ;B = DIVERT $, C = INDUCT $
3558    22E6 CAFD22             JZ      RTCBS       ;IF NOT YET TIMED OUT
3559    22E9 2B                 DCX     H           ;SET RELEASE DELAY
3560    22EA 2B                 DCX     H           ;  FLAG FOR
3561    22EB 3601               MVI     M,1         ;  MCLPC
3562    22ED C5                 PUSH    B           ;SAVE B,C
3563    22EE 0602               MVI     B,2         ;RELEASE
3564    22F0 CD0F11             CALL    OPUT        ;  OFF
3565    22F3 2A0C40             LHLD    GAP         ;DE = PPI GAP
3566    22F6 EB                 XCHG                ;  TO DELAY
3567    22F7 C1                 POP     B           ;MUST BE CAREFUL NOT TO
3568    22F8 E1                 POP     H           ;  LEAVE DATA IN STACK
3569    22F9 CD672D             CALL    RDLPP       ;DELAY PPI PULSES
3570    22FC E5                 PUSH    H           ;RE-SAVE IDENT
3571    22FD 79         RTCBS:  MOV     A,C         ;SAVE INDUCT $
3572    22FE 329442             STA     JINDU       ;  IN MEMORY
3573    2301 CD9927             CALL    CPPPP       ;POINT TO
3574    2304 23                 INX     H           ;  RELEASE DELAY
3575    2305 23                 INX     H           ;  FLAG IN
3576    2306 23                 INX     H           ;  PPDB
3577    2307 3600               MVI     M,0         ;CLEAR FLAG
3578                    ;
3579                    ;PREPARE TCB FOR IDB2 OR UDB1
3580                    ;
3581    2309 CDCB0F             CALL    GETFS       ;GET FREE BLOCK FOR TCB
3582    230C 22B442             SHLD    TCBP        ;SET TCBP FOR APEND
3583    230F 110D00             LXI     D,13        ;POINT H,L TO BYTE 13
3584    2312 19                 DAD     D           ;  IN TCB
3585    2313 D1                 POP     D           ;D,E=IDENT
3586    2314 72                 MOV     M,D         ;SET IDENT
3587    2315 2B                 DCX     H           ;  IN
3588    2316 73                 MOV     M,E         ;  TCB
3589    2317 2B                 DCX     H           ;POINT H,L TO BYTE 11 IN TCB
3590    2318 70                 MOV     M,B         ;SET JDV IN TCB
3591    2319 2B                 DCX     H           ;JUPDA IN TCB
3592    231A 3600               MVI     M,0         ;  = 0
3593    231C 2B                 DCX     H           ;SET JINDU
3594    231D 71                 MOV     M,C         ;  IN TCB
3595    231E 2B                 DCX     H           ;POINT H,L TO BYTE 8 IN TCB
3596                    ;
3597                    ;SET TCBP IN RLSDB FOR TECPC;  IF STRAIGHT INDUCT
3598                    ;  THEN ALSO APPEND TCB TO MERGE FIFO FOR IDBPC
3599                    ;
3600    231F E5                 PUSH    H           ;SAVE POINTER
3601    2320 C5                 PUSH    B           ;SAVE C=JINDU
3602    2321 CDA227             CALL    CPRLP       ;POINT H,L TO RLSDB
3603    2324 22EF42             SHLD    RLDBP       ;SAVE POINTER
3604    2327 23                 INX     H           ;POINT H,L TO BYTE 2
3605    2328 23                 INX     H           ;  IN RLSDB
3606    2329 3A313D             LDA     SIDE0       ;IF
3607    232C E610               ANI     010H        ;  STRAIGHT
3608    232E C24623             JNZ     RSIDE       ;  INDUCT
3609    2331 CD850C             CALL    APEND       ;APPEND TCB TO END OF MERGE FIFO
3610                    ;
3611                    ;SCHEDULE EVENT IDB2
3612                    ;
3613    2334 C1                 POP     B           ;C=JINDU
3614    2335 E1                 POP     H           ;POINT H,L TO BYTE 8 IN TCB
3615    2336 115824             LXI     D,IDB2      ;EVENT
3616    2339 72                 MOV     M,D         ;  IN TCB
3617    233A 2B                 DCX     H           ;  =
3618    233B 73                 MOV     M,E         ;  IDB2
3619    233C EB                 XCHG                ;POINT DE TO BYTE 7 IN TCB
3620    233D 2A2F42             LHLD    PRET        ;POINT HL TO PRET TABLE
3621    2340 CD2F29             CALL    ENTCB       ;SCHEDULE EVENT IDB2
3622    2343 C37423             JMP     RTEC2
3623                    ;
3624                    ;SCHEDULE EVENT UDB1 FOR SIDE INDUCT
3625                    ;
3626    2346 C1         RSIDE:  POP     B           ;C=JINDU
3627    2347 2A3F42             LHLD    FIDDB       ;POINT TO FOLLOWING-INDUCT DATABASE
3628    234A 3A043D             LDA     NUPDA       ;B IS THE LOOP
3629    234D 47                 MOV     B,A         ;  DOWN COUNT
3630    234E 79                 MOV     A,C         ;A = INDUCT $
3631    234F 05         RLSID:  DCR     B           ;SEARCH
3632    2350 FA5F23             JM      RNFID       ;  FOR
3633    2353 BE                 CMP     M           ;  INDUCT $
3634    2354 23                 INX     H           ;  IN
```

```
3635   2355 C24F23          JNZ   RLSID     ;  FIDDB
3636   2358 3A043D          LDA   NUPDA     ;A
3637   235B 90              SUB   B         ;  =
3638   235C C36023          JMP   RLUD1     ;  NEXT
3639   235F AF        RNFID: XRA   A         ;  UPDATE
3640   2360 3C        RLUD1: INR   A         ;  PHOTO $
3641   2361 E1              POP   H         ;POINT H,L TO BYTE 8 IN TCB
3642   2362 23              INX   H         ;SET UPCOMING
3643   2363 23              INX   H         ;  UPDATE
3644   2364 77              MOV   M,A       ;  PHOTO
3645   2365 2B              DCX   H         ;  IN
3646   2366 2B              DCX   H         ;  TCB
3647   2367 119424          LXI   D,UDB1    ;POINT EVENT
3648   236A 72              MOV   M,D       ;  IN TCB
3649   236B 2B              DCX   H         ;  TO
3650   236C 73              MOV   M,E       ;  UDB1
3651   236D EB              XCHG            ;POINT D,E TO BYTE 7 IN TCB
3652   236E 2A3342          LHLD  IDT       ;POINT H,L TO IDT TABLE
3653   2371 CD2F29          CALL  ENTCB     ;SCHEDULE EVENT UDB1
3654                ;
3655                ;SCHEDULE TEC2 AND SET TCBP IN RLSDB
3656                ;
3657   2374 2A0440  RTEC2:  LHLD  PLEMX     ;D,E = TE TIMEOUT
3658   2377 EB              XCHG            ;  PPI COUNT
3659   2378 2AEF42          LHLD  RLDBP     ;POINT TO TIMEOUT POINTER
3660   237B 01A223          LXI   B,TEC2    ;SCHEDULE TE
3661   237E CD192D          CALL  RSCHD     ;  TIMEOUT
3662   2381 3A9442          LDA   JINDU     ;C
3663   2384 4F              MOV   C,A       ;  =JINDU
3664   2385 0601            MVI   B,1       ;RELEASE
3665   2387 CD0F11          CALL  OPUT      ;  ON
3666   238A C9              RET
3667                ;
3668                ;EITHER LAQ EMPTY
3669                ;
3670   238B 3E03    RLSFX:  MVI   A,3       ;FATAL ERROR
3671   238D CD0C2C          CALL  FATAL     ;  3
3672   2390 C9              RET
3673                ;;
3674                ;PACKAGE-PRESENT DWELL TIMEOUT FOR ADDITIONAL DELAY
3675                ;  BEFORE RELEASING PACKAGE.
3676                ;
3677                ;INPUT: DE POINTS TO BYTE 8 IN TCB. TCBP POINTS TO TCB.
3678                ;ALL REGISTERS EXCEPT B, C EFFECTED.
3679                ;
3680   2391 EB      PPDTO:  XCHG            ;POINT TO
3681   2392 23              INX   H         ;  BYTE 10
3682   2393 23              INX   H         ;  IN TCB
3683   2394 5E              MOV   E,M       ;PICK UP
3684   2395 23              INX   H         ;  BACK
3685   2396 56              MOV   D,M       ;  POINTER
3686   2397 AF              XRA   A         ;CLEAR
3687   2398 12              STAX  D         ;  TIME
3688   2399 13              INX   D         ;  OUT
3689   239A 12              STAX  D         ;  POINTER
3690   239B 2AB442          LHLD  TCBP      ;RETURN BLOCK TO
3691   239E CD4F11          CALL  PUTFS     ;  FREE STORAGE
3692   23A1 C9              RET
3693                ;;
3694                ;TRAILING-EDGE-PHOTO-TIME-OUT EVENT.
3695                ;  CREATED BY RLSPC.
3696                ;
3697                ;INPUT: TCBP POINTS TO TCB, D,E POINTS TO BYTE 8 IN TCB.
3698                ;ALL REGISTERS EFFECTED.
3699                ;
3700   23A2 D5      TEC2:   PUSH  D         ;SAVE D,E
3701   23A3 EB              XCHG            ;POINT TO BYTE 9
3702   23A4 23              INX   H         ;  IN TCB
3703   23A5 4E              MOV   C,M       ;PICK UP INDUCT $
3704   23A6 23              INX   H         ;POINT TO BYTE 10
3705   23A7 5E              MOV   E,M       ;PICK UP
3706   23A8 23              INX   H         ;  BACK
3707   23A9 56              MOV   D,M       ;  POINTER
3708   23AA AF              XRA   A         ;CLEAR
3709   23AB 12              STAX  D         ;  TE
3710   23AC 13              INX   D         ;  TIMEOUT
3711   23AD 12              STAX  D         ;  POINTER
3712   23AE 3E0B            MVI   A,11      ;ERROR 11 TO
3713   23B0 CD130F          CALL  ERRPC     ;  INDUCT $ JINDU
3714                ;
3715                ;YIELD TO ANY NEXT PACKAGE TO BE INDUCTED
3716                ;
3717   23B3 00              NOP
3718   23B4 0602            MVI   B,2       ;STOP RELEASE
3719   23B6 CD0F11          CALL  OPUT      ;  $ JINDU
3720   23B9 D1              POP   D         ;POINT D,E TO BYTE 8 IN TCB
3721   23BA 3A313D          LDA   SIDEO     ;IF
3722   23BD E610            ANI   10H       ;  STRAIGHT
```

```
3723  23BF C2C623          JNZ   TE2SI      ;  MERGE
3724  23C2 CDCA23          CALL  TECMG      ;CLEAR MERGE AREA & POSSIBLY RELEASE
3725  23C5 C9              RET
3726                ;
3727  23C6 CD2924   TE2SI:  CALL  CMG        ;CLEAR MERGE AREA & POSSIBLY RELEASE
3728  23C9 C9              RET
3729                ;;
3730                ;CLEAR-MERGE EVENT SCHEDULED BY TECPC FOR STRAIGHT INDUCT,
3731                ;  OR CALLED BY TEC2.
3732                ;  RELEASE NEXT PENDING PACKAGE IF ANY.
3733                ;
3734                ;INPUT: TCBP, D,E POINTS TO BYTE 8 IN TCB.
3735                ;ALL REGISTERS EFFECTED, TCBP, PPDB AND RLSDB EFFECTED.
3736                ;
3737  23CA 3A033D   TECMG:  LDA   NINDU      ;B= INDUCT #
3738  23CD 47              MOV   B,A        ;  = LOOP COUNT
3739  23CE 2A4342          LHLD  PPDB       ;POINT H,L TO BYTE 2
3740  23D1 23              INX   H          ;  IN PPDB FOR
3741  23D2 23              INX   H          ;  INDUCT # 1
3742  23D3 112000          LXI   D,32       ;SET POINTER INCREMENT
3743  23D6 0E00            MVI   C,0        ;C FLAGS ANY PENDING PACKAGE TO RELEASE
3744  23D8 3E02            MVI   A,2        ;A=2
3745                ;
3746  23DA 35       T9CNT:  DCR   M          ;DEC COUNT(B)
3747  23DB FA0E24          JM    TCMFX      ;IF COUNT < 0 THEN FATAL ERROR
3748  23DE C2ED23          JNZ   TNXID      ;IF COUNT(B) = 0
3749  23E1 2B              DCX   H          ;IF
3750  23E2 BE              CMP   M          ;  PWTNG(B)
3751  23E3 23              INX   H          ;  =
3752  23E4 C2ED23          JNZ   TNXID      ;  2
3753                ;
3754  23E7 0C              INR   C          ;IF C ALREADY SET THEN
3755  23E8 0D              DCR   C          ;  FATAL ERROR
3756  23E9 C20E24          JNZ   TCMFX      ;  ELSE
3757  23EC 48              MOV   C,B        ;C= INVERSE INDUCT # FOR PACKAGE RELEASE
3758                ;
3759  23ED 19       TNXID:  DAD   D          ;POINT H,L TO NEXT COUNT IN PPDB
3760  23EE 05              DCR   B          ;DO FOR ALL
3761  23EF C2DA23          JNZ   T9CNT      ;  INDUCTS
3762                ;
3763                ;C= INDUCT # FOR PACKAGE TO BE RELEASED IF ANY
3764                ;
3765  23F2 2AB442          LHLD  TCBP       ;SAVE TCBP IN STACK TO AVOID
3766  23F5 E5              PUSH  H          ;  CONFLICT WITH RLSPC
3767  23F6 0D              DCR   C          ;C=0 MEANS NO MORE PACKAGE TO
3768  23F7 FA0924          JM    TCMGX      ;  RELEASE NOW
3769  23FA 3A033D          LDA   NINDU      ;CONVERT C TO
3770  23FD 91              SUB   C          ;  INDUCT #
3771  23FE 4F              MOV   C,A        ;
3772  23FF CDAC2C          CALL  PEND       ;ADJUST COUNTS IN PPDB FOR INDUCT
3773  2402 2B              DCX   H          ;POINT H,L TO BYTE 1 IN PPDB
3774  2403 3600            MVI   M,0        ;PWTNG(JINDU) = 0
3775  2405 23              INX   H          ;POINT H,L TO BYTE 2 IN PPDB
3776  2406 CDCF22          CALL  RLSPC      ;RELEASE PACKAGE
3777  2409 E1       TCMGX:  POP   H          ;RETURN TCB TO
3778  240A CD4F11          CALL  PUTFS      ;  FREE STORAGE
3779  240D C9              RET
3780                ;
3781                ;ILLEGAL COUNT FOUND IN PPDB
3782                ;
3783  240E 3E06     TCMFX:  MVI   A,6        ;FATAL ERROR
3784  2410 CD0C2C          CALL  FATAL      ;  6
3785  2413 2AB442          LHLD  TCBP       ;RETURN TCB TO
3786  2416 CD4F11          CALL  PUTFS      ;  FREE STORAGE
3787  2419 C9              RET
3788                ;;
3789                ;BLOCK-MERGE EVENT SCHEDULED BY UDBPC.  FOR SIDE INDUCT ONLY.
3790                ;
3791                ;INPUT: TCBP POINTS TO TCB, D,E POINTS TO BYTE 8 IN TCB.
3792                ;
3793  241A 13       BMG:    INX   D          ;A
3794  241B 1A              LDAX  D          ;  =JINDU
3795  241C CD9927          CALL  CPPPP      ;POINT H,L TO
3796  241F 23              INX   H          ;  BYTE 2
3797  2420 23              INX   H          ;  IN PPDB
3798  2421 34              INR   M          ;INC COUNT IN PPDB
3799                ;
3800  2422 2AB442          LHLD  TCBP       ;RETURN TCB TO
3801  2425 CD4F11          CALL  PUTFS      ;  FREE STORAGE
3802  2428 C9              RET
3803                ;;
3804                ;CLEAR-MERGE EVENT FOR A SIDE INDUCT.
3805                ;  SCHEDULED BY SCCMG OR CALLED BY UDBPC.
3806                ;
3807                ;INPUT: TCBP, D,E POINTS TO BYTE 8 IN TCB.
3808                ;ALL REGISTERS EFFECTED, TCBP EFFECTED.
3809                ;
```

```
3810   2429 2AB442   CMG:    LHLD  TCBP      ;SAVE TCB
3811   242C E5               PUSH  H         ;  POINTER
3812   242D 13               INX   D         ;POINT D,E TO BYTE 9 IN TCB
3813   242E 1A               LDAX  D         ;A=JINDU
3814   242F 4F               MOV   C,A       ;C=JINDU
3815   2430 CD9927           CALL  CPPPP     ;POINT H,L TO BYTE 1
3816   2433 23               INX   H         ;  IN PPDB
3817   2434 7E               MOV   A,M       ;A= PWTNG(JINDU)
3818   2435 23               INX   H         ;POINT H,L TO BYTE 2 IN PPDB
3819   2436 35               DCR   M         ;DEC COUNT(JINDU) IN PPDB
3820   2437 FA4C24           JM    CMGEX     ;IF COUNT NOT< 0
3821                 ;
3822   243A C25324           JNZ   CMGX      ;IF COUNT = 0
3823                 ;
3824   243D D602             SUI   2         ;IF PWTNG(JINDU) = 2 THEN
3825   243F C25324           JNZ   CMGX      ;  A = 0, AND
3826                 ;
3827                 ;PACKAGE PENDING AT INDUCT # C
3828                 ;
3829   2442 34               INR   M         ;INC COUNT(JINDU)
3830   2443 2B               DCX   H         ;PWTNG(JINDU)
3831   2444 77               MOV   M,A       ;  = 0
3832   2445 23               INX   H         ;POINT H,L TO BYTE 2 IN PPDB
3833   2446 CDCF22           CALL  RLSPC     ;RELEASE PACKAGE (TCBP EFFECTED)
3834   2449 C35324           JMP   CMGX
3835                 ;
3836                 ;ILLEGAL COUNT IN PPDB
3837                 ;
3838   244C 3600     CMGEX:  MVI   M,0       ;CLEAR COUNT(JINDU)
3839   244E 3E0D             MVI   A,13      ;ERROR
3840   2450 CD130F           CALL  ERRPC     ;  13
3841                 ;
3842                 ;EVENT EXIT
3843                 ;
3844   2453 E1       CMGX:   POP   H         ;RETURN TCB TO
3845   2454 CD4F11           CALL  PUTFS     ;  FREE STORAGE
3846   2457 C9               RET
3847                 ;;
3848                 ;TIME-OUT EVENT FOR IDBPC.  FOR STRAIGHT INDUCT ONLY.
3849                 ;
3850                 ;INPUT: TCBP, D,E POINTS TO BYTE 8 IN TCB.
3851                 ;ALL REGISTERS EFFECTED.
3852                 ;
3853   2458 EB       IDB2:   XCHG            ;POINT H,L TO BYTE 9
3854   2459 23               INX   H         ;  IN TCB
3855   245A 4E               MOV   C,M       ;C = INDUCT #
3856   245B CDA227           CALL  CPRLP     ;POINT HL TO
3857   245E 23               INX   H         ;  BYTE 2
3858   245F 23               INX   H         ;  IN RLSDB
3859   2460 5E               MOV   E,M       ;DE = PSEQ IN RLSDB,
3860   2461 23               INX   H         ;  I.E., POINT DE TO
3861   2462 56               MOV   D,M       ;  INDUCT FIFO
3862   2463 3AB442           LDA   TCBP      ;IF THIS TCB
3863   2466 BB               CMP   E         ;  IS THE
3864   2467 C28E24           JNZ   IB2FX     ;  1ST TCB
3865   246A 3AB542           LDA   TCBP+1    ;  IN
3866   246D BA               CMP   D         ;  INDUCT
3867   246E C28E24           JNZ   IB2FX     ;  FIFO
3868                 ;
3869                 ;NOTHING FOUND WRONG WITH THE INDUCT FIFO
3870                 ;
3871   2471 E5               PUSH  H         ;SAVE POINTER TO BYTE 3 IN RLSDB
3872   2472 210E00           LXI   H,14      ;POINT HL TO BYTE 14
3873   2475 19               DAD   D         ;  IN TCB
3874   2476 5E               MOV   E,M       ;POINT DE
3875   2477 23               INX   H         ;  TO NEXT
3876   2478 56               MOV   D,M       ;  PACKAGE
3877   2479 E1               POP   H         ;POINT HL TO BYTE 3 IN RLSDB
3878   247A 72               MOV   M,D       ;POINT PSEQ IN RLSDB
3879   247B 2B               DCX   H         ;  TO NEXT
3880   247C 73               MOV   M,E       ;  PACKAGE
3881                 ;
3882   247D 3E02             MVI   A,2       ;ERROR
3883   247F CD130F           CALL  ERRPC     ;  2
3884                 ;
3885   2482 0602             MVI   B,2       ;RELEASE
3886   2484 CD0F11           CALL  OPUT      ;  OFF
3887   2487 2AB442           LHLD  TCBP      ;RETURN TCB TO
3888   248A CD4F11           CALL  PUTFS     ;  FREE STORAGE
3889   248D C9               RET
3890                 ;
3891                 ;INDUCT FIFO AT PSEQ IN RLSDB HAS BEEN SCRIBBLED
3892                 ;
3893   248E 3E09     IB2FX:  MVI   A,9       ;FATAL ERROR
3894   2490 CD0C2C           CALL  FATAL     ;  9
3895   2493 C9               RET
3896                 ;;
```

```
3897                    ;EVENT UDB1 IS A PROPER LOWER BOUND FOR UPDATE PHOTO BLOCKED.
3898                    ;
3899                    ;INPUT: TCBP, D.E POINTS TO BYTE 8 OF TCB, TREE PARAMETERS.
3900                    ;ALL REGISTERS EFFECTED.
3901                    ;
3902  2494 CD8725       UDB1:     CALL   UDB12      ;POINT H,L TO PMUZ IN UDDB
3903  2497 5E                     MOV    E,M        ;D,E
3904  2498 23                     INX    H          ;  = PMUZ
3905  2499 56                     MOV    D,M        ;  IN UDDB
3906  249A 7B                     MOV    A,E        ;IF
3907  249B B2                     ORA    D          ;  PMUZ NOT= 0
3908  249C CABB24                 JZ     UB1        ;
3909                    ;
3910                    ;OVERLAPPING MUZ'S, LATTER OVERRIDES FORMER
3911                    ;
3912  249F E5                     PUSH   H          ;SAVE POINTER TO PMUZ MS BYTE
3913  24A0 210700                 LXI    H,7        ;H,L POINTS TO BYTE 7 OF TCB
3914  24A3 19                     DAD    D          ;  POINTED TO BY PMUZ IN UDDB
3915  24A4 118025                 LXI    D,JUNK     ;JUNK
3916  24A7 73                     MOV    M,E        ;  THAT TCB
3917  24A8 23                     INX    H          ;  (ZONE POINTER
3918  24A9 72                     MOV    M,D        ;  OVERRIDEN)
3919  24AA 3E03                   MVI    A,3        ;ERROR 3
3920  24AC CD080F                 CALL   ERPC       ;  TO MASTER KB
3921  24AF E1                     POP    H          ;H,L POINTS TO PMUZ MS BYTE
3922  24B0 AF                     XRA    A          ;CLEAR A
3923  24B1 77                     MOV    M,A        ;CLEAR POINTER
3924  24B2 2B                     DCX    H          ;  PMUZ
3925  24B3 77                     MOV    M,A        ;  IN UDDB
3926  24B4 2AB442                 LHLD   TCBP       ;RETURN TCB TO
3927  24B7 CD4F11                 CALL   PUTFS      ;  FREE STORAGE
3928  24BA C9                     RET
3929                    ;
3930                    ;LEGITIMATE EVENT UDB1
3931                    ;
3932  24BB EB           UB1:      XCHG              ;POINT D,E
3933  24BC 2AB442                 LHLD   TCBP       ;  TO
3934  24BF EB                     XCHG              ;  TCB
3935  24C0 72                     MOV    M,D        ;POINT PMUZ
3936  24C1 2B                     DCX    H          ;  IN UDDB
3937  24C2 73                     MOV    M,E        ;  TO TCB
3938                    ;
3939  24C3 210800                 LXI    H,8        ;POINT H,L TO BYTE 8
3940  24C6 19                     DAD    D          ;  IN TCB
3941  24C7 11D924                 LXI    D,UDB2     ;POINT EVENT
3942  24CA 72                     MOV    M,D        ;  IN TCB
3943  24CB 2B                     DCX    H          ;  TO
3944  24CC 73                     MOV    M,E        ;  UDB2
3945  24CD EB                     XCHG              ;POINT D,E TO BYTE 7 IN TCB
3946  24CE 2A3542                 LHLD   ZNT        ;POINT H,L TO ZNT TABLE
3947  24D1 3A9542                 LDA    JUPDA      ;C
3948  24D4 4F                     MOV    C,A        ;  = UPDATE PHOTO #
3949  24D5 CD2F29                 CALL   ENTCB      ;SCHEDULE EVENT UDB2
3950  24D8 C9                     RET
3951                    ;;
3952                    ;LEADING EDGE OF PACKAGE NOT FOUND IN MOVING UPDATABLE ZONE.
3953                    ;
3954                    ;INPUT: TCBP, D.E POINTS TO BYTE 8 IN TCB.
3955                    ;ALL REGISTERS EFFECTED.
3956                    ;
3957  24D9 CD8725       UDB2:     CALL   UDB12      ;POINT H,L TO PMUZ IN UDDB
3958  24DC AF                     XRA    A          ;CLEAR A
3959  24DD 77                     MOV    M,A        ;ZERO
3960  24DE 23                     INX    H          ;  PMUZ IN UDDB
3961  24DF 77                     MOV    M,A        ;
3962  24E0 3E04                   MVI    A,4        ;PUT ERROR 4 ON KB
3963  24E2 CD080F                 CALL   ERPC       ;
3964                    ;
3965                    ; TEST TO SEE IF UPDATE-DIVERT-COMPLETE OPTION SPECIFIED
3966  24E5 3A3B3D                 LDA    DCO                ;DCO=2?
3967  24E8 D602                   SUI    02H
3968  24EA C28025                 JNZ    JUNK               ;JMP IF DCO NOT = 2.
3969                    ;
3970                    ; TEST FOR I/C MODE
3971  24ED 3A2B42                 LDA    ICMO               ;ICMO=1?
3972  24F0 A7                     ANA    A
3973  24F1 CA8025                 JZ     JUNK               ;JMP IF NOT IN I/C MODE
3974                    ;
3975                    ; TEST IF MESSAGE M IS MASKED
3976  24F4 212B3D                 LXI    H,ERMKO+3          ;H,L= MASK BYTE 4 LOC
3977  24F7 7E                     MOV    A,M                ;A= MASK BYTE 4
3978  24F8 E610                   ANI    10H                ;LOOK AT BIT 4
3979  24FA CA8025                 JZ     JUNK               ;JMP IF M NOT MASKED
3980                    ;
3981                    ; ARE IN I/C MODE,MESSAGE M IS MASKED, AND DCO=2 IS SPECIFIED. SEND
3982                    ; MESSAGE M TO DAS(BY CALLING DSLD) FIRST TEST IF IDENT=0.
3983  24FD 2AB442                 LHLD   TCBP               ;H,L= TCB LOC
3984  2500 110C00                 LXI    D,12
```

```
3985   2503 19              DAD   D               ;H,L= IDENT LSB LOC
3986   2504 7E              MOV   A,M             ;A= IDENT LSB
3987   2505 23              INX   H               ;H,L= IDENT MSB LKOC
3988   2506 46              MOV   B,M             ;B= IDENT MSB
3989   2507 B0              ORA   B
3990   2508 C21D25          JNZ   TRANS           ;JMP IF IDENT NOT = 0
3991                ;
3992                ; IDENT IS ZERO. THEREFORE PACKAGE CODE IS ZERO (DAS WANTS TO RECIRC
3993                ; PACKAGE) AND IS NOT IN TRANSACTION TABLE. THUS PUT 6 ASCII 0'S INTO
3994                ; BUFFER BYT16 FOR PACKAGE CODE VALUE
3995   250B 211843          LXI   H,BYT16         ;H,L= BYT16 LOC
3996   250E 23              INX   H
3997   250F 23              INX   H               ;POINT TO BYTE 2
3998   2510 3E30            MVI   A,'0'           ;A=ASCII 0
3999   2512 0606            MVI   B,6             ;LOOP COUNT=6
4000   2514 77       PC000: MOV   M,A             ;BYTE = ASCII0
4001   2515 23              INX   H               ;POINT TO NEXT BYTE
4002   2516 05              DCR   B               ;DECREMENT COUNTER
4003   2517 C21425          JNZ   PC000           ;DO FOR BYTES 2-7
4004   251A C33025          JMP   CVTJD           ;JMP TO CVTJD WHEN LOADED
4005                ;
4006                ; ACCESS PACKAGE CODE FROM TRANSACTION TABLE AND PUT INTO BYT16 BUFFER.
4007                ; CLEAR PACKAGE CODE FROM TRANSACTION TABLE
4008   251D 2AB442  TRANS: LHLD  TCBP            ;H,L= TCB LOC
4009   2520 110C00          LXI   D,12
4010   2523 19              DAD   D               ;H,L= IDENT LSB LOC
4011   2524 5E              MOV   E,M             ;E= IDENT LSB
4012   2525 23              INX   H               ;H,L= IDENT MSB LOC
4013   2526 56              MOV   D,M             ;D= IDENT MSB
4014   2527 EB              XCHG                  ;H,L=IDENT=LOC IN TRANS TAB OF PACK CODE
4015   2528 E5              PUSH  H               ;STACK=SAME
4016   2529 CD6C1A          CALL  TRANP           ;MOVE PACK CODE TO BYTES 2-7 OF BYT16
4017   252C E1              POP   H               ;H,L= PACK CODE LOC AGAIN
4018   252D CD401A          CALL  ZPACK           ;REMOVE PACK CODE FROM TRANS TAB
4019                ;
4020                ; LOAD THE ASSUMED DVT$ VVV INTO BYTES 10-12 OF BYT16. THE ASSUMED
4021                ; DVT$ IS LOCATED IN LDVDB IN RELATIVE LOC JUPDA-2
4022   2530 2AB442  CVTJD: LHLD  TCBP            ;H,L= TCB LOC
4023   2533 110A00          LXI   D,10
4024   2536 19              DAD   D               ;H,L= JUPDA LOC
4025   2537 7E              MOV   A,M             ;A=JUPDA
4026   2538 D602            SUI   2               ;A=JUPDA-2
4027   253A F24225          JP    CVTJ1           ;JMP IF DVT$ >1
4028   253D 3E00            MVI   A,0             ;A= 0
4029   253F C34A25          JMP   CVTJ2           ;JMP IF PACKAGE DISAPPEARED BEFORE DV$1
4030   2542 1600    CVTJ1: MVI   D,0             ;D= 0
4031   2544 5F              MOV   E,A             ;D,E= REL LDVDB TABLE LOC OF DV$
4032   2545 2A3D42          LHLD  LDVDB           ;H,L= LDVDB LOC
4033   2548 19              DAD   D               ;H,L= LOC IN LDVDB TABLE OF DESIRED DVT$
4034   2549 7E              MOV   A,M             ;A= DVT$ IN BINARY
4035   254A CD010E  CVTJ2: CALL  ECDIG           ;CONVERT TO ASCII
4036   254D 3A7D42          LDA   XYZ2
4037   2550 322243          STA   BYT16+10        ;STORE ASCII 100'S
4038   2553 3A7E42          LDA   XYZ1
4039   2556 322343          STA   BYT16+11        ;STORE ASCII 10'S
4040   2559 3A7F42          LDA   XYZ0
4041   255C 322443          STA   BYT16+12        ;STORE ASCII 1'S
4042                ; CONVERT BINARY JINDU INTO ASCII AND STORE IN BYTES 8-9 OF BYT16
4043   255F 2AB442          LHLD  TCBP            ;H,L = TCB LOC
4044   2562 110900          LXI   D,9
4045   2565 19              DAD   D               ;H,L= JINDU LOC
4046   2566 7E              MOV   A,M             ;A= JINDU IN BINARY
4047   2567 CD010E          CALL  ECDIG           ;CONVERT TO ASCII
4048   256A 3A7E42          LDA   XYZ1
4049   256D 322043          STA   BYT16+8         ;STORE ASCII 10'S
4050   2570 3A7F42          LDA   XYZ0
4051   2573 322143          STA   BYT16+9         ;STORE ASCII 1'S
4052   2576 211843          LXI   H,BYT16         ;H,L=BYT16
4053   2579 23              INX   H               ;H,L= BYTE 1 OF BYT16
4054   257A 3E4D            MVI   A,'M'           ;A= ASCII 'M'
4055   257C 77              MOV   M,A             ;PUT ASCII M INTO BYTE 1 OF BYT16
4056                ; LOAD BYT16 INTO FS
4057   257D CD7D1A          CALL  DSLD            ;PUT MESSAGE M INTO FS
4058                ; RETURN PACKAGE TCB TO FS
4059                ;
4060                ;
4061                ;JUNK AND NO-OPERATION EVENTS DELETED FROM TREE.
4062                ;CREATED BY UDB1, UDBPC, MCLPC.
4063                ;
4064                ;INPUT: TCBP.
4065                ;REGISTERS D, E, H, L EFFECTED.
4066                ;
4067   2580 2AB442  JUNK:  LHLD  TCBP        ;RETURN TCB TO
4068   2583 CD4F11          CALL  PUTFS      ;  FREE STORAGE
4069                ;
4070   2586 C9      NOOP:  RET
4071                ;
```

```
4072                    ;
4073                    ;1ST PART OF UDB1, UDB2.  POINT H,L TO PMUZ IN UDDB.
4074                    ;
4075                    ;INPUT: DE POINTS TO BYTE 8 IN TCB.
4076                    ;OUTPUT: DE POINTS TO BYTE 10 IN TCB,
4077                    ;  JUPDA= UPDATE PHOTO $, BC= UPDATE $ - 1.
4078                    ;ALL REGISTERS EFFECTED.
4079                    ;
4080  2587 13           UDB12:    INX   D          ;D,E POINTS TO
4081  2588 13                     INX   D          ;  BYTE 10 IN TCB
4082  2589 1A                     LDAX  D          ;A = JUPDA (UPDATE PHOTO $)
4083  258A 329542                 STA   JUPDA      ;SET UPDATE PHOTO $ IN MEMORY
4084  258D 2A5142                 LHLD  UDDB       ;H,L POINTS TO UDDB TABLE
4085  2590 4F                     MOV   C,A        ;B,C
4086  2591 0600                   MVI   B,0        ;  =JUPDA
4087  2593 0B                     DCX   B          ;B,C=JUPDA-1
4088  2594 09                     DAD   B          ;H,L
4089  2595 09                     DAD   B          ;  POINTS TO
4090  2596 09                     DAD   B          ;  PMUZ IN UDDB
4091  2597 09                     DAD   B          ;  FOR UD $ JUPDA
4092  2598 C9                     RET
4093                    ;;
4094                    ;CHANGED FROM UDB2 BY UDBPC.  SET BYTES 7, 8 IN TCB
4095                    ;  ZERO TO DENOTE TCB OUT OF TREE.
4096                    ;
4097                    ;INPUT: TCBP.
4098                    ;REGISTERS D, E, H, L EFFECTED.
4099                    ;
4100  2599 2AB442       UDB3:     LHLD   TCBP            ;POINT H,L
4101  259C 110700               LXI    D,7             ;  TO BYTE 7
4102  259F 19                   DAD    D               ;  IN TCB
4103  25A0 AF                   XRA    A               ;A=0
4104  25A1 77                   MOV    M,A             ;ZERO
4105  25A2 23                   INX    H               ;  BYTES
4106  25A3 77                   MOV    M,A             ;  7, 8
4107  25A4 C9                   RET
4108                    ;
4109                    ;         SUBROUTINE DCB1
4110                    ;
4111                    ;         INPUTS: TCBP,D,E POINTS TO BYTE 8 OF TCB.TREE PARAMETERS.
4112                    ;
4113                    ;         OUTPUTS: EVENT DCB2 CREATED AND SCHEDULED. DCB2 IS UPPER
4114                    ;                  BOUND ON MUZ
4115                    ;
4116                    ;         PROCESS: SAME AS FOR EVENT UDB1. ONLY DIFFERENCE IS THAT WE
4117                    ;                  ARE CREATING AND SCHEDULING DCB2 INSTEAD OF UDB2
4118                    ;
4119  25A5 CD8725       DCB1:     CALL   UDB12           ;POINT H,L TO PMUZ IN UDDB
4120  25A8 5E                     MOV    E,M             ;D,E
4121  25A9 23                     INX    H               ;    = PMUZ
4122  25AA 56                     MOV    D,M             ;    INUDDB
4123  25AB 7B                     MOV    A,E             ;IF
4124  25AC B2                     ORA    D               ;    PMUZ NOT= 0
4125  25AD CACC25                 JZ     DB1             ;
4126                    ;
4127                    ;OVERLAPPING MUZ'S, LATTER OVERRIDES FORMER
4128                    ;
4129  25B0 E5                     PUSH   H               ;SAVE POINTER TO PMUZ MS BYTE
4130  25B1 210700                 LXI    H,7             ;H,L POINTS TO BYTE 7 OF TCB
4131  25B4 19                     DAD    D               ;    POINTED TO BY PMUZ IN UDDB
4132  25B5 118025                 LXI    D,JUNK          ;JUNK
4133  25B8 73                     MOV    M,E             ;    THAT TCB
4134  25B9 23                     INX    H               ;    (ZONE POINTER
4135  25BA 72                     MOV    M,D             ;    OVERRIDEN)
4136  25BB 3E03                   MVI    A,3             ;ERROR 3
4137  25BD CD080F                 CALL   ERPC            ;    TO MASTER KB
4138  25C0 E1                     POP    H               ;H,L POINTS TO PMUZ MS BYTE
4139  25C1 AF                     XRA    A               ;CLEAR A
4140  25C2 77                     MOV    M,A             ;CLEAR POINTER
4141  25C3 2B                     DCX    H               ;    PMUZ
4142  25C4 77                     MOV    M,A             ;    IN UDDB
4143  25C5 2AB442                 LHLD   TCBP            ;RETURN TCB TO
4144  25C8 CD4F11                 CALL   PUTFS           ;    FREE STORAGE
4145  25CB C9                     RET
4146                    ; LEGITIMATE EVENT DCB1
4147                    ;
4148  25CC EB           DB1:      XCHG                   ;POINT D,E
4149  25CD 2AB442                 LHLD   TCBP            ;    TO
4150  25D0 EB                     XCHG                   ;    TCB
4151  25D1 72                     MOV    M,D             ;SET PMUZ= CURRENT PACKAGE TCB LOC
4152  25D2 2B                     DCX    H
4153  25D3 73                     MOV    M,E
4154  25D4 210800                 LXI    H,8             ;POINT H,L TO BYTE 8
4155  25D7 19                     DAD    D               ;    IN TCB.
4156  25D8 11EA25                 LXI    D,DCB2          ;SET EVENT = DCB2
4157  25DB 72                     MOV    M,D
4158  25DC 2B                     DCX    H
```

```
4159   25DD 73              MOV   M,E
4160   25DE EB              XCHG                    ;D,E= BYTE 7 IN TCB
4161   25DF 2A3B42          LHLD  DCT               ;H,L= 8000 SERIES TABLE
4162   25E2 3A9542          LDA   JUPDA
4163   25E5 4F              MOV   C,A               ;C= JUPDA
4164   25E6 CD2F29          CALL  ENTCB             ;SCHEDULE EVENT DCB2
4165   25E9 C9              RET
4166                ;       SUBROUTINE DCB2
4167                ;
4168                ;       INPUTS: TCBP, D,E POINTS TO BYTE 8 INTCB
4169                ;
4170                ;       OUTPUTS: PMUZ SET=0, ERROR MESSAGE
4171                ;                'S' TO DAS IF MASKED. PACKAGE TCB RETURNED TO FS
4172                ;
4173                ;       PROCESS: SAME AS FOR UDB2
4174                ;
4175   25EA CD8725  DCB2:   CALL  UDB12     ;POINT H,L TO PMUZ IN UDDB
4176   25ED AF              XRA   A         ;CLEA A
4177   25EE 77              MOV   M,A       ;ZERO
4178   25EF 23              INX   H         ;   PMUZ IN UDDB
4179   25F0 77              MOV   M,A
4180                ; CHECK TO SEE IF IN I/C MODE
4181   25F1 3A2B42          LDA   ICMO      ;ICMO=1?
4182   25F4 A7              ANA   A
4183   25F5 CA6F26          JZ    RTFS      ;JMP IF NOT I/C MODE
4184                ; CHECK TO SEE IF MESSAGE S IS MASKED
4185   25F8 21283D          LXI   H,ERMKO   ;H,L=ERROR MASK LOC
4186   25FB 23              INX   H
4187   25FC 23              INX   H         ;H,L= ERROR MASK BYTE 2 (LOC 2A)
4188   25FD 7E              MOV   A,M       ;LOOK AT MASK
4189   25FE E601            ANI   01H       ;LOOK AT BYTE 0
4190   2600 CA6F26          JZ    RTFS      ;JMP IF MESSAGE A NOT MASKED
4191                ;
4192                ; SEND MESSAGE S TO DAS. CLEAR PACKAGE CODE FROM TRANSACTION TABLE
4193                ; IF IDENT NOT = 0. SEND MESSAGE S BY CALLING DSLD.
4194   2603 2AB442          LHLD  TCBP      ;H,L= PACKAGE TCB LOC
4195   2606 110C00          LXI   D,12
4196   2609 19              DAD   D         ;H,L= BYTE 12 OF TCB
4197   260A 5E              MOV   E,M       ;E= LSB OF IDENT
4198   260B 23              INX   H
4199   260C 56              MOV   D,M       ;D= MSB OF IDENT
4200   260D EB              XCHG            ;H,L= LOC IN TRANSACTION TAB OF PACK CODE
4201   260E E5              PUSH  H         ;STACK= SAME
4202   260F 7C              MOV   A,H       ;A= IDENT MSB
4203   2610 B5              ORA   L         ;IDENT=0 ?
4204   2611 CA1E26          JZ    SABO      ;JMP IF IDENT=0
4205   2614 CD6C1A          CALL  TRANP     ;TRANSFER PACK CODE TO BYT16 BYTES 2-7
4206   2617 E1              POP   H         ;H,L=LOC IN TRANS TAB OF PACK CODE
4207   2618 CD401A          CALL  ZPACK     ;REMOVE PACK CODE FROM TABLE
4208   261B C33326          JMP   DABO      ;JMP TO DABO
4209                ; PACKAGE IDENT = 0. THEREFORE PACK NOT IN TRANS TAB. SET CODE=0
4210   261E 3E30    SABO:   MVI   A,'0'     ;A= ASCII 0
4211   2620 321A43          STA   BYT16+2   ;SET PACKAGE CODE=0 IN BUFFER BYT16
4212   2623 321B43          STA   BYT16+3
4213   2626 321C43          STA   BYT16+4
4214   2629 321D43          STA   BYT16+5
4215   262C 321E43          STA   BYT16+6
4216   262F 321F43          STA   BYT16+7
4217   2632 E1              POP   H         ;ADJUST STACK
4218                ; CONVERT BINARY INDUCT AND DVT IN PACKAGE TCB INTO ASCII AND STORE
4219                ; INTO BUFFER BYT16
4220   2633 2AB442  DABO:   LHLD  TCBP      ;H,L= PACK TCB LOC
4221   2636 110900          LXI   D,9
4222   2639 19              DAD   D         ;H,L= JINDU LOC IN TCB
4223   263A 7E              MOV   A,M       ;A= BINARY JINDU VALUE
4224   263B CD010E          CALL  ECDIG     ;CONVERT TO ASCII
4225   263E 3A7E42          LDA   XYZ1
4226   2641 322043          STA   BYT16+8   ;STORE ASCII 10'S
4227   2644 3A7F42          LDA   XYZ0
4228   2647 322143          STA   BYT16+9   ;STORE ASCII 1'S
4229   264A 2AB442          LHLD  TCBP      ;H,L= PACK TCB LOC
4230   264D 110B00          LXI   D,11
4231   2650 19              DAD   D         ;H,L= JDV IN TCB LOC
4232   2651 7E              MOV   A,M       ;A= BINARY OUT$
4233   2652 CD010E          CALL  ECDIG     ;CONVERT TO ASCII
4234   2655 3A7D42          LDA   XYZ2
4235   2658 322243          STA   BYT16+10  ;STORE ASCII 100'S
4236   265B 3A7E42          LDA   XYZ1
4237   265E 322343          STA   BYT16+11  ;STORE ASCII 10'S
4238   2661 3A7F42          LDA   XYZ0
4239   2664 322443          STA   BYT16+12  ;STORE ASCII 1'S
4240   2667 3E53            MVI   A,'S'     ;A=ASCII S
4241   2669 321943          STA   BYT16+1   ;STORE
4242   266C CD7D1A          CALL  DSLD      ;LOAD BYT16 INTO FS
4243                ; RETURN PACKAGE TCB TO FS
4244                ;
4245   266F 2AB442  RTFS:   LHLD  TCBP      ;H,L= TCB LOC
```

```
4246  2672 CD4F11          CALL  PUTFS     ;RET TCB TO FS
4247  2675 C9              RET             ;RET TO CALLER
4248                ;
4249                ;EVENT TO RESET DIVERT AND SCHEDULE ANY FURTHER RESETS WITH
4250                ;  SAME PPI IF REQUIRED.
4251                ;
4252                ;INPUT: D,E POINTS TO BYTE 8 IN TCB, TCBP, TREE PARAMETERS.
4253                ;ALL REGISTERS EFFECTED.
4254                ;
4255  2676 EB       DVRS:   XCHG                    ;POINT H,L TO BYTE 9
4256  2677 23               INX     H               ;  IN TCB
4257  2678 4E               MOV     C,M             ;C=JDV
4258  2679 34               INR     M               ;INC JDV IN TCB FOR NEXT DV
4259  267A 23               INX     H               ;POINT H,L TO BYTE 10 IN TCB
4260  267B 35               DCR     M               ;IF MORE DV'S
4261  267C CA9426           JZ      DV0RS           ;  TO RESET
4262                ;
4263                ;SCHEDULE DVRS AGAIN TO RESET MORE DV'S
4264                ;
4265  267F C5               PUSH    B               ;SAVE C=JDV
4266  2680 23               INX     H               ;POINT H,L TO BYTE 11 IN TCB
4267  2681 5E               MOV     E,M             ;D,E
4268  2682 23               INX     H               ;  = UDBT
4269  2683 56               MOV     D,M             ;  IN TCB
4270  2684 EB               XCHG                    ;SET UDBT
4271  2685 229942           SHLD    UDBT            ;  FOR ENDVA
4272  2688 21FBFF           LXI     H,0FFFBH        ;POINT H,L TO BYTE 12-5=7
4273  268B 19               DAD     D               ;  IN TCB
4274  268C 0C               INR     C               ;C= NEXT JDV
4275  268D CD1D29           CALL    ENDVA           ;SCHEDULE FURTHER DV RESETS
4276  2690 C1               POP     B               ;C=JDV
4277  2691 C39A26           JMP     DORS
4278                ;
4279                ;NO MORE DV'S TO RESET FOR THIS TASK
4280                ;
4281  2694 2AB442   DV0RS:  LHLD    TCBP            ;RETURN TCB TO
4282  2697 CD4F11           CALL    PUTFS           ;  FREE STORAGE
4283                ;
4284                ;OUTPUT DV RESET SIGNAL
4285                ;
4286  269A 3A033D   DORS:   LDA     NINDU           ;C
4287  269D 81               ADD     C               ;  =NINDU+JDV
4288  269E 4F               MOV     C,A             ;  = DV OUTPUT CH #
4289  269F 0602             MVI     B,2             ;RESET
4290  26A1 CD0F11           CALL    OPUT            ;  DV # JDV
4291  26A4 C9               RET
4292                ;;
4293                ;DIVERT ACTUATION EVENT.
4294                ;
4295                ;INPUT:  TCBP, D,E POINTS TO BYTE 8 OF TCB, TREE PARAMETERS.
4296                ;OUTPUT: DVST.
4297                ;ALL REGISTERS EFFECTED, JUPDA EFFECTED.
4298                ;
4299  26A5 2AB642   DVACT:  LHLD  CTIME     ;FOR DVRCC
4300  26A8 229B42           SHLD  DVST      ;
4301  26AB 13               INX   D         ;POINT H,L TO BYTE 9
4302  26AC EB               XCHG            ;  IN TCB
4303  26AD 3A033D           LDA   NINDU     ;C
4304  26B0 86               ADD   M         ;  =NINDU+JDV
4305  26B1 4F               MOV   C,A       ;  = DIVERT OUTPUT CH #.
4306  26B2 23               INX   H         ;B
4307  26B3 46               MOV   B,M       ;  = DIVERT MODE
4308  26B4 C5               PUSH  B         ;SAVE PARAMETERS FOR OPUT
4309  26B5 05               DCR   B         ;IF B=1, I.E.,
4310  26B6 C2F926           JNZ   DDVNS     ;  DV TO BE SET
4311                ;
4312                ;CHECK LANE-FULL CONDITIONS LOCALLY
4313                ;
4314  26B9 3A2B42           LDA   ICMO              ;A= I/C FLAG
4315  26BC A7               ANA   A                 ;
4316  26BD C2E026           JNZ   DVLF0             ;BYPASS LANE FULL CHECK IF I/C MODE AS
4317                                                ;CHECK HAS ALREADY BEEN MADE AT PEVIOUS
4318                                                ;UPDATE PHOTO
4319  26C0 3A3C3D           LDA   LFO       ;IF LANE-FULL-
4320  26C3 E602             ANI   02H       ;  DV-INHIBIT
4321  26C5 CAE026           JZ    DVLF0     ;  DESIRED
4322  26C8 E5               PUSH  H         ;SAVE POINTER TO TCB BYTE 10
4323  26C9 2B               DCX   H         ;POINT H,L TO BYTE 9 IN TCB
4324  26CA 7E               MOV   A,M       ;A=JDV
4325  26CB 47               MOV   B,A       ;SAVE JDV IN B
4326  26CC CD250D           CALL  CPLFM     ;IF DV LANE FULL THEN
4327  26CF A6               ANA   M         ;  CLEAR FLAG Z
4328  26D0 E1               POP   H         ;POINT H,L TO BYTE 10 IN TCB
4329  26D1 CAE026           JZ    DVLF0     ;IF LANE FULL
4330  26D4 48               MOV   C,B               ;C=JDV
4331  26D5 CD0427           CALL  DVRCC             ;SEND PACKAGE TO LFEDV(LANE FULL ERR DVT
4332  26D8 2AB442           LHLD  TCBP              ;RETURN PACKAGE TCB TO FS
```

```
4333   26DB CD4F11            CALL   PUTFS
4334   26DE C1                POP    B                ;ADJUST STACK
4335   26DF C9                RET                     ;RET
4336                  ;
4337                  ;IF DVTRS=2 THEN SCHEDULE RESET OF DV # JDV
4338                  ;
4339   26E0 3A383D    DVLF0:  LDA    DVTRS     ;IF DVTRS=2,
4340   26E3 D602              SUI    2         ;  I.E., FIXED-LE-
4341   26E5 C2F926            JNZ    DDVNS     ;  DV-PULSE OPTION
4342   26E8 34                INR    M         ;DV MODE IN TCB = 2 TO RESET
4343   26E9 11FDFF            LXI    D,0FFFDH  ;POINT D,E TO
4344   26EC 19                DAD    D         ;  BYTE 7
4345   26ED EB                XCHG             ;  IN TCB
4346   26EE 0E01              MVI    C,1       ;SET PARAMETERS FOR ENTCB
4347   26F0 210840            LXI    H,DVH     ;  ( DVH(1)=DVH )
4348   26F3 CD2F29            CALL   ENTCB     ;SCHEDULE DV RESET
4349   26F6 C3FF26            JMP    DOPUT
4350                 ;
4351                 ;OUTPUT DV SIGNAL AND EXIT
4352                 ;
4353   26F9 2AB442    DDVNS:  LHLD   TCBP      ;RETURN FREE
4354   26FC CD4F11            CALL   PUTFS     ;  STORAGE
4355   26FF C1        DOPUT:  POP    B         ;ACTUATE
4356   2700 CD0F11            CALL   OPUT      ;  DV
4357   2703 C9                RET
4358                 ;;
4359                 ;CALLED BY DVACT FOR PACKAGE RECIRCULATION.
4360                 ;
4361                 ;INPUT: C=JDV (DIVERT #), NDVT, TREE PARAMETERS, DVST= CTIME AT DIVERT.
4362                 ;OUTPUT: UDBT= UD-BLOCKING TIME.
4363                 ;ALL REGISTERS EFFECTED, CTIME EFFECTED.
4364                 ;
4365                 DVRCC:
4366                 ;
4367                 ;SCHEDULE RESETS FOR DIVERTS # JDV TO # LAST FOR THIS UD
4368                 ;
4369   2704 0D                DCR    C         ;C=JDV-1
4370   2705 0600              MVI    B,0       ;B,C=JDV-1
4371   2707 2A3942            LHLD   DVT       ;H,L POINTS TO TIME IN
4372   270A 09                DAD    B         ;  DVT TABLE
4373   270B 09                DAD    B         ;
4374   270C 5E                MOV    E,M       ;D,E= TIME IN DVT
4375   270D 23                INX    H         ;
4376   270E 56                MOV    D,M       ;
4377   270F 2A9B42            LHLD   DVST      ;H,L= CTIME AT DIVERT
4378   2712 7D                MOV    A,L       ;H,L=CTIME- TIME IN DVT FOR
4379   2713 93                SUB    E         ;  JDV=
4380   2714 6F                MOV    L,A       ;  LAST UPDA BLOCKING
4381   2715 7C                MOV    A,H       ;  TIME
4382   2716 9A                SBB    D         ;
4383   2717 67                MOV    H,A       ;
4384   2718 229942            SHLD   UDBT      ;SAVE IN UDBT FOR SDVRS
4385                 ;
4386   271B 3AC042            LDA    MUD       ;A= LAST UD # FOR THIS PPI
4387   271E 5F                MOV    E,A       ;D,E
4388   271F 1600              MVI    D,0       ;  = LAST UD #
4389   2721 2A3D42            LHLD   LDVDB     ;POINT H,L TO LDV OF
4390   2724 19                DAD    D         ;  LAST UD
4391   2725 2B                DCX    H         ;  FOR THIS PPI
4392   2726 79                MOV    A,C       ;A=JDV-1
4393   2727 2B        DPVUD:  DCX    H         ;POINT H,L TO PREVIOUS LDV
4394   2728 1D                DCR    E         ;E = PREVIOUS UD #
4395   2729 BE                CMP    M         ;DOUNTIL JDV-1 >= LDV,
4396   272A DA2727            JC     DPVUD     ;  I.E., LDV < JDV
4397   272D 1C                INR    E         ;COMPUTE
4398   272E D5                PUSH   D         ;  UPDATE PHOTO #
4399   272F 23                INX    H         ;A= LDV
4400   2730 7E                MOV    A,M       ;  FOR THIS UD
4401   2731 0C                INR    C         ;C=JDV
4402   2732 CD7727            CALL   SDVRS     ;SCHEDULE THEIR RESETS
4403                 ;
4404                 ;SCHEDULE EVENT UDB1 FOR NEXT UPDATE PHOTO
4405                 ;
4406   2735 C1                POP    B         ;C = UPDATE PHOTO #
4407   2736 3A043D            LDA    NUPDA     ;IF ALREADY
4408   2739 A9                XRA    C         ;  PASSED LAST
4409   273A 47                MOV    B,A       ;  UPDATE PHOTO
4410   273B 3A313D            LDA    SIDEO     ;  AND NO SIDE-INDUCT
4411   273E E602              ANI    02H       ;  AUTO
4412   2740 B0                ORA    B         ;  RECIRCULATION
4413   2741 C8                RZ               ;  THEN RETURN
4414                 ;
4415   2742 CDCB0F            CALL   GETFS     ;GET FREE BLOCK
4416   2745 110D00            LXI    D,13      ;POINT TO BYTE 13
4417   2748 19                DAD    D         ;  IN TCB
4418   2749 3600              MVI    M,0       ;DEFAULT
4419   274B 2B                DCX    H         ;  IDENT IN TCB
```

```
4420  274C 3600          MVI   M,0        ;  = 0
4421  274E 2B            DCX   H          ;POINT TO BYTE 11 IN TCB
4422  274F 3A3D3D        LDA   LFEDV      ;ROUTE PACKAGE TO LANE-FULL
4423  2752 77            MOV   M,A        ;  ERROR DIVERT
4424  2753 2B            DCX   H          ;POINT TO BYTE 10 IN TCB
4425  2754 71            MOV   M,C        ;SET JUPDA IN TCB
4426  2755 34            INR   M          ;  FOR NEXT UPDATE PHOTO
4427  2756 3A043D        LDA   NUPDA      ;UPDATE PHOTO # NUPDA + 1
4428  2759 BE            CMP   M          ;  MEANS
4429  275A D25F27        JNC   DRTCB      ;  UPDATE PHOTO
4430  275D 3601          MVI   M,1        ;  # 1
4431  275F 2B      DRTCB: DCX   H          ;DEFAULT
4432  2760 3601          MVI   M,1        ;  JINDU IN TCB = 1
4433  2762 2B            DCX   H          ;POINT TO TCB BYTE 8
4434  2763 119424        LXI   D,UDB1     ;LET
4435  2766 72            MOV   M,D        ;  EVENT
4436  2767 2B            DCX   H          ;  BE
4437  2768 73            MOV   M,E        ;  UDB1
4438  2769 EB            XCHG             ;POINT D,E TO BYTE 7 IN TCB
4439  276A 2A9942        LHLD  UDBT       ;SET CTIME FOR
4440  276D 22B642        SHLD  CTIME      ;  ENTCB
4441  2770 2A3742        LHLD  UDT        ;SCHEDULE UDB1 FOR
4442  2773 CD2F29        CALL  ENTCB      ;  UD # C
4443  2776 C9            RET
4444                ;;
4445                ;SCHEDULE RESETS FOR DIVERTS # C TO # A,
4446                ;  ALL FOR CURRENT PPI.
4447                ;  CALLED BY UDBPC, SCHDV OR DVRCC.
4448                ;
4449                ;INPUT: UDBT= LAST UPDA BLOCKING TIME, TREE PARAMETERS.
4450                ;ALL REGISTERS EFFECTED.
4451                ;
4452  2777 91       SDVRS: SUB   C          ;RETURN
4453  2778 D8              RC               ;  IF NO DIVERT TO RESET
4454                ;
4455  2779 3C              INR   A          ;A= # DV'S TO RESET
4456  277A 47              MOV   B,A        ;B= # DIVERTS TO BE RESET
4457  277B CDCB0F          CALL  GETFS      ;GET FREE BLOCK
4458  277E 110C00          LXI   D,12       ;POINT H,L TO BYTE 12
4459  2781 19              DAD   D          ;  IN TCB
4460  2782 EB              XCHG             ;D,E
4461  2783 2A9942          LHLD  UDBT       ;  =
4462  2786 EB              XCHG             ;  UDBT
4463  2787 72              MOV   M,D        ;SET UDBT
4464  2788 2B              DCX   H          ;  IN
4465  2789 73              MOV   M,E        ;  TCB
4466  278A 2B              DCX   H          ;POINT H,L TO BYTE 10 IN TCB
4467  278B 70              MOV   M,B        ;STORE # DV'S TO RESET
4468  278C 2B              DCX   H          ;SET # OF 1ST DV
4469  278D 71              MOV   M,C        ;  TO RESET
4470  278E 2B              DCX   H          ;POINT H,L TO BYTE 8 IN TCB
4471  278F 117626          LXI   D,DVRS     ;LET
4472  2792 72              MOV   M,D        ;  EVENT
4473  2793 2B              DCX   H          ;  BE
4474  2794 73              MOV   M,E        ;  DVRS
4475  2795 CD1D29          CALL  ENDVA      ;SCHEDULE DV RESETS
4476  2798 C9              RET
4477                ;;
4478                ;POINT H,L TO PPDB( INDUCT #).
4479                ;
4480                ;INPUT: A= INDUCT # .
4481                ;REGISTERS A, D, E EFFECTED.
4482                ;
4483  2799 3D       CPPPP: DCR   A          ;A = INDUCT # - 1
4484  279A 87              ADD   A          ;A = ( INDUCT # - 1 ) * 2
4485  279B 2A4342          LHLD  PPDB       ;POINT H,L TO PPDB
4486  279E CD1B0D          CALL  AD16A      ;ADD ( INDUCT # * 32 )
4487  27A1 C9              RET
4488                ;
4489                ;
4490                ;COMPUTE RSLDB POINTER.
4491                ;
4492                ;INPUT: C= INDUCT #.
4493                ;OUTPUT: HL POINTS TO RLSDB( INDUCT # ).
4494                ;REGISTER B ALSO EFFECTED.
4495                ;
4496  27A2 2A4F42   CPRLP: LHLD  RLSDB      ;POINT H,L TO RLSDB START
4497  27A5 0600            MVI   B,0        ;BC
4498  27A7 0D              DCR   C          ;  = INDUCT # - 1
4499  27A8 09              DAD   B          ;POINT
4500  27A9 09              DAD   B          ;  H,L
4501  27AA 09              DAD   B          ;  TO
4502  27AB 09              DAD   B          ;  RLSDB(JINDU)
4503  27AC 0C              INR   C          ;RESTORE C = INDUCT #
4504  27AD C9              RET
4505                ;;.
4506                ;REMOVE THE NEXT ITEM FROM THE FUTURE EVENT TREE
```

```
4507                    ;  AND UPDATE NXTIM.
4508                    ;
4509                    ;INPUT: TREE PARAMETERS, AVL TREE.
4510                    ;OUTPUT: H,L POINTS TO TCB DELETED.
4511                    ;ALL REGISTERS EFFECTED.
4512                    ;
4513  27AE 2ABA42       DEQUE:    LHLD  RLINK      ;GET ADDRESS OF ROOT
4514  27B1 EB                     XCHG                    ;SAVE ADDRESS OF ROOT
4515  27B2 210000                 LXI     H,0             ;SAVE THE SP VALUE
4516  27B5 39                     DAD     SP              ;FOR STACK PURGE
4517  27B6 22C642                 SHLD    STKSV
4518  27B9 21B842                 LXI     H,HEAD          ;STACK (HEAD,+)
4519  27BC E5                     PUSH    H
4520  27BD 2601                   MVI     H,01
4521  27BF E5           TEST:     PUSH    H               ;STACK DIRECTION
4522  27C0 EB                     XCHG                    ;NODE ADDRESS TO (H,L)
4523                              GTLLK   0,0             ;MOVE TO THE LEFT
4523                              IF      00000H      ;IS NODE ADDRESS IN (H,L)?
4523                              LHLD    00000H           ;NO-NODE POINTER TO (H,L)
4523                              ENDIF
4523  27C1 5E                     MOV     E,M             ;MOVE THE LEFT
4523  27C2 23                     INX     H               ;LINK INTO THE
4523  27C3 56                     MOV     D,M             ;(D,E) REGISTER
4523                              IF      00000H            ;IS IT SAVE LINK MODE?
4523                              XCHG                    ;YES - LINK TO (H,L)
4523                              SHLD    00000H            ;LINK TO MEMORY
4523                              ENDIF
4523
4524  27C4 2B                     DCX     H               ;RESTORE NODE ADDRESS
4525  27C5 7A                     MOV     A,D             ;HAVE WE FOUND THE
4526  27C6 B3                     ORA     E               ;LEFTMOST NODE IN TREE?
4527  27C7 CAD027                 JZ      DLTE            ;YES - DELETE IT
4528  27CA E5                     PUSH    H               ;STACK AS PARENT ADDRESS
4529  27CB 26FF                   MVI     H,0FFH          ;MARK DIRECTION LEFT
4530  27CD C3BF27                 JMP     TEST            ;GO LEFT AGAIN
4531                    ;
4532                    ; DELETE A RECORD FROM TREE
4533                    ;
4534  27D0 22C842       DLTE:     SHLD    QP              ;SAVE NODE ADDRESS
4535                              GTRLK   0,0             ;GET ITS RIGHT LINK
4535                              IF      00000H           ;IS NODE ADDRESS IN (H,L)?
4535                              LHLD    00000H           ;NO -.NODE POINTER TO (H,L)
4535                              ENDIF
4535  27D3 23                     INX     H               ;STEP POINTER TO
4535  27D4 23                     INX     H               ;THE RIGHT LINK
4535  27D5 5E                     MOV     E,M             ;MOVE THE RIGHT
4535  27D6 23                     INX     H               ;LINK INTO THE
4535  27D7 56                     MOV     D,M             ;(D,E) REGISTER
4535                              IF      00000H            ;IS IT SAVE LINK MODE?
4535                              XCHG                    ;YES - LINK TO (H,L)
4535                              SHLD    00000H            ;LINK TO MEMORY
4535                              ENDIF
4535
4536  27D8 E1                     POP     H               ;POP AND SAVE THE
4537  27D9 7C                     MOV     A,H             ;DIRECTION INTO NODE
4538  27DA E1                     POP     H               ;GET THE PARENT NODE
4539  27DB 22CA42                 SHLD    QX              ;SAVE ADDRESS OF PARENT
4540  27DE B7                     ORA     A               ;DIRECTION TO CONDITION CODE
4541  27DF F5                     PUSH    PSW             ;SAVE DIRECTION STATUS
4542  27E0 F21329                 JP      QRTY            ;PARENT IS TREE HEADER
4543                              PTLLK   0,0             ;RLINK(SON) TO LLINK(PARENT)
4543  0001              MCTEM     SET     1               ;ASSUME ADDRESS IN (H,L)
4543                              IF      00000H           ;TEST ASSUMPTION
4543                    MCTEM     SET     0               ;FALSE ASSUMPTION
4543                              ENDIF
4543                              IF      00000H           ;IS NEW LINK IN (D,E)?
4543                              IF      MCTEM           ;NO - BUT IS POINTER IN (H,L)?
4543                              XCHG                    ;YES - SAVE NODE POINTER
4543                              ENDIF
4543                              LHLD    00000H           ;NEW LINK TO (H,L)
4543                              XCHG                    ;NEW LINK TO (D,E)
4543                              ENDIF
4543                              IF      00000H           ;IS NODE ADDRESS IN (H,L)?
4543                              LHLD    00000H           ;NO - NODE POINTER TO (H,L)
4543                              ENDIF
4543  27E3 73                     MOV     M,E             ;MOVE THE NEW
4543  27E4 23                     INX     H               ;LEFT LINK VALUE
4543  27E5 72                     MOV     M,D             ;INTO THE NODE
4543
4544  27E6 7A           QRTX:     MOV     A,D             ;IF RLINK(SON) IS ZERO THE
4545  27E7 B3                     ORA     E               ;PARENT IS THE POST ORDER
4546  27E8 CAF427                 JZ      SETTM           ;SUCCESSOR OF TASK NODE
4547  27EB EB           FIND:     XCHG                    ;FIND THE POST ORDER
4548                              GTLLK   0,0             ;SUCCESSOR OF THE
4548                              IF      00000H      ;IS NODE ADDRESS IN (H,L)?
4548                              LHLD    00000H           ;NO-NODE POINTER TO (H,L)
4548                              ENDIF
4548  27EC 5E                     MOV     E,M             ;MOVE THE LEFT
```

```
4548  27ED 23              INX   H              ;LINK INTO THE
4548  27EE 56              MOV   D,M            ;(D,E) REGISTER
4548                       IF    00000H           ;IS IT SAVE LINK MODE?
4548                       XCHG                 ;YES - LINK TO (H,L)
4548                       SHLD  00000H           ;LINK TO MEMORY
4548                       ENDIF
4548
4549  27EF 7A              MOV    A,D           ;SELECTED NODE
4550  27F0 B3              ORA    E
4551  27F1 C2EB27          JNZ    FIND
4552  27F4 110400  SETTM:  LXI    D,4           ;TIME OFFSET LESS ONE TO (D,E)
4553  27F7 19              DAD    D             ;POINT TO NEXT EVENT TIME
4554  27F8 5E              MOV    E,M           ;GET THE TIME OF
4555  27F9 23              INX    H             ;THE NEXT EVENT
4556  27FA 56              MOV    D,M
4557  27FB EB              XCHG                 ;TIME TO (H,L)
4558  27FC 22BD42          SHLD   NXTIM         ;UPDATE NEXT EVENT TIME
4559  27FF F1              POP    PSW           ;RECOVER ENTRY DIRECTION
4560  2800 2ACA42  QLOOP:  LHLD   QX            ;POINT TO PARENT NODE
4561  2803 B7      QLRP:   ORA    A             ;ENTRY MODE TO FLAGS
4562  2804 F2F528          JP     QXIT          ;PARENT IS TREE HEADER
4563                       GTBAL  0,0           ;GET BALANCE FACTOR OF PARENT
4563                       IF     00000H          ;IS NODE ADDRESS IN (H,L)?
4563                       LHLD   00000H          ;NO - NODE POINTER TO (H,L)
4563                       ENDIF
4563  2807 23              INX    H             ;STEP POINTER TO
4563  2808 23              INX    H             ;THE BALANCE BYTE
4563  2809 23              INX    H
4563  280A 23              INX    H
4563  280B 7E              MOV    A,M           ;BALANCE CODE TO A-REG
4563                       IF     00000H          ;TEST IF SAVE NODE
4563                       STA    00000H          ;YES - CODE TO MEMORY
4563                       ENDIF
4563  280C B7              ORA    A             ;SET CONDITION CODES
4563
4564  280D FAC228          JM     BMNUS         ;PARENT WAS HEAVY TO LEFT
4565  2810 CACD28          JZ     BZERO         ;PARENT WAS BALANCED
4566                       GTRLK  QX,QT         ;PARENT WAS HEAVY TO RIGHT
4566                       IF     QX          ;IS NODE ADDRESS IN (H,L)?
4566  2813 2ACA42          LHLD   QX          ;NO - NODE POINTER TO (H,L)
4566                       ENDIF
4566  2816 23              INX    H             ;STEP POINTER TO
4566  2817 23              INX    H             ;THE RIGHT LINK
4566  2818 5E              MOV    E,M           ;MOVE THE RIGHT
4566  2819 23              INX    H             ;LINK INTO THE
4566  281A 56              MOV    D,M           ;(D,E) REGISTER
4566                       IF     QT          ;IS IT SAVE LINK MODE?
4566  281B EB              XCHG                 ;YES - LINK TO (H,L)
4566  281C 22CC42          SHLD   QT          ;LINK TO MEMORY
4566                       ENDIF
4566
4567                       GTLLK  0,0           ;GET THE BALANCE FACTOR OF THE
4567                       IF     00000H      ;IS NODE ADDRESS IN (H,L)?
4567                       LHLD   00000H          ;NO-NODE POINTER TO (H,L)
4567                       ENDIF
4567  281F 5E              MOV    E,M           ;MOVE THE LEFT
4567  2820 23              INX    H             ;LINK INTO THE
4567  2821 56              MOV    D,M           ;(D,E) REGISTER
4567                       IF     00000H          ;IS IT SAVE LINK MODE?
4567                       XCHG                 ;YES - LINK TO (H,L)
4567                       SHLD   00000H          ;LINK TO MEMORY
4567                       ENDIF
4567
4568  2822 2B              DCX    H             ;PARENTS RIGHT SUBTREE AFTER
4569                       GTBAL  0,0           ;GETTING THE LEFT LINK OF THE
4569                       IF     00000H          ;IS NODE ADDRESS IN (H,L)?
4569                       LHLD   00000H          ;NO - NODE POINTER TO (H,L)
4569                       ENDIF
4569  2823 23              INX    H             ;STEP POINTER TO
4569  2824 23              INX    H             ;THE BALANCE BYTE
4569  2825 23              INX    H
4569  2826 23              INX    H
4569  2827 7E              MOV    A,M           ;BALANCE CODE TO A-REG
4569                       IF     00000H          ;TEST IF SAVE NODE
4569                       STA    00000H          ;YES - CODE TO MEMORY
4569                       ENDIF
4569  2828 B7              ORA    A             ;SET CONDITION CODES
4569
4570  2829 CAD228          JZ     CASE3         ;SUBTREE ROOT NODE
4571  282C FA5028          JM     CASE2         ;RIGHT SUBTREE HEAVY ON LEFT
4572  282F 3600            MVI    M,0           ;RIGHT SUBTREE HEAVY ON RIGHT
4573                       PTRLK  QX,0          ;ROTATE SUBTREE INTO BALANCE
4573  0001         MCTEM   SET    1             ;ASSUME ADDRESS IN (H,L)
4573                       IF     QX          ;TEST ASSUMPTION
4573  0000         MCTEM   SET    0             ;FALSE ASSUMPTION
4573                       ENDIF
4573                       IF     00000H          ;IS NEW LINK IN (D,E)?
4573                       IF     MCTEM         ;NO - BUT IS POINTER IN (H,L)?
```

```
4573                            XCHG                    ;YES - SAVE NODE POINTER
4573                            ENDIF
4573                            LHLD  00000H             ;NEW LINK TO (H,L)
4573                            XCHG                    ;NEW LINK TO (D,E)
4573                            ENDIF
4573                            IF    QX              ;IS NODE ADDRESS IN (H,L)?
4573  2831 2ACA42               LHLD  QX              ;NO - NODE POINTER TO (H,L)
4573                            ENDIF
4573  2834 23                   INX   H                 ;NOW STEP TO THE
4573  2835 23                   INX   H                 ;RIGHT LINK BYTES
4573  2836 73                   MOV   M,E               ;MOVE THE NEW
4573  2837 23                   INX   H                 ;RIGHT LINK VALUE
4573  2838 72                   MOV   M,D               ;INTO THE NODE
4573
4574                            PTLLK  QT,QX            ;PARENT TO LLINK(SUBTREE ROOT)
4574  0001          MCTEM       SET   1                 ;ASSUME ADDRESS IN (H,L)
4574                            IF    QT              ;TEST ASSUMPTION
4574  0000          MCTEM       SET   0                 ;FALSE ASSUMPTION
4574                            ENDIF
4574                            IF    QX              ;IS NEW LINK IN (D,E)?
4574                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4574                            XCHG                    ;YES - SAVE NODE POINTER
4574                            ENDIF
4574  2839 2ACA42               LHLD  QX              ;NEW LINK TO (H,L)
4574  283C EB                   XCHG                    ;NEW LINK TO (D,E)
4574                            ENDIF
4574                            IF    QT              ;IS NODE ADDRESS IN (H,L)?
4574  283D 2ACC42               LHLD  QT              ;NO - NODE POINTER TO (H,L)
4574                            ENDIF
4574  2840 73                   MOV   M,E               ;MOVE THE NEW
4574  2841 23                   INX   H                 ;LEFT LINK VALUE
4574  2842 72                   MOV   M,D               ;INTO THE NODE
4574
4575  2843 AF                   XRA    A                ;THE ORIGINAL PARENT IS
4576  2844 EB                   XCHG                    ;NOW BALANCED BUT THE ROOT
4577                            PTBAL  0,0              ;OF RIGHT SUBTREE HAS REPLACED IT
4577                            IF    00000H             ;IS FACTOR IN A-REG?
4577                            LDA   00000H             ;NO - BALANCE TO A-REG
4577                            ENDIF
4577                            IF    00000H             ;IS NODE ADDRESS IN (H,L)
4577                            LHLD  00000H             ;NO - NODE POINTER TO (H,L)
4577                            ENDIF
4577  2845 23                   INX   H                 ;STEP POINTER TO
4577  2846 23                   INX   H                 ;THE BALANCE BYTE
4577  2847 23                   INX   H
4577  2848 23                   INX   H
4577  2849 77                   MOV   M,A               ;BALANCE CODE TO NODE
4577
4578  284A CDF928               CALL   CSUBR            ;SO ADJUST THE GRANDPARENT
4579  284D C30028               JMP    QLOOP            ;TREE HEIGHT CHANGED, SO GO ON
4580                      ;
4581  2850 EB           CASE2:  XCHG                    ;SAVE THE LEFT LINK OF
4582  2851 22CE42               SHLD   QW               ;THE SUBTREE ROOT NODE
4583                            GTRLK  0,0              ;PLACE THIS NODE INTO PARENTS
4583                            IF    00000H             ;IS NODE ADDRESS IN (H,L)?
4583                            LHLD  00000H             ;NO - NODE POINTER TO (H,L)
4583                            ENDIF
4583  2854 23                   INX   H                 ;STEP POINTER TO
4583  2855 23                   INX   H                 ;THE RIGHT LINK
4583  2856 5E                   MOV   E,M               ;MOVE THE RIGHT
4583  2857 23                   INX   H                 ;LINK INTO THE
4583  2858 56                   MOV   D,M               ;(D,E) REGISTER
4583                            IF    00000H               ;IS IT SAVE LINK MODE?
4583                            XCHG                    ;YES - LINK TO (H,L)
4583                            SHLD  00000H               ;LINK TO MEMORY
4583                            ENDIF
4583
4584                            PTLLK  QT,0             ;TREE POSITION WITH PARENT AS
4584  0001          MCTEM       SET   1                 ;ASSUME ADDRESS IN (H,L)
4584                            IF    QT              ;TEST ASSUMPTION
4584  0000          MCTEM       SET   0                 ;FALSE ASSUMPTION
4584                            ENDIF
4584                            IF    00000H             ;IS NEW LINK IN (D,E)?
4584                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4584                            XCHG                    ;YES - SAVE NODE POINTER
4584                            ENDIF
4584                            LHLD  00000H             ;NEW LINK TO (H,L)
4584                            XCHG                    ;NEW LINK TO (D,E)
4584                            ENDIF
4584                            IF    QT              ;IS NODE ADDRESS IN (H,L)?
4584  2859 2ACC42               LHLD  QT              ;NO - NODE POINTER TO (H,L)
4584                            ENDIF
4584  285C 73                   MOV   M,E               ;MOVE THE NEW
4584  285D 23                   INX   H                 ;LEFT LINK VALUE
4584  285E 72                   MOV   M,D               ;INTO THE NODE
4584
4585                            PTRLK  QW,QT            ;ITS LEFT SUBTREE AND ITS OLD LEFT
```

```
4585   0001          MCTEM   SET   1              ;ASSUME ADDRESS IN (H,L)
4585                         IF    QW           ;TEST ASSUMPTION
4585   0000          MCTEM   SET   0              ;FALSE ASSUMPTION
4585                         ENDIF
4585                         IF    QT           ;IS NEW LINK IN (D,E)?
4585                         IF    MCTEM          ;NO - BUT IS POINTER IN (H,L)?
4585                         XCHG                 ;YES - SAVE NODE POINTER
4585                         ENDIF
4585   285F 2ACC42           LHLD  QT           ;NEW LINK TO (H,L)
4585   2862 EB               XCHG                 ;NEW LINK TO (D,E)
4585                         ENDIF
4585                         IF    QW           ;IS NODE ADDRESS IN (H,L)?
4585   2863 2ACE42           LHLD  QW           ;NO - NODE POINTER TO (H,L)
4585                         ENDIF
4585   2866 23               INX   H              ;NOW STEP TO THE
4585   2867 23               INX   H              ;RIGHT LINK BYTES
4585   2868 73               MOV   M,E            ;MOVE THE NEW
4585   2869 23               INX   H              ;RIGHT LINK VALUE
4585   286A 72               MOV   M,D            ;INTO THE NODE
4585
4586                         GTLLK  QW.0            ;SUBTREE AS PARENTS RIGHT SUBTREE
4586                         IF    QW      ;IS NODE ADDRESS IN (H,L)?
4586   286B 2ACE42           LHLD  QW           ;NO-NODE POINTER TO (H,L)
4586                         ENDIF
4586   286E 5E               MOV   E,M            ;MOVE THE LEFT
4586   286F 23               INX   H              ;LINK INTO THE
4586   2870 56               MOV   D,M            ;(D,E) REGISTER
4586                         IF    00000H           ;IS IT SAVE LINK MODE?
4586                         XCHG                 ;YES - LINK TO (H,L)
4586                         SHLD  00000H           ;LINK TO MEMORY
4586                         ENDIF
4586
4587                         PTRLK  QX.0            ;ITS RIGHT SUBTREE REPLACES THE
4587   0001          MCTEM   SET   1              ;ASSUME ADDRESS IN (H,L)
4587                         IF    QX           ;TEST ASSUMPTION
4587   0000          MCTEM   SET   0              ;FALSE ASSUMPTION
4587                         ENDIF
4587                         IF    00000H          ;IS NEW LINK IN (D,E)?
4587                         IF    MCTEM          ;NO - BUT IS POINTER IN (H,L)?
4587                         XCHG                 ;YES - SAVE NODE POINTER
4587                         ENDIF
4587                         LHLD  00000H          ;NEW LINK TO (H,L)
4587                         XCHG                 ;NEW LINK TO (D,E)
4587                         ENDIF
4587                         IF    QX           ;IS NODE ADDRESS IN (H,L)?
4587   2871 2ACA42           LHLD  QX           ;NO - NODE POINTER TO (H,L)
4587                         ENDIF
4587   2874 23               INX   H              ;NOW STEP TO THE
4587   2875 23               INX   H              ;RIGHT LINK BYTES
4587   2876 73               MOV   M,E            ;MOVE THE NEW
4587   2877 23               INX   H              ;RIGHT LINK VALUE
4587   2878 72               MOV   M,D            ;INTO THE NODE
4587
4588                         PTLLK  QW.QX           ;LEFT SUBTREE OF ORIGINAL RIGHT
4588   0001          MCTEM   SET   1              ;ASSUME ADDRESS IN (H,L)
4588                         IF    QW           ;TEST ASSUMPTION
4588   0000          MCTEM   SET   0              ;FALSE ASSUMPTION
4588                         ENDIF
4588                         IF    QX           ;IS NEW LINK IN (D,E)?
4588                         IF    MCTEM          ;NO - BUT IS POINTER IN (H,L)?
4588                         XCHG                 ;YES - SAVE NODE POINTER
4588                         ENDIF
4588   2879 2ACA42           LHLD  QX           ;NEW LINK TO (H,L)
4588   287C EB               XCHG                 ;NEW LINK TO (D,E)
4588                         ENDIF
4588                         IF    QW           ;IS NODE ADDRESS IN (H,L)?
4588   287D 2ACE42           LHLD  QW           ;NO - NODE POINTER TO (H,L)
4588                         ENDIF
4588   2880 73               MOV   M,E            ;MOVE THE NEW
4588   2881 23               INX   H              ;LEFT LINK VALUE
4588   2882 72               MOV   M,D            ;INTO THE NODE
4588
4589   2883 2B               DCX    H             ;SUBTREE OF THE PARENT
4590                         GTBAL  0,0           ;GET BALANCE FACTOR OF PARENTS REPLCMT
4590                         IF    00000H          ;IS NODE ADDRESS IN (H,L)?
4590                         LHLD  00000H          ;NO - NODE POINTER TO (H,L)
4590                         ENDIF
4590   2884 23               INX   H              ;STEP POINTER TO
4590   2885 23               INX   H              ;THE BALANCE BYTE
4590   2886 23               INX   H
4590   2887 23               INX   H
4590   2888 7E               MOV   A,M            ;BALANCE CODE TO A-REG
4590                         IF    00000H          ;TEST IF SAVE NODE
4590                         STA   00000H          ;YES - CODE TO MEMORY
4590                         ENDIF
4590   2889 B7               ORA   A              ;SET CONDITION CODES
4590
```

```
4591   288A CAB628          JZ     CSSZ            ;BALANCED NODE
4592   288D F2B428          JP     CSSP            ;HEAVY ON THE RIGHT
4593   2890 AF              XRA    A               ;NEW RIGHT SUBTREE OF PARENT
4594                        PTBAL  QX,0            ;BRINGS IT INTO BALANCE BUT
4594                        IF     00000H           ;IS FACTOR IN A-REG?
4594                        LDA    00000H           ;NO - BALANCE TO A-REG
4594                        ENDIF
4594                        IF     QX            ;IS NODE ADDRESS IN (H,L)
4594   2891 2ACA42          LHLD   QX            ;NO - NODE POINTER TO (H,L)
4594                        ENDIF
4594   2894 23              INX    H               ;STEP POINTER TO
4594   2895 23              INX    H               ;THE BALANCE BYTE
4594   2896 23              INX    H
4594   2897 23              INX    H
4594   2898 77              MOV    M,A             ;BALANCE CODE TO NODE
4594
4595   2899 3C              INR    A               ;ROOT OF ORIGINAL RIGHT SUBTREE
4596                 CSSX:   PTBAL  QT,0            ;IS NOW HEAVY TO THE RIGHT
4596                        IF     00000H           ;IS FACTOR IN A-REG?
4596                        LDA    00000H           ;NO - BALANCE TO A-REG
4596                        ENDIF
4596                        IF     QT            ;IS NODE ADDRESS IN (H,L)
4596   289A 2ACC42          LHLD   QT            ;NO - NODE POINTER TO (H,L)
4596                        ENDIF
4596   289D 23              INX    H               ;STEP POINTER TO
4596   289E 23              INX    H               ;THE BALANCE BYTE
4596   289F 23              INX    H
4596   28A0 23              INX    H
4596   28A1 77              MOV    M,A             ;BALANCE CODE TO NODE
4596
4597   28A2 2ACE42          LHLD   QW              ;SAVE ADDRESS OF PARENTS
4598   28A5 22CC42          SHLD   QT              ;REPLACEMENT NODE AND
4599   28A8 AF              XRA    A               ;ALSO MARK IT AS A
4600                        PTBAL  0,0             ;BALANCED SUBTREE AND
4600                        IF     00000H           ;IS FACTOR IN A-REG?
4600                        LDA    00000H           ;NO - BALANCE TO A-REG
4600                        ENDIF
4600                        IF     00000H           ;IS NODE ADDRESS IN (H,L)
4600                        LHLD   00000H           ;NO - NODE POINTER TO (H,L)
4600                        ENDIF
4600   28A9 23              INX    H               ;STEP POINTER TO
4600   28AA 23              INX    H               ;THE BALANCE BYTE
4600   28AB 23              INX    H
4600   28AC 23              INX    H
4600   28AD 77              MOV    M,A             ;BALANCE CODE TO NODE
4600
4601   28AE CDF928          CALL   CSUBR           ;ADJUST THE GRANDPARENT NODE LINK
4602   28B1 C30028          JMP    QLOOP           ;TREE HEIGHT CHANGED - SO GO ON
4603                ;
4604   28B4 3EFF     CSSP:  MVI    A,0FFH          ;PARENT WILL BE LEFT HEAVY
4605                 CSSZ:  PTBAL  QX,0            ;SET PARENT BALANCE (0,-)
4605                        IF     00000H           ;IS FACTOR IN A-REG?
4605                        LDA    00000H           ;NO - BALANCE TO A-REG
4605                        ENDIF
4605                        IF     QX            ;IS NODE ADDRESS IN (H,L)
4605   28B6 2ACA42          LHLD   QX            ;NO - NODE POINTER TO (H,L)
4605                        ENDIF
4605   28B9 23              INX    H               ;STEP POINTER TO
4605   28BA 23              INX    H               ;THE BALANCE BYTE
4605   28BB 23              INX    H
4605   28BC 23              INX    H
4605   28BD 77              MOV    M,A             ;BALANCE CODE TO NODE
4605
4606   28BE AF              XRA    A               ;BUT ORIGINAL RIGHT SUBTREE
4607   28BF C39A28          JMP    CSSX            ;OF PARENT WILL BE BALANCED
4608                ;
4609   28C2 AF       BMNUS: XRA    A               ;REMOVED ITEM ON LEFT SO
4610   28C3 77              MOV    M,A             ;PARENT IS NOW BALANCED
4611   28C4 E1              POP    H               ;BUT WE MAY HAVE UPSET
4612   28C5 7C              MOV    A,H             ;BALANCES ABOVE THIS NODE
4613   28C6 E1              POP    H               ;SO BACKTRACK UP THE TREE
4614   28C7 22CA42          SHLD   QX              ;SET NEW PARENT ADDRESS
4615   28CA C30328          JMP    QLRP            ;AND GO ON
4616                ;
4617   28CD 3C       BZERO: INR    A               ;PARENT NOW HEAVY ON RIGHT
4618   28CE 77              MOV    M,A             ;BUT SUBTREE HEIGHT UNCHANGED
4619   28CF C3F128          JMP    QDONE           ;SO THE TOTAL TREE IS NOW OK
4620                ;
4621   28D2 3EFF     CASE3: MVI    A,0FFH          ;PARENTS REPLACEMENT WILL
4622                        PTBAL  QT,0            ;BE HEAVY TO THE LEFT
4622                        IF     00000H           ;IS FACTOR IN A-REG?
4622                        LDA    00000H           ;NO - BALANCE TO A-REG
4622                        ENDIF
4622                        IF     QT            ;IS NODE ADDRESS IN (H,L)
4622   28D4 2ACC42          LHLD   QT            ;NO - NODE POINTER TO (H,L)
4622                        ENDIF
4622   28D7 23              INX    H               ;STEP POINTER TO
```

```
4622  28D8 23              INX   H             ;THE BALANCE BYTE
4622  28D9 23              INX   H
4622  28DA 23              INX   H
4622  28DB 77              MOV   M,A           ;BALANCE CODE TO NODE
4622
4623                       PTRLK  QX,0         ;SUBTREE ROOT LLINK TO PARENT RLINK
4623  0001          MCTEM  SET   1             ;ASSUME ADDRESS IN (H,L)
4623                       IF    QX           ;TEST ASSUMPTION
4623  0000          MCTEM  SET   0             ;FALSE ASSUMPTION
4623                       ENDIF
4623                       IF    00000H        ;IS NEW LINK IN (D,E)?
4623                       IF    MCTEM         ;NO - BUT IS POINTER IN (H,L)?
4623                       XCHG                ;YES - SAVE NODE POINTER
4623                       ENDIF
4623                       LHLD  00000H        ;NEW LINK TO (H,L)
4623                       XCHG                ;NEW LINK TO (D,E)
4623                       ENDIF
4623                       IF    QX           ;IS NODE ADDRESS IN (H,L)?
4623  28DC 2ACA42          LHLD  QX           ;NO - NODE POINTER TO (H,L)
4623                       ENDIF
4623  28DF 23              INX   H             ;NOW STEP TO THE
4623  28E0 23              INX   H             ;RIGHT LINK BYTES
4623  28E1 73              MOV   M,E           ;MOVE THE NEW
4623  28E2 23              INX   H             ;RIGHT LINK VALUE
4623  28E3 72              MOV   M,D           ;INTO THE NODE
4623
4624                       PTLLK  QT,QX        ;ADD PARENT AS LEFT SUBTREE
4624  0001          MCTEM  SET   1             ;ASSUME ADDRESS IN (H,L)
4624                       IF    QT           ;TEST ASSUMPTION
4624  0000          MCTEM  SET   0             ;FALSE ASSUMPTION
4624                       ENDIF
4624                       IF    QX           ;IS NEW LINK IN (D,E)?
4624                       IF    MCTEM         ;NO - BUT IS POINTER IN (H,L)?
4624                       XCHG                ;YES - SAVE NODE POINTER
4624                       ENDIF
4624  28E4 2ACA42          LHLD  QX           ;NEW LINK TO (H,L)
4624  28E7 EB              XCHG                ;NEW LINK TO (D,E)
4624                       ENDIF
4624                       IF    QT           ;IS NODE ADDRESS IN (H,L)?
4624  28E8 2ACC42          LHLD  QT           ;NO - NODE POINTER TO (H,L)
4624                       ENDIF
4624  28EB 73              MOV   M,E           ;MOVE THE NEW
4624  28EC 23              INX   H             ;LEFT LINK VALUE
4624  28ED 72              MOV   M,D           ;INTO THE NODE
4624
4625  28EE CDF928          CALL   CSUBR        ;LINK NEW PARENT TO TREE
4626                ;
4627  28F1 2AC642   QDONE:  LHLD   STKSV        ;PURGE THE STACK BACK
4628  28F4 F9              SPHL                ;TO ITS ENTRY VALUE
4629  28F5 2AC842   QXIT:   LHLD   QP           ;ADDRESS OF SELECTED NODE
4630  28F8 C9              RET                 ;RETURN
4631                ;
4632  28F9 2ACC42   CSUBR:  LHLD   QT           ;GET ADDRESS OF OLD PARENT
4633  28FC EB              XCHG                ;NODE REPLACEMENT AND
4634  28FD E1              POP    H            ;POP THE ROUTINE RETURN THEN
4635  28FE E3              XTHL                ;GET GRANDPARENTS EXIT DIRECTION
4636  28FF 7C              MOV    A,H          ;SAVE IT
4637  2900 E1              POP    H            ;THE RETURN AGAIN
4638  2901 E3              XTHL                ;GET GRANDPARENT ADDRESS
4639  2902 22CA42          SHLD   QX           ;AND SAVE IT
4640  2905 B7              ORA    A            ;DIRECTION TO CONDITION CODE
4641  2906 FA0F29          JM     CS225        ;WE CAME FROM THE LEFT
4642                       PTRLK  0,0          ;ADJUST THE RIGHT LINK
4642  0001          MCTEM  SET   1             ;ASSUME ADDRESS IN (H,L)
4642                       IF    00000H        ;TEST ASSUMPTION
4642                MCTEM  SET   0             ;FALSE ASSUMPTION
4642                       ENDIF
4642                       IF    00000H        ;IS NEW LINK IN (D,E)?
4642                       IF    MCTEM         ;NO - BUT IS POINTER IN (H,L)?
4642                       XCHG                ;YES - SAVE NODE POINTER
4642                       ENDIF
4642                       LHLD  00000H        ;NEW LINK TO (H,L)
4642                       XCHG                ;NEW LINK TO (D,E)
4642                       ENDIF
4642                       IF    00000H        ;IS NODE ADDRESS IN (H,L)?
4642                       LHLD  00000H        ;NO - NODE POINTER TO (H,L)
4642                       ENDIF
4642  2909 23              INX   H             ;NOW STEP TO THE
4642  290A 23              INX   H             ;RIGHT LINK BYTES
4642  290B 73              MOV   M,E           ;MOVE THE NEW
4642  290C 23              INX   H             ;RIGHT LINK VALUE
4642  290D 72              MOV   M,D           ;INTO THE NODE
4642
4643  290E C9              RET                 ;AND RETURN
4644                ;
4645                CS225:  PTLLK  0,0         ;ADJUST THE LEFT LINK
4645  0001          MCTEM  SET   1             ;ASSUME ADDRESS IN (H,L)
```

```
4645                            IF    00000H            ;TEST ASSUMPTION
4645                    MCTEM   SET   0                 ;FALSE ASSUMPTION
4645                            ENDIF
4645                            IF    00000H            ;IS NEW LINK IN (D,E)?
4645                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4645                            XCHG                    ;YES - SAVE NODE POINTER
4645                            ENDIF
4645                            LHLD  00000H            ;NEW LINK TO (H,L)
4645                            XCHG                    ;NEW LINK TO (D,E)
4645                            ENDIF
4645                            IF    00000H            ;IS NODE ADDRESS IN (H,L)?
4645                            LHLD  00000H            ;NO - NODE POINTER TO (H,L)
4645                            ENDIF
4645    290F 73                 MOV   M,E               ;MOVE THE NEW
4645    2910 23                 INX   H                 ;LEFT LINK VALUE
4645    2911 72                 MOV   M,D               ;INTO THE NODE
4645
4646    2912 C9                 RET                     ;AND RETURN
4647                    ;
4648                    QRTY:   PTRLK  0,0              ;ESTABLISH THE NEW ROOT
4648    0001            MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
4648                            IF    00000H            ;TEST ASSUMPTION
4648                    MCTEM   SET   0                 ;FALSE ASSUMPTION
4648                            ENDIF
4648                            IF    00000H            ;IS NEW LINK IN (D,E)?
4648                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4648                            XCHG                    ;YES - SAVE NODE POINTER
4648                            ENDIF
4648                            LHLD  00000H            ;NEW LINK TO (H,L)
4648                            XCHG                    ;NEW LINK TO (D,E)
4648                            ENDIF
4648                            IF    00000H            ;IS NODE ADDRESS IN (H,L)?
4648                            LHLD  00000H            ;NO - NODE POINTER TO (H,L)
4648                            ENDIF
4648    2913 23                 INX   H                 ;NOW STEP TO THE
4648    2914 23                 INX   H                 ;RIGHT LINK BYTES
4648    2915 73                 MOV   M,E               ;MOVE THE NEW
4648    2916 23                 INX   H                 ;RIGHT LINK VALUE
4648    2917 72                 MOV   M,D               ;INTO THE NODE
4648
4649    2918 2B                 DCX    H                ;NODE FOR THE TREE AND
4650    2919 2B                 DCX    H                ;ADJUST THE POINTERS SO
4651    291A C3E627             JMP    QRTX             ;THAT WE MAY CONTINUE
4652                    ;;
4653                    ;SCHEDULE DIVERT ACTUATION EVENT AT UDBT + DVT(C).
4654                    ;  CALLED BY SCDVA, DVRS, SDVRS.
4655                    ;
4656                    ;INPUT: H,L POINTS TO BYTE 7 IN TCB, TREE PARAMETERS, UDBT, C=JDV.
4657                    ;OUTPUT: EVENT IN TREE, UPDATED PPIDB.
4658                    ;ALL REGISTERS EFFECTED.
4659                    ;
4660    291D EB         ENDVA:  XCHG                    ;POINT D,E TO BYTE 7 IN TCB
4661    291E 2A3942             LHLD  DVT               ;POINT H,L TO DVT TABLE
4662    2921 0600               MVI   B,0               ;B,C=JDV
4663    2923 09                 DAD   B                 ;POINT H,L TO
4664    2924 09                 DAD   B                 ;  DVT(JDV)
4665    2925 2B                 DCX   H                 ;  MS BYTE
4666    2926 46                 MOV   B,M               ;B,C
4667    2927 2B                 DCX   H                 ;  =
4668    2928 4E                 MOV   C,M               ;  DVT(JDV)
4669    2929 2A9942             LHLD  UDBT              ;H,L=UDBT
4670    292C C33A29             JMP   ENTC1
4671                    ;
4672                    ;SCHEDULE EVENT AT CTIME + H,L(C).
4673                    ;
4674                    ;INPUT: D,E POINTS TO BYTE 7 IN TCB, H,L POINTS TO TIME
4675                    ;  TABLE, C ITS INDEX, TREE PARAMETERS.
4676                    ;OUTPUT: EVENT IN TREE.
4677                    ;ALL REGISTERS EFFECTED.
4678                    ;
4679    292F 0600       ENTCB:  MVI   B,0               ;B,C= INDEX
4680    2931 09                 DAD   B                 ;POINT H,L
4681    2932 09                 DAD   B                 ;  TO TIME
4682    2933 2B                 DCX   H                 ;  MS BYTE
4683    2934 46                 MOV   B,M               ;B,C
4684    2935 2B                 DCX   H                 ;  = TABLE
4685    2936 4E                 MOV   C,M               ;  TIME
4686    2937 2AB642             LHLD  CTIME             ;H,L= CTIME
4687                    ;
4688    293A 09         ENTC1:  DAD   B                 ;H,L= EVENT TIME
4689    293B EB                 XCHG                    ;D,E= EVTIM, POINT H,L TO
4690    293C 2B                 DCX   H                 ;  BYTE 6 IN TCB
4691    293D 72                 MOV   M,D               ;SET EVTIM
4692    293E 2B                 DCX   H                 ;  IN
4693    293F 73                 MOV   M,E               ;  TCB
4694    2940 11FBFF             LXI   D,0FFFBH          ;POINT H,L TO
4695    2943 19                 DAD   D                 ;  TCB
```

```
4696                    ;
4697                    ;
4698                    ;INSERT TCB INTO EVENT TREE & UPDATE PPIDB.
4699                    ;
4700                    ;INPUT: H,L POINTS TO TCB, TREE PARAMETERS FOR THIS PPI,
4701                    ;  PPIDP POINTS TO CTIME IN PPIDB (SEE EXEC, GTREP).
4702                    ;ALL REGISTERS EFFECTED.
4703                    ;
4704  2944 CD5629       ENQU2:   CALL  ENQUE      ;INSERT EVENT TCB
4705  2947 2AD842              LHLD  PPIDP      ;POINT D,E TO
4706  294A EB                  XCHG             ;  CTIME IN PPIDB
4707  294B 13                  INX   D          ;SKIP PPI CLOCK TIME SINCE
4708  294C 13                  INX   D          ;  IT MAY HAVE BEEN CHANGED BY PPIIN
4709  294D 21B842              LXI   H,HEAD     ;POINT H,L TO TREE HEADER
4710  2950 060E                MVI   B,14       ;UPDATE
4711  2952 CDCF10              CALL  MOVEB      ;  PPIDB
4712  2955 C9                  RET
4713                    ;;
4714                    ; ADD A NEW NODE TO THE FUTURE EVENT TREE
4715                    ;
4716  2956 EB           ENQUE:   XCHG                    ;SAVE NEW NODE ADDRESS
4717  2957 2ABA42              LHLD   RLINK             ;TEST IF THE
4718  295A 7C                  MOV    A,H               ;TREE IS EMPTY
4719  295B B5                  ORA    L
4720  295C CAE22B              JZ     TREMT             ;THE TREE IS EMPTY
4721  295F D5                  PUSH   D                 ;SAVE NODE ADDRESS
4722  2960 EB                  XCHG                     ;SAVE ADDRESS OF THE ROOT
4723  2961 21B842              LXI    H,HEAD            ;POINT TO THE TREE HEADER
4724  2964 22CC42              SHLD   QT                ;SET PARENT ADDRESS
4725  2967 EB                  XCHG                     ;GET BALANCE POINT NODE
4726  2968 22D042              SHLD   QS                ;AND SAVE IT
4727  296B AF                  XRA    A                 ;MARK EXIT DIRECTION
4728  296C 32D442              STA    QDIR              ;AS RIGHT AND SAVE IT
4729  296F 22C842              SHLD   QP                ;SET CURRENT NODE ADDRESS
4730  2972 E1                  POP    H                 ;RECOVER THE ADDRESS
4731  2973 E5                  PUSH   H                 ;OF THE NEW NODE
4732  2974 CDEE2B              CALL   NDCLR             ;CLEAR ITS LINKS & GET 'TIME'
4733  2977 2ABD42              LHLD   NXTIM             ;GET THE NEXT SCHEDULED
4734  297A EB                  XCHG                     ;EVENT TIME AND COMPARE
4735  297B 7B                  MOV    A,E               ;IT TO THE SCHEDULED TIME
4736  297C 95                  SUB    L                 ;IN THE NEW FUTURE EVENT
4737  297D 7A                  MOV    A,D               ;(NXTIME)-(NEW TIME)
4738  297E 9C                  SBB    H
4739                    ;         ADD    A                 ;DECISION BIT TO CONDITION CODE
4740  297F FA8529              JM     ENQ10             ;NEW EVENT IS NOT NEXT
4741  2982 22BD42              SHLD   NXTIM             ;UPDATE NEXT EVENT TIME
4742  2985 22D642       ENQ10:   SHLD   NEWTM           ;SAVE THIS EVENT TIME
4743  2988 2AC842              LHLD   QP                ;COMPARE THE NEW 'PPI'
4744  298B CDFD2B       ENQ2:    CALL   CMPTM            ;TIME WITH NODE VALUE
4745  298E F2CA29              JP     ENQ4              ;MOVE TO RIGHT IN SUBTREE
4746                    ENQ3:    GTLLK  QP,QW             ;MOVE TO THE LEFT
4746                             IF     QP       ;IS NODE ADDRESS IN (H,L)?
4746  2991 2AC842                LHLD   QP              ;NO-NODE POINTER TO (H,L)
4746                             ENDIF
4746  2994 5E                    MOV    E,M               ;MOVE THE LEFT
4746  2995 23                    INX    H                 ;LINK INTO THE
4746  2996 56                    MOV    D,M               ;(D,E) REGISTER
4746                             IF     QW              ;IS IT SAVE LINK MODE?
4746  2997 EB                    XCHG                     ;YES - LINK TO (H,L)
4746  2998 22CE42                SHLD   QW              ;LINK TO MEMORY
4746                             ENDIF
4746
4747  299B 7C                    MOV    A,H               ;BUT TEST FOR A
4748  299C B5                    ORA    L                 ;NULL LEFT SUBTREE
4749  299D CAE929                JZ     ENQ5              ;NULL - ADD NEW NODE
4750  29A0 16FF                  MVI    D,0FFH            ;MARK DIRECTION AS LEFT
4751                    ENQ3A:     GTBAL  0,0               ;TEST THE BALANCE FACTOR
4751                               IF     00000H              ;IS NODE ADDRESS IN (H,L)?
4751                               LHLD   00000H              ;NO - NODE POINTER TO (H,L)
4751                               ENDIF
4751  29A2 23                      INX    H                 ;STEP POINTER TO
4751  29A3 23                      INX    H                 ;THE BALANCE BYTE
4751  29A4 23                      INX    H
4751  29A5 23                      INX    H
4751  29A6 7E                      MOV    A,M               ;BALANCE CODE TO A-REG
4751                               IF     00000H              ;TEST IF SAVE NODE
4751                               STA    00000H              ;YES - CODE TO MEMORY
4751                               ENDIF
4751  29A7 B7                      ORA    A                 ;SET CONDITION CODES
4751
4752  29A8 CAC129                  JZ     ENQ3B             ;OF THIS NODE - BALANCED
4753  29AB 7A                      MOV    A,D               ;GET EXIT DIRECTION FROM
4754  29AC 32D442                  STA    QDIR              ;THIS NODE'S PARENT AND SAVE
4755  29AF 2AC842                  LHLD   QP                ;RECORD PARENT OF THE
4756  29B2 22CC42                  SHLD   QT                ;NEW REBALANCE POINT
4757  29B5 2ACE42                  LHLD   QW                ;MARK THE NEXT NODE
4758  29B8 22D042                  SHLD   QS                ;AS REBALANCE POINT
```

```
4759   29BB 22C842            SHLD   QP              ;AND AS CURRENT NODE
4760   29BE C38B29            JMP    ENQ2            ;FIND DIRECTION OF NEXT STEP
4761                  ;
4762   29C1 2ACE42   ENQ3B:   LHLD   QW          ;MARK THIS AS
4763   29C4 22C842            SHLD   QP          ;  THE CURRENT NODE
4764   29C7 C38B29            JMP    ENQ2        ;FIND DIRECTION OF NEXT STEP
4765                  ;
4766                  ENQ4:    GTRLK  QP,QW           ;MOVE TO THE RIGHT
4766                           IF     QP            ;IS NODE ADDRESS IN (H,L)?
4766   29CA 2AC842             LHLD   QP            ;NO - NODE POINTER TO (H,L)
4766                           ENDIF
4766   29CD 23                 INX    H               ;STEP POINTER TO
4766   29CE 23                 INX    H               ;THE RIGHT LINK
4766   29CF 5E                 MOV    E,M             ;MOVE THE RIGHT
4766   29D0 23                 INX    H               ;LINK INTO THE
4766   29D1 56                 MOV    D,M             ;(D,E) REGISTER
4766                           IF     QW             ;IS IT SAVE LINK MODE?
4766   29D2 EB                 XCHG                   ;YES - LINK TO (H,L)
4766   29D3 22CE42             SHLD   QW             ;LINK TO MEMORY
4766                           ENDIF
4766
4767   29D6 7C                 MOV    A,H             ;BUT TEST FOR A
4768   29D7 B5                 ORA    L               ;NULL RIGHT SUBTREE
4769   29D8 1600               MVI    D,0             ;BUT MARK A STEP RIGHT
4770   29DA C2A229             JNZ    ENQ3A           ;CONTINUE IF NOT NULL SUBTREE
4771   29DD D1                 POP    D               ;GET NEW NODE ADDRESS
4772                           PTRLK  QP,0            ;ATTACH AS RIGHT SUBTREE
4772   0001           MCTEM    SET    1               ;ASSUME ADDRESS IN (H,L)
4772                           IF     QP            ;TEST ASSUMPTION
4772   0000           MCTEM    SET    0               ;FALSE ASSUMPTION
4772                           ENDIF
4772                           IF     00000H           ;IS NEW LINK IN (D,E)?
4772                           IF     MCTEM           ;NO - BUT IS POINTER IN (H,L)?
4772                           XCHG                   ;YES - SAVE NODE POINTER
4772                           ENDIF
4772                           LHLD   00000H           ;NEW LINK TO (H,L)
4772                           XCHG                   ;NEW LINK TO (D,E)
4772                           ENDIF
4772                           IF     QP            ;IS NODE ADDRESS IN (H,L)?
4772   29DE 2AC842             LHLD   QP            ;NO - NODE POINTER TO (H,L)
4772                           ENDIF
4772   29E1 23                 INX    H               ;NOW STEP TO THE
4772   29E2 23                 INX    H               ;RIGHT LINK BYTES
4772   29E3 73                 MOV    M,E             ;MOVE THE NEW
4772   29E4 23                 INX    H               ;RIGHT LINK VALUE
4772   29E5 72                 MOV    M,D             ;INTO THE NODE
4772
4773   29E6 C3F029             JMP    ENQ6            ;IF NEEDED - BALANCE THE TREE
4774                  ;
4775   29E9 D1        ENQ5:    POP    D               ;GET NEW NODE ADDRESS
4776                           PTLLK  QP,0            ;ATTACH AS LEFT SUBTREE
4776   0001           MCTEM    SET    1               ;ASSUME ADDRESS IN (H,L)
4776                           IF     QP            ;TEST ASSUMPTION
4776   0000           MCTEM    SET    0               ;FALSE ASSUMPTION
4776                           ENDIF
4776                           IF     00000H           ;IS NEW LINK IN (D,E)?
4776                           IF     MCTEM           ;NO - BUT IS POINTER IN (H,L)?
4776                           XCHG                   ;YES - SAVE NODE POINTER
4776                           ENDIF
4776                           LHLD   00000H           ;NEW LINK TO (H,L)
4776                           XCHG                   ;NEW LINK TO (D,E)
4776                           ENDIF
4776                           IF     QP            ;IS NODE ADDRESS IN (H,L)?
4776   29EA 2AC842             LHLD   QP            ;NO - NODE POINTER TO (H,L)
4776                           ENDIF
4776   29ED 73                 MOV    M,E             ;MOVE THE NEW
4776   29EE 23                 INX    H               ;LEFT LINK VALUE
4776   29EF 72                 MOV    M,D             ;INTO THE NODE
4776
4777                  ;
4778                  ;BALANCE TREE
4779                  ;
4780   29F0 EB        ENQ6:    XCHG                   ;NOW SAVE THE ADDRESS
4781   29F1 22CE42             SHLD   QW              ;OF THE ADDED NODE
4782   29F4 2AD042             LHLD   QS              ;DETERMINE IF LEFT OR
4783   29F7 CDFD2B             CALL   CMPTM           ;RIGHT SUBTREE TO BE
4784   29FA F5                 PUSH   PSW             ;ADJUSTED - SAVE INDICATOR
4785   29FB 2AD042             LHLD   QS              ;POINT TO REBALANCE POINT
4786   29FE FA092A             JM     EN6A1           ;TEST FOR LEFT SUBTREE
4787                           GTRLK  0,0             ;IT IS RIGHT SUBTREE
4787                           IF     00000H           ;IS NODE ADDRESS IN (H,L)?
4787                           LHLD   00000H           ;NO - NODE POINTER TO (H,L)
4787                           ENDIF
4787   2A01 23                 INX    H               ;STEP POINTER TO
4787   2A02 23                 INX    H               ;THE RIGHT LINK
4787   2A03 5E                 MOV    E,M             ;MOVE THE RIGHT
4787   2A04 23                 INX    H               ;LINK INTO THE
```

```
4787   2A05 56                MOV    D,M              ;(D,E) REGISTER
4787                          IF     00000H             ;IS IT SAVE LINK MODE?
4787                          XCHG                    ;YES - LINK TO (H,L)
4787                          SHLD   00000H             ;LINK TO MEMORY
4787                          ENDIF
4787
4788   2A06 C30C2A            JMP     EN6A2           ;GO TO COMMON PATH
4789                  ;
4790                  EN6A1:  GTLLK  0,0              ;IT IS LEFT SUBTREE
4790                          IF     00000H       ;IS NODE ADDRESS IN (H,L)?
4790                          LHLD   00000H             ;NO-NODE POINTER TO (H,L)
4790                          ENDIF
4790   2A09 5E                MOV    E,M              ;MOVE THE LEFT
4790   2A0A 23                INX    H                ;LINK INTO THE
4790   2A0B 56                MOV    D,M              ;(D,E) REGISTER
4790                          IF     00000H             ;IS IT SAVE LINK MODE?
4790                          XCHG                    ;YES - LINK TO (H,L)
4790                          SHLD   00000H             ;LINK TO MEMORY
4790                          ENDIF
4790
4791   2A0C EB        EN6A2:  XCHG                    ;GET START OF CHAIN
4792   2A0D 22C842            SHLD    QP              ;INITIALIZE THE
4793   2A10 22D242            SHLD    QR              ;PATH POINTERS
4794   2A13 EB        EN6A3:  XCHG                    ;SAVE NEXT NODE ADDRESS
4795   2A14 2ACE42            LHLD    QW              ;BUT TEST IF IT IS
4796   2A17 7D                MOV     A,L             ;THE NEWLY ADDED NODE
4797   2A18 AB                XRA     E
4798   2A19 C2212A            JNZ     EN6A4           ;NO
4799   2A1C 7C                MOV     A,H             ;MAYBE - COMPARE THE
4800   2A1D AA                XRA     D               ;SECOND BYTES
4801   2A1E CA532A            JZ      ENQ7            ;OK - BALANCE FACTORS ADJUSTED
4802   2A21 EB        EN6A4:  XCHG                    ;NEW NODE ADDRESS TO (H,L)
4803   2A22 CDFD2B            CALL    CMPTM           ;COMPARE THE KEYS
4804   2A25 2AC842            LHLD    QP              ;AGAIN POINT TO THE NODE
4805   2A28 FA412A            JM      EN6A5           ;WE MUST CHASE LEFTWARDS
4806   2A2B 3E01              MVI     A,01            ;SET BALANCE FACTOR TO +
4807                          PTBAL   0,0             ;AND ADJUST THIS NODE
4807                          IF     00000H             ;IS FACTOR IN A-REG?
4807                          LDA    00000H             ;NO - BALANCE TO A-REG
4807                          ENDIF
4807                          IF     00000H             ;IS NODE ADDRESS IN (H,L)
4807                          LHLD   00000H             ;NO - NODE POINTER TO (H,L)
4807                          ENDIF
4807   2A2D 23                INX    H                ;STEP POINTER TO
4807   2A2E 23                INX    H                ;THE BALANCE BYTE
4807   2A2F 23                INX    H
4807   2A30 23                INX    H
4807   2A31 77                MOV    M,A              ;BALANCE CODE TO NODE
4807
4808                          GTRLK   QP,QP             ;MOVE TO THE RIGHT
4808                          IF     QP             ;IS NODE ADDRESS IN (H,L)?
4808   2A32 2AC842            LHLD   QP             ;NO - NODE POINTER TO (H,L)
4808                          ENDIF
4808   2A35 23                INX    H                ;STEP POINTER TO
4808   2A36 23                INX    H                ;THE RIGHT LINK
4808   2A37 5E                MOV    E,M              ;MOVE THE RIGHT
4808   2A38 23                INX    H                ;LINK INTO THE
4808   2A39 56                MOV    D,M              ;(D,E) REGISTER
4808                          IF     QP              ;IS IT SAVE LINK MODE?
4808   2A3A EB                XCHG                    ;YES - LINK TO (H,L)
4808   2A3B 22C842            SHLD   QP              ;LINK TO MEMORY
4808                          ENDIF
4808
4809   2A3E C3132A            JMP     EN6A3           ;CONTINUE
4810
4811
4812                  ;
4813                  EN6A5:  PTBAL   0,0             ;SET NEW BALANCE AS (-)
4813                          IF     00000H             ;IS FACTOR IN A-REG?
4813                          LDA    00000H             ;NO - BALANCE TO A-REG
4813                          ENDIF
4813                          IF     00000H             ;IS NODE ADDRESS IN (H,L)
4813                          LHLD   00000H             ;NO - NODE POINTER TO (H,L)
4813                          ENDIF
4813   2A41 23                INX    H                ;STEP POINTER TO
4813   2A42 23                INX    H                ;THE BALANCE BYTE
4813   2A43 23                INX    H
4813   2A44 23                INX    H
4813   2A45 77                MOV    M,A              ;BALANCE CODE TO NODE
4813
4814                          GTLLK   QP,QP             ;MOVE TO THE LEFT
4814                          IF     QP       ;IS NODE ADDRESS IN (H,L)?
4814   2A46 2AC842            LHLD   QP             ;NO-NODE POINTER TO (H,L)
4814                          ENDIF
4814   2A49 5E                MOV    E,M              ;MOVE THE LEFT
4814   2A4A 23                INX    H                ;LINK INTO THE
4814   2A4B 56                MOV    D,M              ;(D,E) REGISTER
```

```
4814                            IF    QP              ;IS IT SAVE LINK MODE?
4814  2A4C EB                   XCHG                    ;YES - LINK TO (H,L)
4814  2A4D 22C842               SHLD  QP              ;LINK TO MEMORY
4814                            ENDIF
4814
4815  2A50 C3132A               JMP    EN6A3            ;CONTINUE
4816                ;
4817  2A53 F1       EN07:       POP    PSW              ;RECOVER REBALANCE DIRECTION
4818  2A54 FA072B               JM     EN7AM            ;WE ARE ADJUSTING A LEFT SUBTREE
4819                            GTBAL  QS.0             ;WAS REBALANCE POINT BALANCED?
4819                            IF    QS              ;IS NODE ADDRESS IN (H,L)?
4819  2A57 2AD042               LHLD  QS              ;NO - NODE POINTER TO (H,L)
4819                            ENDIF
4819  2A5A 23                   INX   H                 ;STEP POINTER TO
4819  2A5B 23                   INX   H                 ;THE BALANCE BYTE
4819  2A5C 23                   INX   H
4819  2A5D 23                   INX   H
4819  2A5E 7E                   MOV   A,M               ;BALANCE CODE TO A-REG
4819                            IF    00000H              ;TEST IF SAVE NODE
4819                            STA   00000H              ;YES - CODE TO MEMORY
4819                            ENDIF
4819  2A5F B7                   ORA   A                 ;SET CONDITION CODES
4819
4820  2A60 C2672A               JNZ    EN7P1            ;NO
4821  2A63 3E01                 MVI    A,01             ;YES - SET BALANCE FACTOR TO (+)
4822  2A65 77                   MOV    M,A              ;BECAUSE THE TREE IS HIGHER
4823  2A66 C9                   RET                     ;WE ARE DONE
4824                ;
4825  2A67 F26D2A   EN7P1:      JP     EN7P2            ;SUBTREE WAS HEAVY TO RIGHT
4826  2A6A AF       EN7X1:      XRA    A                ;IT WAS HEAVY TO THE LEFT
4827  2A6B 77                   MOV    M,A              ;NOW IT IS BALANCED
4828  2A6C C9                   RET                     ;WE ARE DONE
4829                ;
4830                EN7P2:      GTBAL  QR.0             ;TREE TOO FAR OUT OF BALANCE
4830                            IF    QR              ;IS NODE ADDRESS IN (H,L)?
4830  2A6D 2AD242               LHLD  QR              ;NO - NODE POINTER TO (H,L)
4830                            ENDIF
4830  2A70 23                   INX   H                 ;STEP POINTER TO
4830  2A71 23                   INX   H                 ;THE BALANCE BYTE
4830  2A72 23                   INX   H
4830  2A73 23                   INX   H
4830  2A74 7E                   MOV   A,M               ;BALANCE CODE TO A-REG
4830                            IF    00000H              ;TEST IF SAVE NODE
4830                            STA   00000H              ;YES - CODE TO MEMORY
4830                            ENDIF
4830  2A75 B7                   ORA   A                 ;SET CONDITION CODES
4830
4831  2A76 FA972A               JM     EN7P9            ;DETERMINE ADJUSTMENT NEEDS
4832  2A79 2AD242               LHLD   QR               ;SAVE ADDRESS OF ROOT
4833  2A7C 22C842               SHLD   QP               ;OF THE RIGHT SUBTREE
4834                            GTLLK  0,0              ;ATTACH ITS LEFT SUBTREE AS
4834                            IF    00000H          ;IS NODE ADDRESS IN (H,L)?
4834                            LHLD  00000H              ;NO-NODE POINTER TO (H,L)
4834                            ENDIF
4834  2A7F 5E                   MOV   E,M               ;MOVE THE LEFT
4834  2A80 23                   INX   H                 ;LINK INTO THE
4834  2A81 56                   MOV   D,M               ;(D,E) REGISTER
4834                            IF    00000H              ;IS IT SAVE LINK MODE?
4834                            XCHG                    ;YES - LINK TO (H,L)
4834                            SHLD  00000H              ;LINK TO MEMORY
4834                            ENDIF
4834
4835                            PTRLK  QS.0             ;THE RIGHT SUBTREE OF REBALANCE POINT
4835  0001          MCTEM       SET   1                 ;ASSUME ADDRESS IN (H,L)
4835                            IF    QS              ;TEST ASSUMPTION
4835  0000          MCTEM       SET   0                 ;FALSE ASSUMPTION
4835                            ENDIF
4835                            IF    00000H              ;IS NEW LINK IN (D,E)?
4835                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4835                            XCHG                    ;YES - SAVE NODE POINTER
4835                            ENDIF
4835                            LHLD  00000H              ;NEW LINK TO (H,L)
4835                            XCHG                    ;NEW LINK TO (D,E)
4835                            ENDIF
4835                            IF    QS              ;IS NODE ADDRESS IN (H,L)?
4835  2A82 2AD042               LHLD  QS              ;NO - NODE POINTER TO (H,L)
4835                            ENDIF
4835  2A85 23                   INX   H                 ;NOW STEP TO THE
4835  2A86 23                   INX   H                 ;RIGHT LINK BYTES
4835  2A87 73                   MOV   M,E               ;MOVE THE NEW
4835  2A88 23                   INX   H                 ;RIGHT LINK VALUE
4835  2A89 72                   MOV   M,D               ;INTO THE NODE
4835
4836                            PTLLK  QR.QS            ;BALANCE POINT BECOMES LEFT SUBTREE
4836  0001          MCTEM       SET   1                 ;ASSUME ADDRESS IN (H,L)
4836                            IF    QR              ;TEST ASSUMPTION
4836  0000          MCTEM       SET   0                 ;FALSE ASSUMPTION
```

```
4836                            ENDIF
4836                            IF    QS              ;IS NEW LINK IN (D,E)?
4836                            IF    MCTEM              ;NO - BUT IS POINTER IN (H,L)?
4836                            XCHG                     ;YES - SAVE NODE POINTER
4836                            ENDIF
4836  2A8A 2AD042               LHLD  QS              ;NEW LINK TO (H,L)
4836  2A8D EB                   XCHG                     ;NEW LINK TO (D,E)
4836                            ENDIF
4836                            IF    QR              ;IS NODE ADDRESS IN (H,L)?
4836  2A8E 2AD242               LHLD  QR              ;NO - NODE POINTER TO (H,L)
4836                            ENDIF
4836  2A91 73                   MOV   M,E                ;MOVE THE NEW
4836  2A92 23                   INX   H                  ;LEFT LINK VALUE
4836  2A93 72                   MOV   M,D                ;INTO THE NODE
4836
4837  2A94 C3422B               JMP    EN7T5             ;RESET BOTH BALANCE FACTORS
4838
4839
4840                    ;
4841                    EN7P9:  GTLLK  QR,QP              ;MOVE RIGHT THEN LEFT FROM BALANCE POINT
4841                            IF    QR      ;IS NODE ADDRESS IN (H,L)?
4841  2A97 2AD242               LHLD  QR              ;NO-NODE POINTER TO (H,L)
4841                            ENDIF
4841  2A9A 5E                   MOV   E,M                ;MOVE THE LEFT
4841  2A9B 23                   INX   H                  ;LINK INTO THE
4841  2A9C 56                   MOV   D,M                ;(D,E) REGISTER
4841                            IF    QP               ;IS IT SAVE LINK MODE?
4841  2A9D EB                   XCHG                     ;YES - LINK TO (H,L)
4841  2A9E 22C842               SHLD  QP              ;LINK TO MEMORY
4841                            ENDIF
4841
4842                            GTRLK  0,0               ;THIS CODE EFFECTS A CLOCKWISE ROTATION
4842                            IF    00000H              ;IS NODE ADDRESS IN (H,L)?
4842                            LHLD  00000H              ;NO - NODE POINTER TO (H,L)
4842                            ENDIF
4842  2AA1 23                   INX   H                  ;STEP POINTER TO
4842  2AA2 23                   INX   H                  ;THE RIGHT LINK
4842  2AA3 5E                   MOV   E,M                ;MOVE THE RIGHT
4842  2AA4 23                   INX   H                  ;LINK INTO THE .
4842  2AA5 56                   MOV   D,M                ;(D,E) REGISTER
4842                            IF    00000H                ;IS IT SAVE LINK MODE?
4842                            XCHG                     ;YES - LINK TO (H,L)
4842                            SHLD  00000H                ;LINK TO MEMORY
4842                            ENDIF
4842
4843                            PTLLK  QR,0              ;OF THE RIGHT SUBTREE FOLLOWED BY
4843  0001              MCTEM   SET   1                  ;ASSUME ADDRESS IN (H,L)
4843                            IF    QR              ;TEST ASSUMPTION
4843  0000              MCTEM   SET   0                  ;FALSE ASSUMPTION
4843                            ENDIF
4843                            IF    00000H              ;IS NEW LINK IN (D,E)?
4843                            IF    MCTEM              ;NO - BUT IS POINTER IN (H,L)?
4843                            XCHG                     ;YES - SAVE NODE POINTER
4843                            ENDIF
4843                            LHLD  00000H              ;NEW LINK TO (H,L)
4843                            XCHG                     ;NEW LINK TO (D,E)
4843                            ENDIF
4843                            IF    QR              ;IS NODE ADDRESS IN (H,L)?
4843  2AA6 2AD242               LHLD  QR              ;NO - NODE POINTER TO (H,L)
4843                            ENDIF
4843  2AA9 73                   MOV   M,E                ;MOVE THE NEW
4843  2AAA 23                   INX   H                  ;LEFT LINK VALUE
4843  2AAB 72                   MOV   M,D                ;INTO THE NODE
4843
4844                            PTRLK  QP,QR             ;A COUNTER CLOCKWISE ROTATION OF
4844  0001              MCTEM   SET   1                  ;ASSUME ADDRESS IN (H,L)
4844                            IF    QP              ;TEST ASSUMPTION
4844  0000              MCTEM   SET   0                  ;FALSE ASSUMPTION
4844                            ENDIF
4844                            IF    QR              ;IS NEW LINK IN (D,E)?
4844                            IF    MCTEM              ;NO - BUT IS POINTER IN (H,L)?
4844                            XCHG                     ;YES - SAVE NODE POINTER
4844                            ENDIF
4844  2AAC 2AD242               LHLD  QR              ;NEW LINK TO (H,L)
4844  2AAF EB                   XCHG                     ;NEW LINK TO (D,E)
4844                            ENDIF
4844                            IF    QP              ;IS NODE ADDRESS IN (H,L)?
4844  2AB0 2AC842               LHLD  QP              ;NO - NODE POINTER TO (H,L)
4844                            ENDIF
4844  2AB3 23                   INX   H                  ;NOW STEP TO THE .
4844  2AB4 23                   INX   H                  ;RIGHT LINK BYTES
4844  2AB5 73                   MOV   M,E                ;MOVE THE NEW
4844  2AB6 23                   INX   H                  ;RIGHT LINK VALUE
4844  2AB7 72                   MOV   M,D                ;INTO THE NODE
4844
4845                            GTLLK  QP,0              ;THE TREE. IN THIS CONTEXT THE
4845                            IF    QP      ;IS NODE ADDRESS IN (H,L)?
```

```
4845   2AB8 2AC842            LHLD  QP            ;NO-NODE POINTER TO (H,L)
4845                          ENDIF
4845   2ABB 5E                MOV   E,M              ;MOVE THE LEFT
4845   2ABC 23                INX   H                ;LINK INTO THE
4845   2ABD 56                MOV   D,M              ;(D,E) REGISTER
4845                          IF    00000H             ;IS IT SAVE LINK MODE?
4845                          XCHG                   ;YES - LINK TO (H,L)
4845                          SHLD  00000H               ;LINK TO MEMORY
4845                          ENDIF
4845
4846                          PTRLK  QS,0            ;REBALANCE POINT IS
4846   0001           MCTEM   SET   1                ;ASSUME ADDRESS IN (H,L)
4846                          IF    QS             ;TEST ASSUMPTION
4846   0000           MCTEM   SET   0                ;FALSE ASSUMPTION
4846                          ENDIF
4846                          IF    00000H              ;IS NEW LINK IN (D,E)?
4846                          IF    MCTEM            ;NO - BUT IS POINTER IN (H,L)?
4846                          XCHG                   ;YES - SAVE NODE POINTER
4846                          ENDIF
4846                          LHLD  00000H              ;NEW LINK TO (H,L)
4846                          XCHG                   ;NEW LINK TO (D,E)
4846                          ENDIF
4846                          IF    QS             ;IS NODE ADDRESS IN (H,L)?
4846   2ABE 2AD042            LHLD  QS             ;NO - NODE POINTER TO (H,L)
4846                          ENDIF
4846   2AC1 23                INX   H                ;NOW STEP TO THE
4846   2AC2 23                INX   H                ;RIGHT LINK BYTES
4846   2AC3 73                MOV   M,E              ;MOVE THE NEW
4846   2AC4 23                INX   H                ;RIGHT LINK VALUE
4846   2AC5 72                MOV   M,D              ;INTO THE NODE
4846
4847                          PTLLK  QP,QS           ;CONSIDERED TO BE THE ROOT
4847   0001           MCTEM   SET   1                ;ASSUME ADDRESS IN (H,L)
4847                          IF    QP             ;TEST ASSUMPTION
4847   0000           MCTEM   SET   0                ;FALSE ASSUMPTION
4847                          ENDIF
4847                          IF    QS             ;IS NEW LINK IN (D,E)?
4847                          IF    MCTEM            ;NO - BUT IS POINTER IN (H,L)?
4847                          XCHG                   ;YES - SAVE NODE POINTER
4847                          ENDIF
4847   2AC6 2AD042            LHLD  QS             ;NEW LINK TO (H,L)
4847   2AC9 EB                XCHG                   ;NEW LINK TO (D,E)
4847                          ENDIF
4847                          IF    QP             ;IS NODE ADDRESS IN (H,L)?
4847   2ACA 2AC842            LHLD  QP             ;NO - NODE POINTER TO (H,L)
4847                          ENDIF
4847   2ACD 73                MOV   M,E              ;MOVE THE NEW
4847   2ACE 23                INX   H                ;LEFT LINK VALUE
4847   2ACF 72                MOV   M,D              ;INTO THE NODE
4847
4848   2AD0 23                INX    H               ;STEP (H,L) TO
4849   2AD1 23                INX    H               ;POINT AT THE BALANCE
4850   2AD2 23                INX    H               ;FACTOR OF THE NEW ROOT
4851   2AD3 56                MOV    D,M             ;GET THE ORIGINAL
4852   2AD4 AF                XRA    A               ;BALANCE FACTOR AND
4853   2AD5 77                MOV    M,A             ;MARK THE ROOT BALANCED
4854   2AD6 B2                ORA    D               ;TEST ORIGINAL BALANCE FACTOR
4855   2AD7 CA422B            JZ     EN7T5           ;BALANCED - SO SUBTREES ARE BALANCED
4856   2ADA FAF22A            JM     EN7P8           ;IT WAS HEAVY TO THE LEFT
4857   2ADD AF                XRA    A               ;IT WAS HEAVY TO THE RIGHT
4858                          PTBAL  QR,0            ;SET BALANCE FACTORS
4858                          IF    00000H             ;IS FACTOR IN A-REG?
4858                          LDA   00000H              ;NO - BALANCE TO A-REG
4858                          ENDIF
4858                          IF    QR             ;IS NODE ADDRESS IN (H,L)
4858   2ADE 2AD242            LHLD  QR             ;NO - NODE POINTER TO (H,L)
4858                          ENDIF
4858   2AE1 23                INX   H                ;STEP POINTER TO
4858   2AE2 23                INX   H                ;THE BALANCE BYTE
4858   2AE3 23                INX   H
4858   2AE4 23                INX   H
4858   2AE5 77                MOV   M,A              ;BALANCE CODE TO NODE
4858
4859   2AE6 2F                CMA                    ;FOR THE SUBTREES THAT
4860                          PTBAL  QS,0            ;WERE ROTATED DURING ADJUSTMENT
4860                          IF    00000H             ;IS FACTOR IN A-REG?
4860                          LDA   00000H              ;NO - BALANCE TO A-REG
4860                          ENDIF
4860                          IF    QS             ;IS NODE ADDRESS IN (H,L)
4860   2AE7 2AD042            LHLD  QS             ;NO - NODE POINTER TO (H,L)
4860                          ENDIF
4860   2AEA 23                INX   H                ;STEP POINTER TO
4860   2AEB 23                INX   H                ;THE BALANCE BYTE
4860   2AEC 23                INX   H
4860   2AED 23                INX   H
4860   2AEE 77                MOV   M,A              ;BALANCE CODE TO NODE
4860
```

```
4861  2AEF C3C32B           JMP    NQA10          ;ADJUST LINKAGE TO TOTAL TREE
4862                ;
4863  2AF2 AF       EN7P8:  XRA    A              ;SET THE BALANCE FACTORS
4864                        PTBAL  QS,0           ;FOR THE SUBTREES THAT WERE
4864                        IF     00000H          ;IS FACTOR IN A-REG?
4864                        LDA    00000H          ;NO - BALANCE TO A-REG
4864                        ENDIF
4864                        IF     QS           ;IS NODE ADDRESS IN (H,L)
4864  2AF3 2AD042           LHLD   QS           ;NO - NODE POINTER TO (H,L)
4864                        ENDIF
4864  2AF6 23               INX    H              ;STEP POINTER TO
4864  2AF7 23               INX    H              ;THE BALANCE BYTE
4864  2AF8 23               INX    H
4864  2AF9 23               INX    H
4864  2AFA 77               MOV    M,A            ;BALANCE CODE TO NODE
4864
4865  2AFB 3C               INR    A              ;ROTATED DURING ADJUSTMENT
4866                        PTBAL  QR,0
4866                        IF     00000H          ;IS FACTOR IN A-REG?
4866                        LDA    00000H          ;NO - BALANCE TO A-REG
4866                        ENDIF
4866                        IF     QR           ;IS NODE ADDRESS IN (H,L)
4866  2AFC 2AD242           LHLD   QR           ;NO - NODE POINTER TO (H,L)
4866                        ENDIF
4866  2AFF 23               INX    H              ;STEP POINTER TO
4866  2B00 23               INX    H              ;THE BALANCE BYTE
4866  2B01 23               INX    H
4866  2B02 23               INX    H
4866  2B03 77               MOV    M,A            ;BALANCE CODE TO NODE
4866
4867  2B04 C3C32B           JMP    NQA10          ;ADJUST LINKAGE TO TOTAL TREE
4868                ;
4869                EN7AM:  GTBAL  QS,0           ;WAS REBALANCE POINT BALANCED?
4869                        IF     QS           ;IS NODE ADDRESS IN (H,L)?
4869  2B07 2AD042           LHLD   QS           ;NO - NODE POINTER TO (H,L)
4869                        ENDIF
4869  2B0A 23               INX    H              ;STEP POINTER TO
4869  2B0B 23               INX    H              ;THE BALANCE BYTE
4869  2B0C 23               INX    H
4869  2B0D 23               INX    H
4869  2B0E 7E               MOV    A,M            ;BALANCE CODE TO A-REG
4869                        IF     00000H          ;TEST IF SAVE NODE
4869                        STA    00000H          ;YES - CODE TO MEMORY
4869                        ENDIF
4869  2B0F B7               ORA    A              ;SET CONDITION CODES
4869
4870  2B10 C2162B           JNZ    EN7M1          ;NO
4871  2B13 2F               CMA                   ;YES - SET BALANCE FACTOR TO (-)
4872  2B14 77               MOV    M,A            ;BECAUSE THE TREE IS HIGHER.
4873  2B15 C9               RET                   ;WE ARE DONE
4874                ;
4875  2B16 F26A2A   EN7M1:  JP     EN7X1          ;SUBTREE WAS HEAVY TO RIGHT
4876                        GTBAL  QR,0           ;TREE TOO FAR OUT OF BALANCE
4876                        IF     QR           ;IS NODE ADDRESS IN (H,L)?
4876  2B19 2AD242           LHLD   QR           ;NO - NODE POINTER TO (H,L)
4876                        ENDIF
4876  2B1C 23               INX    H              ;STEP POINTER TO
4876  2B1D 23               INX    H              ;THE BALANCE BYTE
4876  2B1E 23               INX    H
4876  2B1F 23               INX    H
4876  2B20 7E               MOV    A,M            ;BALANCE CODE TO A-REG
4876                        IF     00000H          ;TEST IF SAVE NODE
4876                        STA    00000H          ;YES - CODE TO MEMORY
4876                        ENDIF
4876  2B21 B7               ORA    A              ;SET CONDITION CODES
4876
4877  2B22 F2562B           JP     EN7M9          ;DETERMINE ADJUSTMENT NEEDS
4878  2B25 2AD242           LHLD   QR             ;SAVE ADDRESS OF ROOT
4879  2B28 22C842           SHLD   QP             ;OF THE LEFT SUBTREE
4880                        GTRLK  0,0            ;ATTACH ITS RIGHT SUBTREE AS
4880                        IF     00000H          ;IS NODE ADDRESS IN (H,L)?
4880                        LHLD   00000H          ;NO - NODE POINTER TO (H,L)
4880                        ENDIF
4880  2B2B 23               INX    H              ;STEP POINTER TO
4880  2B2C 23               INX    H              ;THE RIGHT LINK
4880  2B2D 5E               MOV    E,M            ;MOVE THE RIGHT
4880  2B2E 23               INX    H              ;LINK INTO THE
4880  2B2F 56               MOV    D,M            ;(D,E) REGISTER
4880                        IF     00000H          ;IS IT SAVE LINK MODE?
4880                        XCHG                  ;YES - LINK TO (H,L)
4880                        SHLD   00000H          ;LINK TO MEMORY
4880                        ENDIF
4880
4881                        PTLLK  QS,0           ;THE LEFT SUBTREE OF REBALANCE POINT
4881  0001          MCTEM   SET    1              ;ASSUME ADDRESS IN (H,L)
4881                        IF     QS           ;TEST ASSUMPTION
4881  0000          MCTEM   SET    0              ;FALSE ASSUMPTION
```

```
4881                        ENDIF
4881                        IF    00000H            ;IS NEW LINK IN (D,E)?
4881                        IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4881                        XCHG                    ;YES - SAVE NODE POINTER
4881                        ENDIF
4881                        LHLD  00000H            ;NEW LINK TO (H,L)
4881                        XCHG                    ;NEW LINK TO (D,E)
4881                        ENDIF
4881                        IF    QS              ;IS NODE ADDRESS IN (H,L)?
4881  2B30 2AD042           LHLD  QS              ;NO - NODE POINTER TO (H,L)
4881                        ENDIF
4881  2B33 73               MOV   M,E               ;MOVE THE NEW
4881  2B34 23               INX   H                 ;LEFT LINK VALUE
4881  2B35 72               MOV   M,D               ;INTO THE NODE
4881
4882                        PTRLK  QR,QS            ;BALANCE POINT BECOMES RIGHT SUBTREE
4882  0001          MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
4882                        IF    QR              ;TEST ASSUMPTION
4882  0000          MCTEM   SET   0                 ;FALSE ASSUMPTION
4882                        ENDIF
4882                        IF    QS              ;IS NEW LINK IN (D,E)?
4882                        IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4882                        XCHG                    ;YES - SAVE NODE POINTER
4882                        ENDIF
4882  2B36 2AD042           LHLD  QS              ;NEW LINK TO (H,L)
4882  2B39 EB               XCHG                    ;NEW LINK TO (D,E)
4882                        ENDIF
4882                        IF    QR              ;IS NODE ADDRESS IN (H,L)?
4882  2B3A 2AD242           LHLD  QR              ;NO - NODE POINTER TO (H,L)
4882                        ENDIF
4882  2B3D 23               INX   H                 ;NOW STEP TO THE
4882  2B3E 23               INX   H                 ;RIGHT LINK BYTES
4882  2B3F 73               MOV   M,E               ;MOVE THE NEW
4882  2B40 23               INX   H                 ;RIGHT LINK VALUE
4882  2B41 72               MOV   M,D               ;INTO THE NODE
4882
4883                ;
4884  2B42 AF       EN7T5:  XRA   A                 ;TOTAL TREE IS BALANCED - SO
4885                        PTBAL  QS,0             ;RESET THE BALANCE FACTORS
4885                        IF    00000H            ;IS FACTOR IN A-REG?
4885                        LDA   00000H            ;NO - BALANCE TO A-REG
4885                        ENDIF
4885                        IF    QS              ;IS NODE ADDRESS IN (H,L)
4885  2B43 2AD042           LHLD  QS              ;NO - NODE POINTER TO (H,L)
4885                        ENDIF
4885  2B46 23               INX   H                 ;STEP POINTER TO
4885  2B47 23               INX   H                 ;THE BALANCE BYTE
4885  2B48 23               INX   H
4885  2B49 23               INX   H
4885  2B4A 77               MOV   M,A               ;BALANCE CODE TO NODE
4885
4886                        PTBAL  QR,0             ;IN THE AFFECTED NODES
4886                        IF    00000H            ;IS FACTOR IN A-REG?
4886                        LDA   00000H            ;NO - BALANCE TO A-REG
4886                        ENDIF
4886                        IF    QR              ;IS NODE ADDRESS IN (H,L)
4886  2B4B 2AD242           LHLD  QR              ;NO - NODE POINTER TO (H,L)
4886                        ENDIF
4886  2B4E 23               INX   H                 ;STEP POINTER TO
4886  2B4F 23               INX   H                 ;THE BALANCE BYTE
4886  2B50 23               INX   H
4886  2B51 23               INX   H
4886  2B52 77               MOV   M,A               ;BALANCE CODE TO NODE
4886
4887  2B53 C3C32B           JMP    NQA10            ;ADJUST LINKAGE TO TOTAL TREE
4888                ;
4889                EN7M9:  GTRLK  QR,QP            ;MOVE LEFT THEN RIGHT FROM BALANCE POINT
4889                        IF    QR              ;IS NODE ADDRESS IN (H,L)?
4889  2B56 2AD242           LHLD  QR              ;NO - NODE POINTER TO (H,L)
4889                        ENDIF
4889  2B59 23               INX   H                 ;STEP POINTER TO
4889  2B5A 23               INX   H                 ;THE RIGHT LINK
4889  2B5B 5E               MOV   E,M               ;MOVE THE RIGHT
4889  2B5C 23               INX   H                 ;LINK INTO THE
4889  2B5D 56               MOV   D,M               ;(D,E) REGISTER
4889                        IF    QP              ;IS IT SAVE LINK MODE?
4889  2B5E EB               XCHG                    ;YES - LINK TO (H,L)
4889  2B5F 22C842           SHLD  QP              ;LINK TO MEMORY
4889                        ENDIF
4889
4890                        GTLLK  0,0              ;THIS CODE EFFECTS A COUNTER CLOCKWISE
4890                        IF    00000H          ;IS NODE ADDRESS IN (H,L)?
4890                        LHLD  00000H            ;NO-NODE POINTER TO (H,L)
4890                        ENDIF
4890  2B62 5E               MOV   E,M               ;MOVE THE LEFT
4890  2B63 23               INX   H                 ;LINK INTO THE
4890  2B64 56               MOV   D,M               ;(D,E) REGISTER
```

```
4890                        IF    00000H            ;IS IT SAVE LINK MODE?
4890                        XCHG                  ;YES - LINK TO (H,L)
4890                        SHLD  00000H            ;LINK TO MEMORY
4890                        ENDIF
4890
4891                        PTRLK  QR,0             ;ROTATION OF THE LEFT SUBTREE FOLLOWED
4891  0001          MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
4891                        IF    QR              ;TEST ASSUMPTION
4891  0000          MCTEM   SET   0                 ;FALSE ASSUMPTION
4891                        ENDIF
4891                        IF    00000H            ;IS NEW LINK IN (D,E)?
4891                        IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4891                        XCHG                  ;YES - SAVE NODE POINTER
4891                        ENDIF
4891                        LHLD  00000H            ;NEW LINK TO (H,L)
4891                        XCHG                  ;NEW LINK TO (D,E)
4891                        ENDIF
4891                        IF    QR              ;IS NODE ADDRESS IN (H,L)?
4891  2B65 2AD242           LHLD  QR              ;NO - NODE POINTER TO (H,L)
4891                        ENDIF
4891  2B68 23               INX   H                 ;NOW STEP TO THE
4891  2B69 23               INX   H                 ;RIGHT LINK BYTES
4891  2B6A 73               MOV   M,E               ;MOVE THE NEW
4891  2B6B 23               INX   H                 ;RIGHT LINK VALUE
4891  2B6C 72               MOV   M,D               ;INTO THE NODE
4891
4892                        PTLLK  QP,QR            ;BY A CLOCKWISE ROTATION OF THE TREE.
4892  0001          MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
4892                        IF    QP              ;TEST ASSUMPTION
4892  0000          MCTEM   SET   0                 ;FALSE ASSUMPTION
4892                        ENDIF
4892                        IF    QR              ;IS NEW LINK IN (D,E)?
4892                        IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4892                        XCHG                  ;YES - SAVE NODE POINTER
4892                        ENDIF
4892  2B6D 2AD242           LHLD  QR              ;NEW LINK TO (H,L)
4892  2B70 EB               XCHG                  ;NEW LINK TO (D,E)
4892                        ENDIF
4892                        IF    QP              ;IS NODE ADDRESS IN (H,L)?
4892  2B71 2AC842           LHLD  QP              ;NO - NODE POINTER TO (H,L)
4892                        ENDIF
4892  2B74 73               MOV   M,E               ;MOVE THE NEW
4892  2B75 23               INX   H                 ;LEFT LINK VALUE
4892  2B76 72               MOV   M,D               ;INTO THE NODE
4892
4893                        GTRLK  QP,0             ;IN THIS CONTEXT THE REBALANCE
4893                        IF    QP              ;IS NODE ADDRESS IN (H,L)?
4893  2B77 2AC842           LHLD  QP              ;NO - NODE POINTER TO (H,L)
4893                        ENDIF
4893  2B7A 23               INX   H                 ;STEP POINTER TO
4893  2B7B 23               INX   H                 ;THE RIGHT LINK
4893  2B7C 5E               MOV   E,M               ;MOVE THE RIGHT
4893  2B7D 23               INX   H                 ;LINK INTO THE
4893  2B7E 56               MOV   D,M               ;(D,E) REGISTER
4893                        IF    00000H            ;IS IT SAVE LINK MODE?
4893                        XCHG                  ;YES - LINK TO (H,L)
4893                        SHLD  00000H            ;LINK TO MEMORY
4893                        ENDIF
4893
4894                        PTLLK  QS,0             ;POINT IS CONSIDERED TO BE
4894  0001          MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
4894                        IF    QS              ;TEST ASSUMPTION
4894  0000          MCTEM   SET   0                 ;FALSE ASSUMPTION
4894                        ENDIF
4894                        IF    00000H            ;IS NEW LINK IN (D,E)?
4894                        IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4894                        XCHG                  ;YES - SAVE NODE POINTER
4894                        ENDIF
4894                        LHLD  00000H            ;NEW LINK TO (H,L)
4894                        XCHG                  ;NEW LINK TO (D,E)
4894                        ENDIF
4894                        IF    QS              ;IS NODE ADDRESS IN (H,L)?
4894  2B7F 2AD042           LHLD  QS              ;NO - NODE POINTER TO (H,L)
4894                        ENDIF
4894  2B82 73               MOV   M,E               ;MOVE THE NEW
4894  2B83 23               INX   H                 ;LEFT LINK VALUE
4894  2B84 72               MOV   M,D               ;INTO THE NODE
4894
4895                        PTRLK  QP,QS            ;THE ROOT NODE OF THE TREE
4895  0001          MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
4895                        IF    QP              ;TEST ASSUMPTION
4895  0000          MCTEM   SET   0                 ;FALSE ASSUMPTION
4895                        ENDIF
4895                        IF    QS              ;IS NEW LINK IN (D,E)?
4895                        IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4895                        XCHG                  ;YES - SAVE NODE POINTER
4895                        ENDIF
```

```
4895   2B85 2AD042            LHLD  QS            ;NEW LINK TO (H,L)
4895   2B88 EB                XCHG                   ;NEW LINK TO (D,E)
4895                          ENDIF
4895                          IF    QP              ;IS NODE ADDRESS IN (H,L)?
4895   2B89 2AC842            LHLD  QP              ;NO - NODE POINTER TO (H,L)
4895                          ENDIF
4895   2B8C 23                INX   H                  ;NOW STEP TO THE
4895   2B8D 23                INX   H                  ;RIGHT LINK BYTES
4895   2B8E 73                MOV   M,E                ;MOVE THE NEW
4895   2B8F 23                INX   H                  ;RIGHT LINK VALUE
4895   2B90 72                MOV   M,D                ;INTO THE NODE
4895
4896   2B91 23                INX    H                 ;STEP TO THE BALANCE FACTOR
4897   2B92 56                MOV    D,M               ;GET ORINAL BALANCE FACTOR
4898   2B93 AF                XRA    A                 ;OF NEW SUBTREE ROOT NODE
4899   2B94 77                MOV    M,A               ;AND MARK IT AS BALANCED
4900   2B95 B2                ORA    D                 ;TEST ORIGINAL BALANCE FACTOR
4901   2B96 CA422B            JZ     EN7T5             ;OK - ALL SUBTREES BALANCED
4902   2B99 F2B12B            JP     EN7M8             ;IT WAS HEAVY TO THE RIGHT
4903   2B9C AF                XRA    A                 ;IT WAS HEAVY TO THE LEFT
4904                          PTBAL  QR,0              ;SET BALANCE FACTORS FOR
4904                          IF     00000H             ;IS FACTOR IN A-REG?
4904                          LDA    00000H             ;NO - BALANCE TO A-REG
4904                          ENDIF
4904                          IF     QR              ;IS NODE ADDRESS IN (H,L)
4904   2B9D 2AD242            LHLD   QR              ;NO - NODE POINTER TO (H,L)
4904                          ENDIF
4904   2BA0 23                INX    H                 ;STEP POINTER TO
4904   2BA1 23                INX    H                 ;THE BALANCE BYTE
4904   2BA2 23                INX    H
4904   2BA3 23                INX    H
4904   2BA4 77                MOV    M,A               ;BALANCE CODE TO NODE
4904
4905   2BA5 3C                INR     A                ;THE SUBTREES THAT WERE
4906                          PTBAL  QS,0              ;ROTATED DURING ADJUSTMENT
4906                          IF     00000H             ;IS FACTOR IN A-REG?
4906                          LDA    00000H             ;NO - BALANCE TO A-REG
4906                          ENDIF
4906                          IF     QS              ;IS NODE ADDRESS IN (H,L)
4906   2BA6 2AD042            LHLD   QS              ;NO - NODE POINTER TO (H,L)
4906                          ENDIF
4906   2BA9 23                INX    H                 ;STEP POINTER TO
4906   2BAA 23                INX    H                 ;THE BALANCE BYTE
4906   2BAB 23                INX    H
4906   2BAC 23                INX    H
4906   2BAD 77                MOV    M,A               ;BALANCE CODE TO NODE
4906
4907   2BAE C3C32B            JMP     NQA10            ;ADJUST LINKAGE TO TOTAL TREE
4908                  ;
4909   2BB1 AF        EN7M8:  XRA     A                ;SET BALANCE FACTORS
4910                          PTBAL  QS,0              ;FOR THE SUBTREES THAT WERE
4910                          IF     00000H             ;IS FACTOR IN A-REG?
4910                          LDA    00000H             ;NO - BALANCE TO A-REG
4910                          ENDIF
4910                          IF     QS              ;IS NODE ADDRESS IN (H,L)
4910   2BB2 2AD042            LHLD   QS              ;NO - NODE POINTER TO (H,L)
4910                          ENDIF
4910   2BB5 23                INX    H                 ;STEP POINTER TO
4910   2BB6 23                INX    H                 ;THE BALANCE BYTE
4910   2BB7 23                INX    H
4910   2BB8 23                INX    H
4910   2BB9 77                MOV    M,A               ;BALANCE CODE TO NODE
4910
4911   2BBA 2F                CMA                      ;ROTATED DURING ADJUSTMENT
4912                          PTBAL  QR,0
4912                          IF     00000H             ;IS FACTOR IN A-REG?
4912                          LDA    00000H             ;NO - BALANCE TO A-REG
4912                          ENDIF
4912                          IF     QR              ;IS NODE ADDRESS IN (H,L)
4912   2BBB 2AD242            LHLD   QR              ;NO - NODE POINTER TO (H,L)
4912                          ENDIF
4912   2BBE 23                INX    H                 ;STEP POINTER TO
4912   2BBF 23                INX    H                 ;THE BALANCE BYTE
4912   2BC0 23                INX    H
4912   2BC1 23                INX    H
4912   2BC2 77                MOV    M,A               ;BALANCE CODE TO NODE
4912
4913                  ;
4914   2BC3 3AD442    NQA10:  LDA     QDIR             ;GET THE ENTRY DIRECTION
4915   2BC6 B7                ORA     A                ;INTO THE BALANCE POINT
4916   2BC7 FAD72B            JM      ENQ11            ;AND TEST IT
4917                          PTRLK  QT,QP             ;LINK ADJUSTED SUBTREE
4917   0001           MCTEM   SET    1                 ;ASSUME ADDRESS IN (H,L)
4917                          IF     QT              ;TEST ASSUMPTION
4917   0000           MCTEM   SET    0                 ;FALSE ASSUMPTION
4917                          ENDIF
4917                          IF     QP              ;IS NEW LINK IN (D,E)?
```

```
4917                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4917                            XCHG                    ;YES - SAVE NODE POINTER
4917                            ENDIF
4917   2BCA 2AC842              LHLD  QP              ;NEW LINK TO (H,L)
4917   2BCD EB                  XCHG                    ;NEW LINK TO (D,E)
4917                            ENDIF
4917                            IF    QT              ;IS NODE ADDRESS IN (H,L)?
4917   2BCE 2ACC42              LHLD  QT              ;NO - NODE POINTER TO (H,L)
4917                            ENDIF
4917   2BD1 23                  INX   H                 ;NOW STEP TO THE
4917   2BD2 23                  INX   H                 ;RIGHT LINK BYTES
4917   2BD3 73                  MOV   M,E               ;MOVE THE NEW
4917   2BD4 23                  INX   H                 ;RIGHT LINK VALUE
4917   2BD5 72                  MOV   M,D               ;INTO THE NODE
4917
4918   2BD6 C9                  RET                     ;AS A RIGHT SUBTREE
4919                    ;
4920                    ENQ11:  PTLLK  QT,QP            ;LINK ADJUSTED SUBTREE
4920   0001             MCTEM   SET   1                 ;ASSUME ADDRESS IN (H,L)
4920                            IF    QT              ;TEST ASSUMPTION
4920   0000             MCTEM   SET   0                 ;FALSE ASSUMPTION
4920                            ENDIF
4920                            IF    QP              ;IS NEW LINK IN (D,E)?
4920                            IF    MCTEM             ;NO - BUT IS POINTER IN (H,L)?
4920                            XCHG                    ;YES - SAVE NODE POINTER
4920                            ENDIF
4920   2BD7 2AC842              LHLD  QP              ;NEW LINK TO (H,L)
4920   2BDA EB                  XCHG                    ;NEW LINK TO (D,E)
4920                            ENDIF
4920                            IF    QT              ;IS NODE ADDRESS IN (H,L)?
4920   2BDB 2ACC42              LHLD  QT              ;NO - NODE POINTER TO (H,L)
4920                            ENDIF
4920   2BDE 73                  MOV   M,E               ;MOVE THE NEW
4920   2BDF 23                  INX   H                 ;LEFT LINK VALUE
4920   2BE0 72                  MOV   M,D               ;INTO THE NODE
4920
4921   2BE1 C9                  RET                     ;AS A LEFT SUBTREE
4922                    ;
4923   2BE2 EB          TREMT:  XCHG                    ;RECOVER THE NODE ADDRESS
4924   2BE3 22BA42              SHLD  RLINK             ;MARK TREE NONEMPTY
4925   2BE6 CDEE2B              CALL  NDCLR             ;RESET LINK AND BALANCE FACTOR
4926   2BE9 EB                  XCHG                    ;GET EVENT TIME FROM NODE
4927   2BEA 22BD42              SHLD  NXTIM             ;SET AS NEXT EVENT TIME
4928   2BED C9                  RET                     ;ALL DONE
4929                    ;
4930   2BEE AF          NDCLR:  XRA   A                 ;RESET THE LINKS AND
4931   2BEF 77                  MOV   M,A               ;THE BALANCE FACTOR OF
4932   2BF0 23                  INX   H                 ;THE SELECTED NODE
4933   2BF1 77                  MOV   M,A
4934   2BF2 23                  INX   H
4935   2BF3 77                  MOV   M,A
4936   2BF4 23                  INX   H
4937   2BF5 77                  MOV   M,A
4938   2BF6 23                  INX   H
4939   2BF7 77                  MOV   M,A
4940   2BF8 23                  INX   H                 ;STEP TO THE EVENT TIME
4941   2BF9 5E                  MOV   E,M               ;RETURN WITH THE
4942   2BFA 23                  INX   H                 ;EVENT TIME IN (D,E)
4943   2BFB 56                  MOV   D,M
4944   2BFC C9                  RET
4945                    ;
4946   2BFD 110500      CMPTM:  LXI   D,05              ;POINT TO THE TIME WORD
4947   2C00 19                  DAD   D                 ;OF SELECTED NODE
4948   2C01 5E                  MOV   E,M               ;GET LOW ORDER BITS
4949   2C02 23                  INX   H                 ;POINT AT HIGH BITS
4950   2C03 56                  MOV   D,M               ;AND GET THEM
4951   2C04 2AD642              LHLD  NEWTM             ;GET TIME OF NEW NODE
4952   2C07 7D                  MOV   A,L               ;FORM
4953   2C08 93                  SUB   E                 ;(NEW TIME)-(NODE TIME)
4954   2C09 7C                  MOV   A,H
4955   2C0A 9A                  SBB   D
4956                    ;        ADD    A                ;DECISION BIT TO SIGN
4957   2C0B C9                  RET                     ;EXIT
4958                    ;;
4959                    ;FATAL ERROR HANDLER.  WAIT FOR "CLEAR" FROM MASTER KEYBOARD
4960                    ;  AND THEN COLD START THE COMPUTER.
4961                    ;  IF "SEND" IS HIT INSTEAD, WILL RETURN TO CALLER.
4962                    ;
4963                    ;INPUT: A= ERROR CODE.
4964                    ;ALL REGISTERS RESTORED.
4965                    ;
4966   2C0C F5          FATAL:  PUSH  PSW             ;SAVE PSW FOR DEBUGGER
4967   2C0D 3A6842              LDA   SL700           ;IF MASTER KEYBOARD
4968   2C10 B7                  ORA   A               ;  IS T.I.
4969   2C11 CA162C              JZ    FATL            ;  SILENT 700
4970   2C14 F1                  POP   PSW             ;RESTORE PSW
4971   2C15 D7                  RST   2               ;ENTER DEBUGGER
```

```
4972                    ;
4973                    ;NORMAL FATAL ERROR HANDLING
4974                    ;
4975   2C16 F1          FATL:   POP   PSW       ;A = ERROR CODE
4976   2C17 E5                  PUSH  H         ;SAVE
4977   2C18 D5                  PUSH  D         ;  ALL
4978   2C19 C5                  PUSH  B         ;  REGISTERS
4979   2C1A F5                  PUSH  PSW       ;
4980   2C1B F6F0                ORI   0F0H      ;FATAL ERROR CODES ARE 0F0H TO 0FFH
4981   2C1D CD080F              CALL  ERPC      ;DISPLAY MESSAGE
4982   2C20 CD0911              CALL  OFCVY     ;TURN OFF CONVEYOR
4983   2C23 3E01                MVI   A,1       ;LOCK OUT
4984   2C25 32AC42              STA   FSQ0      ;  ALL OTHER KB PROCESSING
4985                   ;
4986                   ;  ;BEEP AND WAIT FOR COMMAND
4987                   ;
4988   2C28 063B        FALRM:  MVI   B,';'     ;B = "BEEP"
4989   2C2A 3A093D              LDA   MAKB      ;C
4990   2C2D 4F                  MOV   C,A       ;  = MASTER KB #
4991   2C2E CD6C11              CALL  PUTKB     ;ALARM OPERATOR
4992   2C31 110500              LXI   D,5       ;IDLE ABOUT
4993   2C34 CD5A2D              CALL  RDELA     ;  500 MS.
4994   2C37 C5                  PUSH  B         ;SAVE MASTER KB #
4995   2C38 CD0E10              CALL  GETKB     ;C = KB #, A = CHAR
4996   2C3B 47                  MOV   B,A       ;SAVE CHARACTER IN B
4997   2C3C D1                  POP   D         ;E = MASTER KB #
4998   2C3D 79                  MOV   A,C       ;WAIT UNTIL
4999   2C3E BB                  CMP   E         ;  THERE IS INPUT
5000   2C3F C2282C              JNZ   FALRM     ;  FROM MASTER KB.
5001   2C42 78                  MOV   A,B       ;A = CHAR
5002   2C43 FE3B                CPI   ';'       ;IF ';' THEN
5003   2C45 CA4B2C              JZ    FATLX     ;  RETURN
5004   2C48 C3DD03              JMP   EXEC      ;ELSE COLD START
5005                   ;
5006   2C4B CD190C      FATLX:  CALL  DEVII     ;RE-INITIALIZE DEVICES
5007   2C4E F1                  POP   PSW       ;RESTORE
5008   2C4F C1                  POP   B         ;  ALL
5009   2C50 D1                  POP   D         ;  REGISTERS
5010   2C51 E1                  POP   H         ;
5011   2C52 C9                  RET
5012                   ;;
5013                   ;GET DCDB QUEUE POINTER IN D,E.
5014                   ;
5015                   ;INPUT: C=JDV.
5016                   ;OUTPUT: B,C=JDV, H,L POINTS TO DCDB(JDV), SO DOES DCDBP.
5017                   ;
5018   2C53 2A5742      GDCQ:   LHLD  DCDB      ;POINT H,L TO DCDB TABLE
5019   2C56 0600                MVI   B,0       ;B,C=JDV
5020   2C58 09                  DAD   B         ;POINT H,L TO
5021   2C59 09                  DAD   B         ;  DCDB(JDV)
5022   2C5A 2B                  DCX   H         ;  MS BYTE
5023   2C5B 56                  MOV   D,M       ;D,E
5024   2C5C 2B                  DCX   H         ;  =
5025   2C5D 5E                  MOV   E,M       ;  DCDB(JDV)
5026   2C5E 220143              SHLD  DCDBP     ;SET DCDBP
5027   2C61 C9                  RET
5028                   ;
5029                   ;
5030                   ;GET JDV, IDENT FROM LAQ IN PPDB.
5031                   ;
5032                   ;INPUT: H,L POINTS TO BYTE 2 IN PPDB.
5033                   ;OUTPUT: B=JDV, D,E=IDENT, H,L POINTS TO IDENT MS BYTE IN PPDB,
5034                   ;  FLAG Z SET & H,L NOT EFFECTED ON NEGATIVE RETURN.
5035                   ;ALL REGISTERS EXCEPT C EFFECTED.
5036                   ;
5037   2C62 110400      GLAQ:   LXI   D,4       ;POINT  TO BYTE 6
5038   2C65 19                  DAD   D         ;  IN PPDB
5039   2C66 7E                  MOV   A,M       ;IF
5040   2C67 B7                  ORA   A         ;  LAQ
5041   2C68 C8                  RZ              ;  NOT EMPTY
5042                   ;
5043                   ;REMOVE JDV,IDENT FROM LAQ IN PPDB
5044                   ;
5045   2C69 3A0040              LDA   FLAGS     ;IF AUTO-INDUCT
5046   2C6C B7                  ORA   A         ;  IS
5047   2C6D F2752C              JP    GDLAQ     ;  DESIRED
5048   2C70 23                  INX   H         ;PICK UP LAQ
5049   2C71 5E                  MOV   E,M       ;  ITEM WITHOUT
5050   2C72 C37D2C              JMP   GLAQ1     ;  DELETING
5051                   ;
5052   2C75 35          GDLAQ:  DCR   M         ;DEC LAQSZ IN PPDB
5053   2C76 23                  INX   H         ;H,L POINTS TO LAQ1
5054   2C77 7E                  MOV   A,M       ;PICK UP LAQ 1ST ITEM # LAQ1
5055   2C78 5F                  MOV   E,A       ;SAVE LAQ1 IN E
5056   2C79 3C                  INR   A         ;NEXT ITEM IN LAQ BECOMES 1ST ITEM
5057   2C7A E607                ANI   07H       ;WRAP AROUND IF NECESSARY
5058   2C7C 77                  MOV   M,A       ;UPDATE LAQ1 IN PPDB
```

```
5059   2C7D 2C       GLAQ1:   INR   L          ;H,L POINTS TO LAQ (ALSO RESET FLAG Z)
5060   2C7E 1600               MVI   D,0        ;D,E=LAQ1
5061   2C80 19                 DAD   D          ;H,L POINTS TO
5062   2C81 19                 DAD   D          ;  ITEM # LAQ1
5063   2C82 19                 DAD   D          ;  IN LAQ
5064   2C83 46                 MOV   B,M        ;B=JDV
5065   2C84 23                 INX   H          ;H,L POINTS TO IDENT
5066   2C85 5E                 MOV   E,M        ;D,E
5067   2C86 23                 INX   H          ;  = PACKAGE
5068   2C87 56                 MOV   D,M        ;  IDENTIFICATION
5069   2C88 C9                 RET              ;POSITIVE RETURN
5070                  ;;
5071                  ;CREATE  UDB1  FOR NEXT UPDATE PHOTO.  COPY IDENT, JDV,
5072                  ;  JUPDA, JINDU FROM OLD TCB TO NEW TCB AND INCREMENT JUPDA.
5073                  ;  CALLED BY IDBPC OR UDBPC.
5074                  ;
5075                  ;INPUT: B,C POINTS TO BYTE 13 IN OLD TCB.
5076                  ;OUTPUT: NEW TCB, H,L POINTS TO BYTE 7 IN NEW TCB,
5077                  ;  A=JINDU, D,E= OLD JUPDA.
5078                  ;ALL REGISTERS EFFECTED.
5079                  ;
5080   2C89 CDCB0F   GUDB1:   CALL  GETFS      ;GET FREE BLOCK FOR UDB1
5081   2C8C 110D00             LXI   D,13       ;POINT H,L TO BYTE 13
5082   2C8F 19                 DAD   D          ;  IN TCB
5083   2C90 0A                 LDAX  B          ;COPY
5084   2C91 77                 MOV   M,A        ;  IDENT
5085   2C92 0B                 DCX   B          ;  FROM
5086   2C93 2B                 DCX   H          ;  OLD TCB
5087   2C94 0A                 LDAX  B          ;  TO
5088   2C95 77                 MOV   M,A        ;  NEW TCB
5089   2C96 0B                 DCX   B          ;MOVE POINTERS TO
5090   2C97 2B                 DCX   H          ;  BYTES 11
5091   2C98 0A                 LDAX  B          ;COPY
5092   2C99 77                 MOV   M,A        ;  JDV
5093   2C9A 0B                 DCX   B          ;MOVE POINTERS TO
5094   2C9B 2B                 DCX   H          ;  BYTES 10
5095   2C9C 0A                 LDAX  B          ;A= OLD JUPDA
5096   2C9D 5F                 MOV   E,A        ;E= OLD JUPDA FOR UDBPC
5097   2C9E 3C                 INR   A          ;A= NEW JUPDA
5098   2C9F 77                 MOV   M,A        ;SET JUPDA IN NEW TCB
5099   2CA0 0B                 DCX   B          ;MOVE POINTERS TO
5100   2CA1 2B                 DCX   H          ;  BYTES 9
5101   2CA2 0A                 LDAX  B          ;COPY
5102   2CA3 77                 MOV   M,A        ;  JINDU
5103   2CA4 2B                 DCX   H          ;POINT H,L TO BYTE 8 IN NEW TCB
5104   2CA5 019424             LXI   B,UDB1     ;EVENT
5105   2CA8 70                 MOV   M,B        ;  IN TCB
5106   2CA9 2B                 DCX   H          ;  =
5107   2CAA 71                 MOV   M,C        ;  UDB1
5108   2CAB C9                 RET
5109                  ;;
5110                  ;SET UP COUNT IN PPDB FOR UNBLOCKED PP # C+1 OR A PENDING
5111                  ;  UNCODED PACKAGE AT PP.  FOR STRAIGHT MERGE ONLY.
5112                  ;  CALLED BY MCLPC OR TECMG.
5113                  ;
5114                  ;INPUT: C= INDUCT #, PWTNG & COUNT IN PPDB.
5115                  ;OUTPUT: COUNT(JINDU) IN PPDB, POINTED TO BY H,L.
5116                  ;ALL REGISTERS EXCEPT C EFFECTED.
5117                  ;
5118   2CAC 2A4342   PEND:    LHLD  PPDB       ;POINT H,L TO BYTE 1
5119   2CAF 23                 INX   H          ;  IN PPDB
5120   2CB0 3A033D             LDA   NINDU      ;IF IT IS
5121   2CB3 3D                 DCR   A          ;  A SINGLE INDUCT
5122   2CB4 C2BE2C             JNZ   PENDS      ;  SYSTEM
5123                  ;
5124                  ;SPECIAL HANDLING FOR SINGLE INDUCT SYSTEMS
5125                  ;
5126   2CB7 7E                 MOV   A,M        ;PICK UP PWTNG FLAG IN PPDB
5127   2CB8 23                 INX   H          ;POINT TO COUNT IN PPDB
5128   2CB9 FE02               CPI   2          ;IF PWTNG FLAG = 2 THEN
5129   2CBB C0                 RNZ              ;  INCREMENT
5130   2CBC 34                 INR   M          ;  COUNT
5131   2CBD C9                 RET
5132                  ;
5133                  ;ADJUST COUNT FOR MULTIPLE STRAIGHT INDUCT SYSTEMS
5134                  ;
5135   2CBE 110100   PENDS:   LXI   D,0001H    ;INITIALIZE D = MAX COUNT
5136                                           ;  & E = COUNT INCREMENT
```

```
5137   2CC1 43              MOV   B,E       ;B= INDUCT #, = 1 INITIALLY
5138                        ;
5139   2CC2 78       PNWCN: MOV   A,B       ;IF INDUCT
5140   2CC3 B9              CMP   C         ;  NOT #
5141   2CC4 CAD72C          JZ    PNXID     ;  JINDU
5142   2CC7 7E              MOV   A,M       ;A= PWTNG IN PPDB
5143   2CC8 FE02            CPI   2         ;IF NOT=
5144   2CCA CACF2C          JZ    PMXCN     ;  2
5145   2CCD 1E00            MVI   E,0       ;INC COUNT ONLY IF EVERY PWTNG = 2
5146   2CCF 7A       PMXCN: MOV   A,D       ;A= MAX COUNT
5147   2CD0 23              INX   H         ;POINT H,L TO BYTE 2 IN PPDB
5148   2CD1 BE              CMP   M         ;IF COUNT
5149   2CD2 D2D62C          JNC   PNXI      ;  > MAX COUNT THEN
5150   2CD5 56              MOV   D,M       ;MAX COUNT = COUNT
5151   2CD6 2B       PNXI:  DCX   H         ;POINT H,L TO BYTE 1 IN PPDB
5152   2CD7 D5       PNXID: PUSH  D         ;MOVE H,L
5153   2CD8 112000          LXI   D,32      ;  TO
5154   2CDB 19              DAD   D         ;  NEXT
5155   2CDC D1              POP   D         ;  INDUCT
5156   2CDD 04              INR   B         ;NEXT INDUCT #
5157   2CDE 3A033D          LDA   NINDU     ;DO FOR
5158   2CE1 B8              CMP   B         ;  ALL
5159   2CE2 D2C22C          JNC   PNWCN     ;  INDUCTS
5160                        ;
5161   2CE5 7A              MOV   A,D       ;B
5162   2CE6 83              ADD   E         ;  = MAX COUNT + COUNT INC
5163   2CE7 47              MOV   B,A       ;  = NEW COUNT
5164   2CE8 79              MOV   A,C       ;A= INDUCT #
5165   2CE9 CD9927          CALL  CPPPP     ;POINT H,L TO
5166   2CEC 23              INX   H         ;  BYTE 2
5167   2CED 23              INX   H         ;  IN PPDB
5168   2CEE 70              MOV   M,B       ;UPDATE COUNT(JINDU) IN PPDB
5169   2CEF C9              RET
5170                        ;;
5171                        ;PUT JDV & IDENT INTO LAQ IN PPDB FOR INDUCT # C+1.
5172                        ;  CALLED BY SNPC.
5173                        ;
5174                        ;INPUT: C= INDUCT #, JDV,IDENT.
5175                        ;OUTPUT: PPDBP POINTS TO BYTE 1 IN PPDB, FLAG Z SET AND
5176                        ;  A=0 ON LAQ FULL.
5177                        ;A
5178   2CF0 79       PLAQ:  MOV   A,C       ;A= INDUCT #
5179   2CF1 CD9927          CALL  CPPPP     ;POINT H,L TO BYTE 1
5180   2CF4 23              INX   H         ;  IN PPDB(JINDU)
5181   2CF5 22EB42          SHLD  PPDBP     ;OUTPUT POINTER
5182   2CF8 110500          LXI   D,5       ;POINT H,L TO BYTE 6
5183   2CFB 19              DAD   D         ;  IN PPDB
5184   2CFC 3A373D          LDA   QLIMI     ;A= LAQ SIZE LIMIT DESIRED
5185   2CFF 96              SUB   M         ;IF LAQ NOT
5186   2D00 C8              RZ              ;  FULL
5187                        ;
5188                        ;LAQ NOT YET FULL; INSERT JDV & IDENT
5189                        ;
5190   2D01 7E              MOV   A,M       ;PICK UP CURRENT LAQ SIZE
5191   2D02 34              INR   M         ;INC LAQSZ IN PPDB
5192   2D03 23              INX   H         ;POINT H,L TO BYTE 7 IN PPDB
5193   2D04 86              ADD   M         ;COMPUTE LAQ NEW ITEM #
5194   2D05 E607            ANI   07H       ;MASK TO 3 BITS
5195   2D07 5F              MOV   E,A       ;D,E
5196   2D08 1600            MVI   D,0       ;  = LAQ NEW ITEM #
5197   2D0A 23              INX   H         ;POINT H,L TO LAQ ITEM 0
5198   2D0B 19              DAD   D         ;POINT H,L TO
5199   2D0C 19              DAD   D         ;  NEW ITEM
5200   2D0D 19              DAD   D         ;  LOCATION
5201   2D0E EB              XCHG            ;POINT D,E TO NEW LOCATION
5202   2D0F 219642          LXI   H,JDV     ;POINT H,L TO JDV,IDENT
5203   2D12 0603            MVI   B,3       ;B= LOOP COUNT
5204   2D14 CDCF10          CALL  MOVEB     ;COPY JDV,IDENT INTO LAQ
5205   2D17 04              INR   B         ;POSITIVE
5206   2D18 C9              RET             ;  RETURN
5207                        ;
5208   2D19 D5       RSCHD: PUSH  D         ;SAVE DELAY COUNT
5209   2D1A E5              PUSH  H         ;SAVE POINTER
5210   2D1B 5E              MOV   E,M       ;PICK UP
5211   2D1C 23              INX   H         ;  TCB
5212   2D1D 56              MOV   D,M       ;  POINTER
5213   2D1E 7B              MOV   A,E       ;IF NONZERO
5214   2D1F B2              ORA   D         ;  THEN EVENT
5215   2D20 CA2D2D          JZ    RSSEE     ;  PENDING
5216                        ;
5217                        ;JUNK ANY EVENT ALREADY PENDING
5218                        ;
5219   2D23 210700          LXI   H,7       ;POINT TO BYTE 7
5220   2D26 19              DAD   D         ;  IN TCB
5221   2D27 118025          LXI   D,JUNK    ;JUNK
5222   2D2A 73              MOV   M,E       ;  THE
5223   2D2B 23              INX   H         ;  PENDING
5224   2D2C 72              MOV   M,D       ;  EVENT
5225                        ;
```

```
5226                    ;SCHEDULE NEW EVENT
5227                    ;
5228  2D2D CDCB0F       RSSEE:  CALL  GETFS     ;GET NEW BLOCK FOR TCB
5229  2D30 EB                   XCHG            ;POINT DE TO TCB
5230  2D31 E1                   POP   H         ;POINT TO TCB
5231  2D32 E5                   PUSH  H         ;  POINTER
5232  2D33 73                   MOV   M,E       ;SET UP
5233  2D34 23                   INX   H         ;  TCB
5234  2D35 72                   MOV   M,D       ;  POINTER
5235  2D36 210B00               LXI   H,11      ;POINT TO BYTE 11
5236  2D39 19                   DAD   D         ;  IN TCB
5237  2D3A D1                   POP   D         ;POIN DE TO TCB POINTER
5238  2D3B 72                   MOV   M,D       ;SET BACKWARD
5239  2D3C 2B                   DCX   H         ;  POINTER
5240  2D3D 73                   MOV   M,E       ;  IN TCB
5241  2D3E 2B                   DCX   H         ;POINT TO BYTE 9 IN TCB
5242  2D3F 3A9442               LDA   JINDU     ;SAVE INDUCT #
5243  2D42 77                   MOV   M,A       ;  IN TCB
5244  2D43 2B                   DCX   H         ;POINT TO BYTE 8 IN TCB
5245  2D44 70                   MOV   M,B       ;SET EVENT
5246  2D45 2B                   DCX   H         ;  ENTRY IN
5247  2D46 71                   MOV   M,C       ;  TCB
5248  2D47 EB                   XCHG            ;COMPUTE
5249  2D48 C1                   POP   B         ;  DE
5250  2D49 2AB642               LHLD  CTIME     ;  = EVENT
5251  2D4C 09                   DAD   B         ;  DUE
5252  2D4D EB                   XCHG            ;  TIME
5253  2D4E 2B                   DCX   H         ;POINT TO BYTE 6 IN TCB
5254  2D4F 72                   MOV   M,D       ;SET EVENT
5255  2D50 2B                   DCX   H         ;  DUE TIME
5256  2D51 73                   MOV   M,E       ;  IN TCB
5257  2D52 11FBFF               LXI   D,0FFFBH  ;POINT TO
5258  2D55 19                   DAD   D         ;  TCB
5259  2D56 CD4429               CALL  ENQU2     ;SCHEDULE EVENT
5260  2D59 C9                   RET
5261                    ;
5262  2D5A E5           RDELA:  PUSH  H         ;SAVE CALLER'S H,L, A
5263  2D5B F5                   PUSH  PSW       ;  IN STACK
5264  2D5C C5                   PUSH  B         ;GET TREE
5265  2D5D D5                   PUSH  D         ;  PARAMETERS
5266  2D5E AF                   XRA   A         ;  FOR
5267  2D5F CDBA0F               CALL  GTREP     ;  REAL
5268  2D62 D1                   POP   D         ;  TIME
5269  2D63 C1                   POP   B         ;  CLOCK
5270  2D64 C3692D               JMP   RDLPI
5271                    ;
5272                    ;
5273                    ;DELAY D,E PPI PULSES.
5274                    ;
5275                    ;INPUT: TREE PARAMETERS, D,E= # PULSES TO DELAY.
5276                    ;OUTPUT: TCBP POINTS TO DELAYED EVENT.
5277                    ;REGISTERS A, B, C, H, L RESTORED, STACK EFFECTED AFTER DELAY.
5278                    ;  TCBP, TREE DATA EFFECTED.
5279                    ;
5280  2D67 E5           RDLPP:  PUSH  H         ;SAVE CALLER'S H,L, A
5281  2D68 F5                   PUSH  PSW       ;  IN STACK
5282  2D69 2AB642       RDLPI:  LHLD  CTIME     ;SAVE DELAYED EVENT
5283  2D6C 19                   DAD   D         ;  TIME IN
5284  2D6D E5                   PUSH  H         ;  STACK
5285  2D6E CDCB0F               CALL  GETFS     ;GET FREE BLOCK
5286  2D71 22B442               SHLD  TCBP      ;SAVE POINTER
5287  2D74 110500               LXI   D,5       ;POINT TO BYTE 5
5288  2D77 19                   DAD   D         ;  IN TCB
5289  2D78 D1                   POP   D         ;SET
5290  2D79 73                   MOV   M,E       ;  DELAYED
5291  2D7A 23                   INX   H         ;  EVENT
5292  2D7B 72                   MOV   M,D       ;  TIME
5293  2D7C 23                   INX   H         ;POINT TO BYTE 7 IN TCB
5294  2D7D 119B2D               LXI   D,R2DEL   ;SET
5295  2D80 73                   MOV   M,E       ;  EVENT
5296  2D81 23                   INX   H         ;  TO BE
5297  2D82 72                   MOV   M,D       ;  DELA2
5298  2D83 23                   INX   H         ;POINT TO BYTE 9
5299  2D84 F1                   POP   PSW       ;SAVE CALLER'S A
5300  2D85 77                   MOV   M,A       ;  IN TCB
5301  2D86 23                   INX   H         ;POINT TO BYTE 10
5302  2D87 71                   MOV   M,C       ;SAVE CALLER'S
5303  2D88 23                   INX   H         ;  B,C
5304  2D89 70                   MOV   M,B       ;  IN TCB
5305  2D8A 23                   INX   H         ;POINT TO BYTE 12
5306  2D8B D1                   POP   D         ;SAVE CALLER'S
5307  2D8C 73                   MOV   M,E       ;  H,L
5308  2D8D 23                   INX   H         ;  IN
5309  2D8E 72                   MOV   M,D       ;  TCB
5310  2D8F 23                   INX   H         ;POINT TO BYTE 14
```

```
5311   2D90 D1              POP   D         ;SAVE
5312   2D91 73              MOV   M,E       ;  RETURN
5313   2D92 23              INX   H         ;  ADDRESS
5314   2D93 72              MOV   M,D       ;  IN TCB
5315   2D94 2AB442          LHLD  TCBP      ;SCHEDULE
5316   2D97 CD4429          CALL  ENQU2     ;  EVENT DELA2
5317   2D9A C9              RET             ;RETURN TO CALLER OF CALLER
5318                ;
5319                ;RESUMPTION OF DELAYED EVENT.
5320                ;  PART OF MODULE DELAY.
5321                ;
5322                ;INPUT: TCBP, D,E POINTS TO BYTE 8 OF TCB, TREE PARAMETERS.
5323                ;
5324   2D9B 210700  R2DEL:  LXI   H,7       ;POINT TO BYTE 15
5325   2D9E 19              DAD   D         ;  IN TCB
5326   2D9F 56              MOV   D,M       ;PUT CALLER'S
5327   2DA0 2B              DCX   H         ;  RETURN ADDRESS
5328   2DA1 5E              MOV   E,M       ;  IN
5329   2DA2 D5              PUSH  D         ;  STACK
5330   2DA3 2B              DCX   H         ;POINT TO BYTE 13
5331   2DA4 56              MOV   D,M       ;SAVE CALLER'S
5332   2DA5 2B              DCX   H         ;  H,L
5333   2DA6 5E              MOV   E,M       ;  IN
5334   2DA7 D5              PUSH  D         ;  STACK
5335   2DA8 2B              DCX   H         ;POINT TO BYTE 11
5336   2DA9 46              MOV   B,M       ;RESTORE
5337   2DAA 2B              DCX   H         ;  CALLER'S
5338   2DAB 4E              MOV   C,M       ;  B,C
5339   2DAC 2B              DCX   H         ;POINT TO BYTE 10
5340   2DAD 7E              MOV   A,M       ;SAVE CALLER'S A
5341   2DAE F5              PUSH  PSW       ;  IN STACK
5342   2DAF 2AB442          LHLD  TCBP      ;RETURN TCB TO
5343   2DB2 CD4F11          CALL  PUTFS     ;  FREE STORAGE
5344   2DB5 F1              POP   PSW       ;RESTORE CALLER'S A
5345   2DB6 E1              POP   H         ;RESTORE CALLER'S H,L
5346   2DB7 C9              RET             ;RETURN TO CALLER OF DELAY
5347                ;
5348                ;
5349                ;;
5350                ;*************************************************************************
5351                ;        OPTIONAL TRANSLATION AND PROGRAMMING MODULES
5352                ;*************************************************************************
5353                ;
5354   2DB8                 ORG   2F00H     ;TRANSLATION MODULE
5355                ;
5356                ;
5357                ;INITIALIZE TRANSLATION DATABASE AND TABLE.
5358                ;
5359                ;INPUT: INTERRUPTS DISABLED.
5360                ;ALL REGISTERS EFFECTED.
5361                ;
5362   2F00 2A4D42  XLINI:  LHLD  XLDB      ;POINT TO XLDB FOR TABLE # 1
5363   2F03 E5              PUSH  H         ;SAVE POINTER AT TOP OF STACK
5364   2F04 21C03D          LXI   H,XLTBS   ;POINT TO TABLE ADDRESSES
5365   2F07 3A0B3D          LDA   NXLTB     ;SET COUNTER B
5366   2F0A 47              MOV   B,A       ;  = # TRANSLATE TABLES
5367   2F0B 5E      XXLDB:  MOV   E,M       ;PICK UP
5368   2F0C 23              INX   H         ;  TABLE
5369   2F0D 56              MOV   D,M       ;  ADDRESS
5370   2F0E 23              INX   H         ;POINT TO NEXT TABLE AD
5371   2F0F E3              XTHL            ;POINT TO BYTE 1 IN XLDB,
5372   2F10 23              INX   H         ;  LEAVING TABLE-FULL FLAG TBFU = 0
5373   2F11 73              MOV   M,E       ;SET POINTER
5374   2F12 23              INX   H         ;  XLTB
5375   2F13 72              MOV   M,D       ;  IN XLDB
5376   2F14 3A0C3D          LDA   TBSIZ     ;COMPUTE D,E = XLTB2
```

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows.

1. A system for monitoring and controlling the flow of articles in a warehousing system including a sorting conveyer with a plurality of discharge locations in which stored articles are to be batch picked and sorted for shipment comprising:

means for storing data representing each article to be picked, said data including a number assigned to each article which is unique to each article and sort destination information for each article;

indicia mean associated with each article to be picked including said unique number;

scanning means for scanning said indicia means on an article transported within the warehousing system;

computer circuit means coupled to said data storing means and to said scanning means for correlating position information of each article with the article identification number;

display means coupled to said computer circuit means for providing display information to an operator of the status of each article to be picked and sorted;

a conveyer control circuit for selectively controlling the discharge of articles; and

means coupling said conveyer control circuit to said computer circuit means for controlling the diverting of articles in response to the correlation of their unique identification number with associated sort destination information stored in said data storing means.

2. The system as defined in claim 1 wherein data for each batch of articles to be picked is stored on a recording medium which can be transported from a remote central control area, and wherein said system includes data transfer means coupled to said computer circuit means for reading and transferring data from said recording medium to said data storing means.

3. The system as defined in claim 2 wherein said display means includes an operator terminal coupled to said computer circuit means and including a keyboard and CRT display for selectively interrogating the computer circuit means for status information.

4. The system as defined in claim 3 wherein said display means further includes printer means coupled to said computer circuit means for providing a printout of the status of articles to be picked and sorted.

5. The system as defined in claim 4 wherein said display means further includes additional printer means for printing alarm messages relating to predetermined events occuring during the operation of said system.

6. A system for monitoring the status of articles being processed in a warehouse system including article conveying means comprising:

a plurality of label reading means positioned at selected locations within the warehouse system for reading a unique identification number contained on a label applied to each article moving within the system and uniquely identifying each such article;

data storage means including each identification number of articles to be processed and including processing information associated with each such article;

computer circuit means coupled to said label reading means and to said data storage means for monitoring the position of articles travelling within the system and for generating signals representative of the position and therefore the processing status of such articles;

display means coupled to said computer circuit means for displaying the status of articles or groups of articles in the system; and

control means coupled to said computer circuit means for controlling the operation of the warehouse conveying means to control the movement of articles or groups of articles in the system.

7. The system as defined in claim 6 wherein said data storage means includes a data recording medium including each identification number stored thereon together with associated processing information.

8. The system as defined in claim 7 wherein said data storage means further includes data transfer means for reading and writing data to and from said recording medium respectively, said data transfer means coupled to said computer circuit means.

9. The system as defined in claim 8 and further including printer means coupled to said computer circuit means for providing a printout of the status of articles to be processed.

10. The system as defined in claim 9 and further including an operator keyboard and wherein said computer circuit means is programmed to provide reports selectively printed by said printing means when requested by commands from said operator keyboard.

11. A system for monitoring the status of articles being processed comprising:

a plurality of label reading means for reading a unique identification number contained on a label applied to each article moving within the system and uniquely identifying each such article;

data storage means including a data recording medium for storing each identification number together with associated processing information for articles to be processed;

computer circuit means coupled to said label reading means and to said data storage means for monitoring the position of articles travelling within the system and for generating signals representative of the position and therefore the processing status of such articles wherein said data storage means further includes data transfer means for reading and writing data to and from said recording medium respectively, said data transfer means coupled to said computer circuit means;

display means coupled to said computer circuit means for displaying the status of articles or groups of articles in the system said display means including printer means coupled to said computer circuit means for providing a printout of the status of articles to be processed;

an operator keyboard coupled to said computer circuit means and wherein said computer circuit means is programmed to provide reports selectively printed by said printing means when requested by commands from said operator keyboard;

a sorting conveyer with a plurality of discharge chutes;

a conveyer control circuit for selectively controlling the discharge of articles onto said discharge chutes; and

means coupling said conveyer control circuit to said computer circuit means for controlling the diverting of articles in response to the correlation of their unique identification number with associated sort destination information stored in said data storage means.

12. A system for monitoring and controlling the batch order picking and sorting of articles in a ware-

house comprising:

a plurality of label reading means for reading a unique identification number contained on a label applied to each article moving within the system and uniquely identifying each such article;

bulk data storage means including each identification number of articles to be sorted and including sort destination information for each such article;

computer circuit means coupled to said label reading means and to said bulk data storage means for monitoring the physical position of articles travelling within the system and for generating signals representative of the position and therefore the sorting status of such articles;

display means coupled to said computer circuit means for displaying the status of articles or groups of articles in the system;

a conveyer control circuit for selectively controlling the discharge of articles at discharge locations in the warehouse; and

means coupling said conveyer control circuit to said computer circuit means for controlling the diverting of articles in response to the correlation of their unique identification number with associated sort destination information stored in said data storage means.

13. The system as defined in claim **12** wherein said bulk data storage means receives data from a data recording medium including each identification number stored thereon together with associated sorting information.

14. The system as defined in claim **13** wherein said data storage means includes data transfer means for reading and writing data to and from said recording medium respectively, said data transfer means coupled to said computer circuit means.

15. The system as defined in claim **14** wherein said recording medium is a diskette.

16. The system as defined in claim **15** wherein said display means includes printer means coupled to said computer circuit means for providing a printout of the status of articles to be sorted.

17. The system as defined in claim **16** and further including an operator keyboard and wherein said computer circuit means is programmed to provide reports selectively printed by said printing means when requested by commands from said operator keyboard.

* * * * *

United States Patent [19]

Kurland

[11] **Patent Number: 4,547,851**

[45] **Date of Patent: Oct. 15, 1985**

[54] **INTEGRATED INTERACTIVE RESTAURANT COMMUNICATION METHOD FOR FOOD AND ENTERTAINMENT PROCESSING**

[76] Inventor: **Lawrence G. Kurland**, 26 Farmington La., Melville, N.Y. 11747

[21] Appl. No.: **474,983**

[22] Filed: **Mar. 14, 1983**

[51] **Int. Cl.**[4] **G06F 3/04**; G06F 15/24; G06F 15/44

[52] **U.S. Cl.** **364/401**; 364/410; 364/900

[58] **Field of Search** ..... 364/400–401, 364/404–405, 410–412, 200 MS File, 900 MS File; 235/7 R, 383

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,588,838	6/1971	Felcheck	364/900 X
3,668,307	6/1972	Face et al.	178/5.6
3,691,295	9/1972	Fisk	178/58
3,746,780	7/1973	Stetten et al.	178/6.6 A
3,752,908	8/1973	Boenke et al.	178/5.6
3,757,225	9/1973	Ulicki	325/308
3,836,888	9/1974	Boenke et al.	340/172.5
3,903,402	9/1975	Petit et al.	235/151.21
3,968,327	7/1976	Gregg, III	178/6.8
4,001,785	1/1977	Miyazaki et al.	340/172.5
4,008,369	2/1977	Theurer et al.	358/84
4,028,733	6/1977	Ulicki	360/10
4,054,911	10/1977	Fletcher et al.	358/141
4,064,490	12/1977	Nagel	364/200
4,075,686	2/1978	Calle et al.	364/200
4,084,229	4/1978	Taylor et al.	364/200
4,117,605	10/1978	Kurland et al.	35/9 A
4,122,519	10/1978	Bielawski et al.	364/200
4,128,757	12/1978	Garner, Jr.	235/383
4,143,360	3/1979	Bernhart et al.	340/711
4,164,024	8/1979	Gilbert	364/900
4,191,956	3/1980	Groothuis	340/789
4,222,111	9/1980	Sloan et al.	364/900
4,247,106	1/1981	Jeffers et al.	364/410 X
4,251,691	2/1981	Kakihara et al.	179/2 TV
4,264,925	4/1981	Freeman et al.	358/86
4,283,709	8/1981	Lucero et al.	364/410 X
4,296,476	10/1981	Mayer et al.	364/900
4,306,388	12/1981	Yuter	52/6
4,333,152	1/1982	Best	364/410 X
4,388,689	1/1983	Hayman et al.	364/401
4,396,985	8/1983	Ohara	364/405
4,415,065	11/1983	Sandstedt	235/383

OTHER PUBLICATIONS

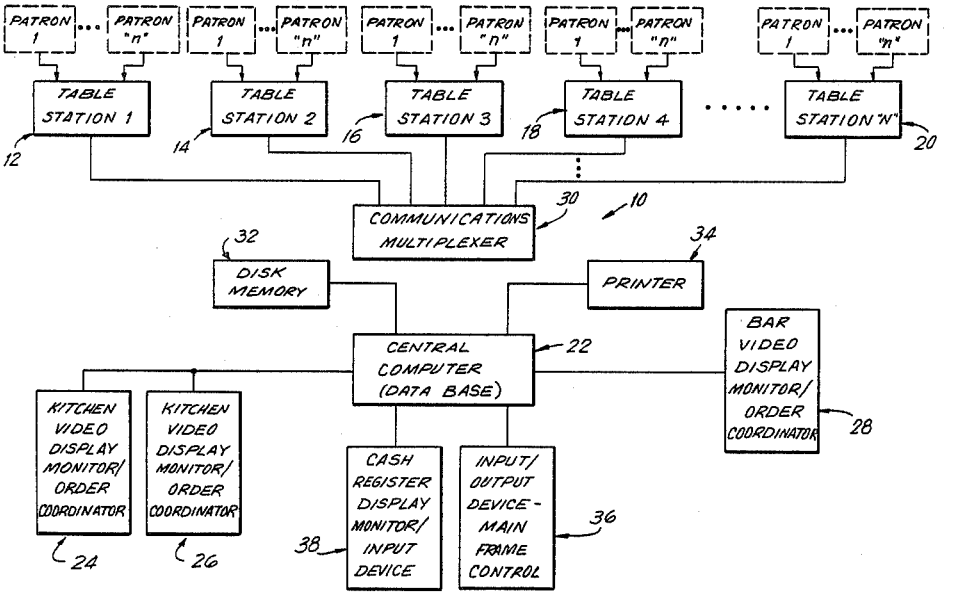
Wilkinson, Jr., "Restaurant System", *IBM Technical Disclosure Bulletin*, vol. 24, No. 9, Feb. 1982, (pp. 4630–4631).

Primary Examiner—Jerry Smith
Assistant Examiner—Gary V. Harkcom
Attorney, Agent, or Firm—Stiefel, Gross, Kurland & Pavane

[57] **ABSTRACT**

An interactive restaurant communication system (**10**) provides integrated food and entertainment processing which enables restaurant patrons to accomplish both food selection and select and receive entertainment on a common video monitor (**56**) at their table. The patrons can obtain menus for individual food selection on the video monitor (**56**) at their tables and individually enter their orders into a table station "intelligent" terminal (**12, 14, 16, 18, 20**) at their tables. In addition they can select from and interactively play a variety of remotely retrievable interactive entertainment activities using the video monitor (**56**) while waiting for the food to arrive, and if desired, where applicable, have the food and entertainment charges automatically added to a composite bill which may be printed at the table station terminal (**12, 14, 16, 18, 20**) or at a remote central location. The food and entertainment functions of the terminal (**12, 14, 16, 18, 20**) are down-line loaded from a central data base (**22, 32**) in response to terminal requests therefor.

11 Claims, 7 Drawing Figures



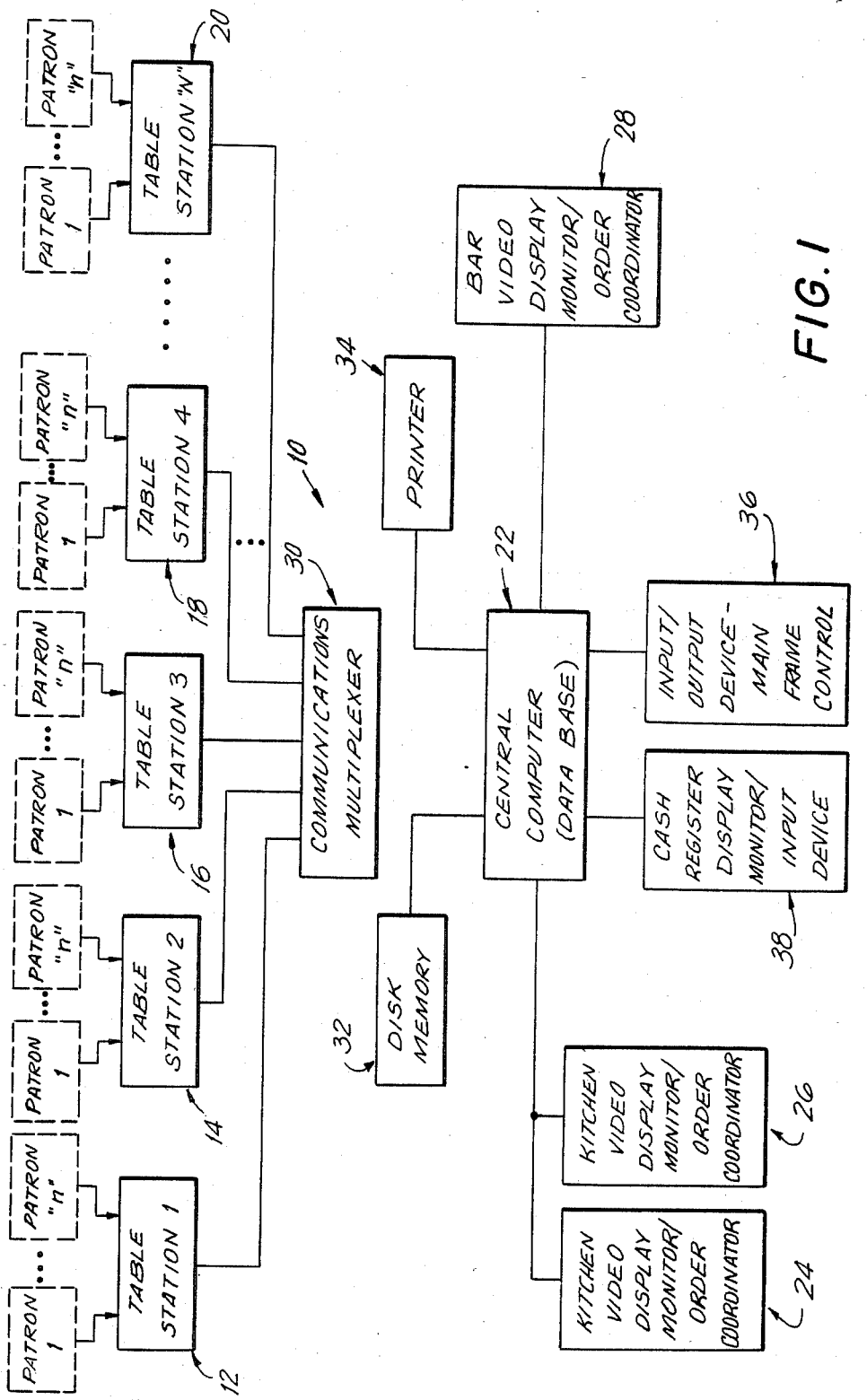

PATRON 1
PATRON "n"
TABLE STATION 1
12
TABLE STATION 2
14
TABLE STATION 3
16
TABLE STATION 4
18
TABLE STATION "N"
20
10
COMMUNICATIONS MULTIPLEXER
30
PRINTER
34
CENTRAL COMPUTER (DATA BASE)
22
DISK MEMORY
32
BAR VIDEO DISPLAY MONITOR/ ORDER COORDINATOR
28
INPUT/ OUTPUT DEVICE - MAIN FRAME CONTROL
36
CASH REGISTER DISPLAY MONITOR/ INPUT DEVICE
38
KITCHEN VIDEO DISPLAY MONITOR/ ORDER COORDINATOR
24
KITCHEN VIDEO DISPLAY MONITOR/ ORDER COORDINATOR
26

FIG. 1

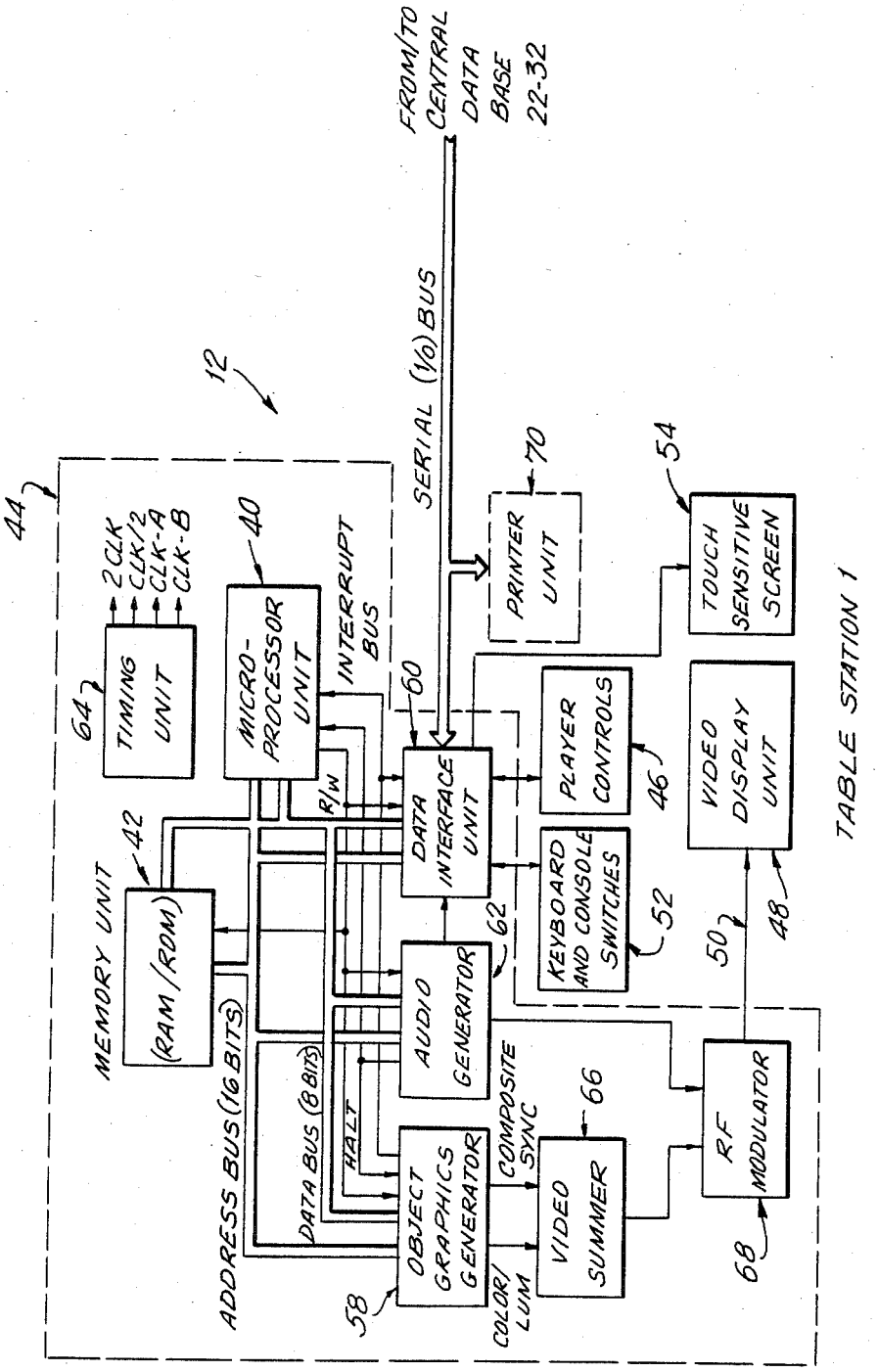
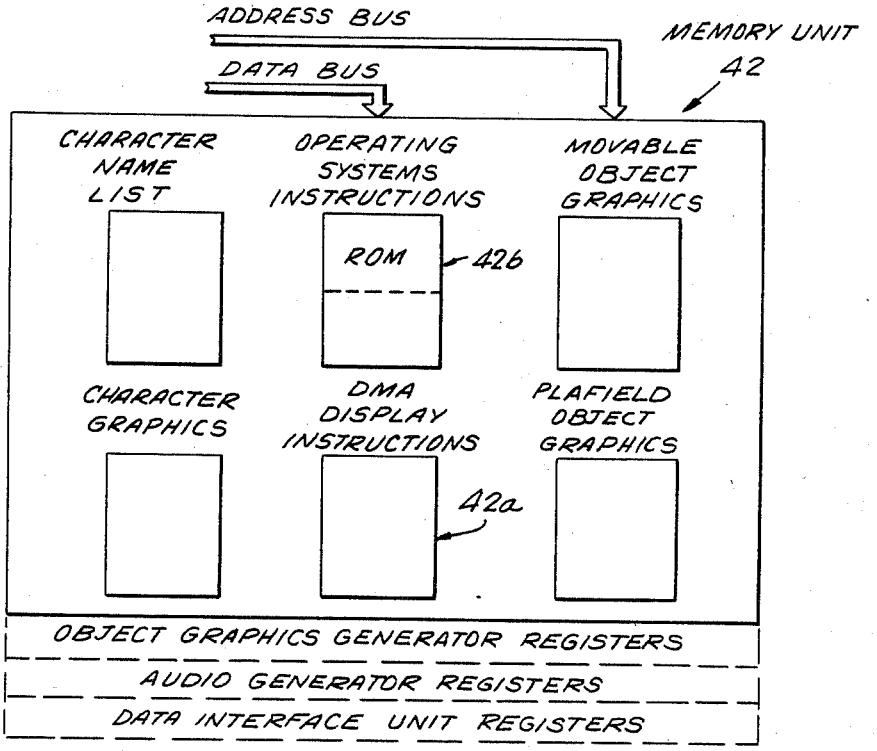

FROM/TO CENTRAL DATA BASE 22-32
SERIAL (1/6) BUS
12
44
2 CLK
CLK/2
CLK-A
CLK-B
64
TIMING UNIT
40
MICRO-PROCESSOR UNIT
INTERRUPT BUS
R/W
60
70
PRINTER UNIT
54
TOUCH SENSITIVE SCREEN
MEMORY UNIT
42
(RAM/ROM)
ADDRESS BUS (16 BITS)
DATA BUS (8 BITS)
HALT
DATA INTERFACE UNIT
PLAYER CONTROLS
46
KEYBOARD AND CONSOLE SWITCHES
52
62
AUDIO GENERATOR
50
VIDEO DISPLAY UNIT
48
OBJECT GRAPHICS GENERATOR
58
COMPOSITE SYNC
COLOR/LUM
66
VIDEO SUMMER
RF MODULATOR
68
TABLE STATION 1

FIG. 2

FIG. 3


ADDRESS BUS
DATA BUS
MEMORY UNIT
42
CHARACTER NAME LIST
OPERATING SYSTEMS INSTRUCTIONS
MOVABLE OBJECT GRAPHICS
ROM
426
CHARACTER GRAPHICS
DMA DISPLAY INSTRUCTIONS
PLAFIELD OBJECT GRAPHICS
42a
OBJECT GRAPHICS GENERATOR REGISTERS
AUDIO GENERATOR REGISTERS
DATA INTERFACE UNIT REGISTERS

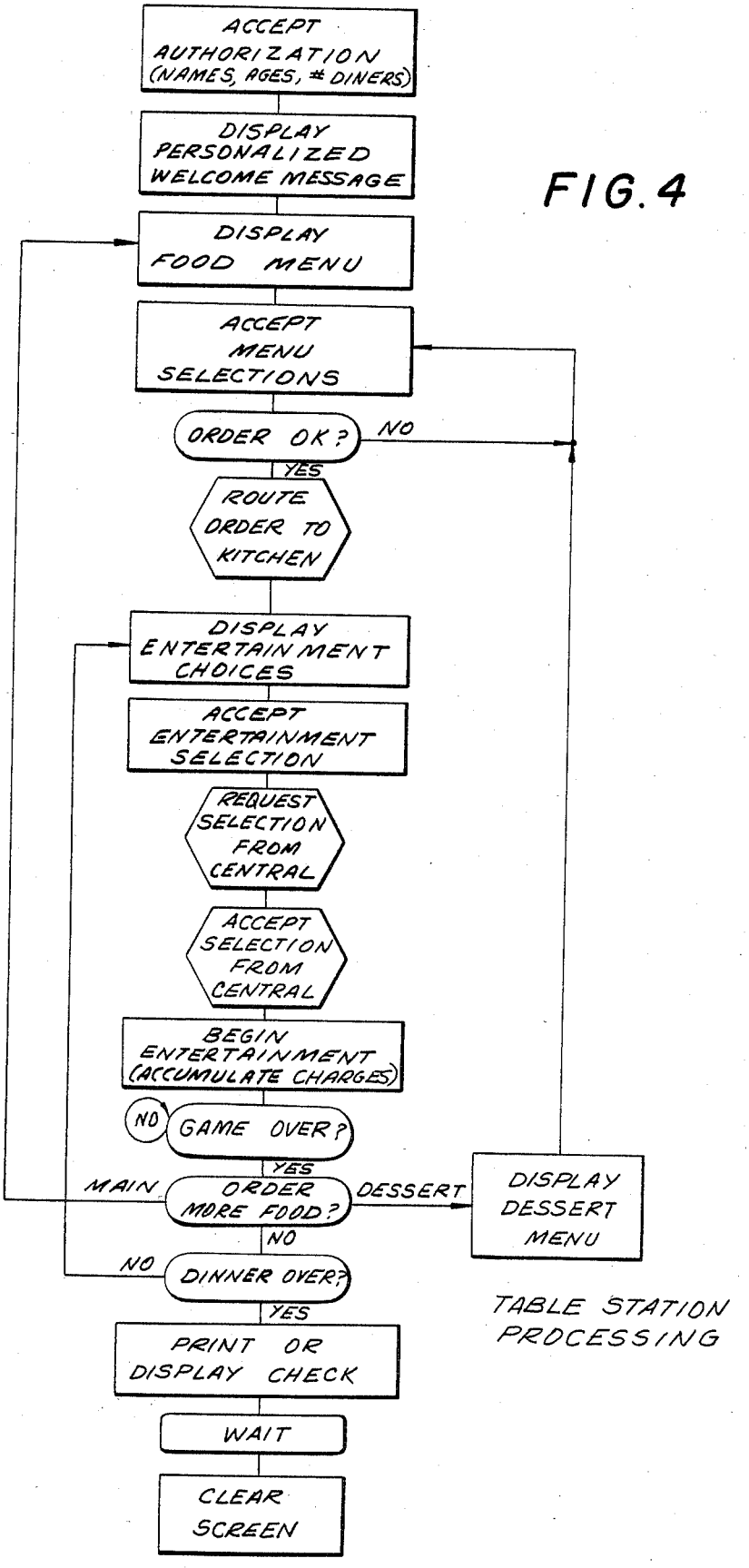
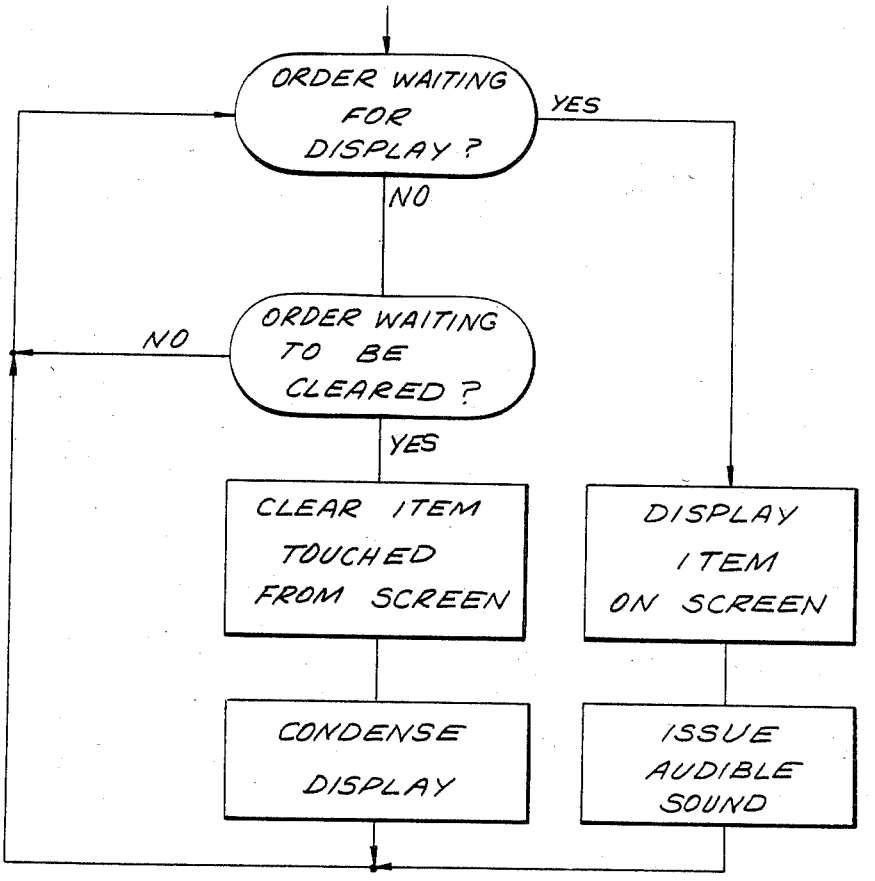

ACCEPT AUTHORIZATION (NAMES, AGES, # DINERS)
DISPLAY PERSONALIZED WELCOME MESSAGE
DISPLAY FOOD MENU
ACCEPT MENU SELECTIONS
ORDER OK?
NO
YES
ROUTE ORDER TO KITCHEN
DISPLAY ENTERTAINMENT CHOICES
ACCEPT ENTERTAINMENT SELECTION
REQUEST SELECTION FROM CENTRAL
ACCEPT SELECTION FROM CENTRAL
BEGIN ENTERTAINMENT (ACCUMULATE CHARGES)
NO
GAME OVER?
YES
MAIN
ORDER MORE FOOD?
DESSERT
DISPLAY DESSERT MENU
NO
NO
DINNER OVER?
YES
PRINT OR DISPLAY CHECK
WAIT
CLEAR SCREEN
TABLE STATION PROCESSING

FIG. 4

FIG. 5



SAMPLE DISPLAY FORMAT:

TIME ON	TIME LOGGED	TABLE	ITEM
7	6:47	21-1	BURGER PLATTER M/R
7	6:47	21-2	CHEFS SALAD RUSS
7	6:47	21-3	ONION RINGS
12	6:42	14-1	BEEF STEW
12	6:42	14-2	VEAL PARMIGIAN SPAG

KITCHEN/BAR PROCESSING

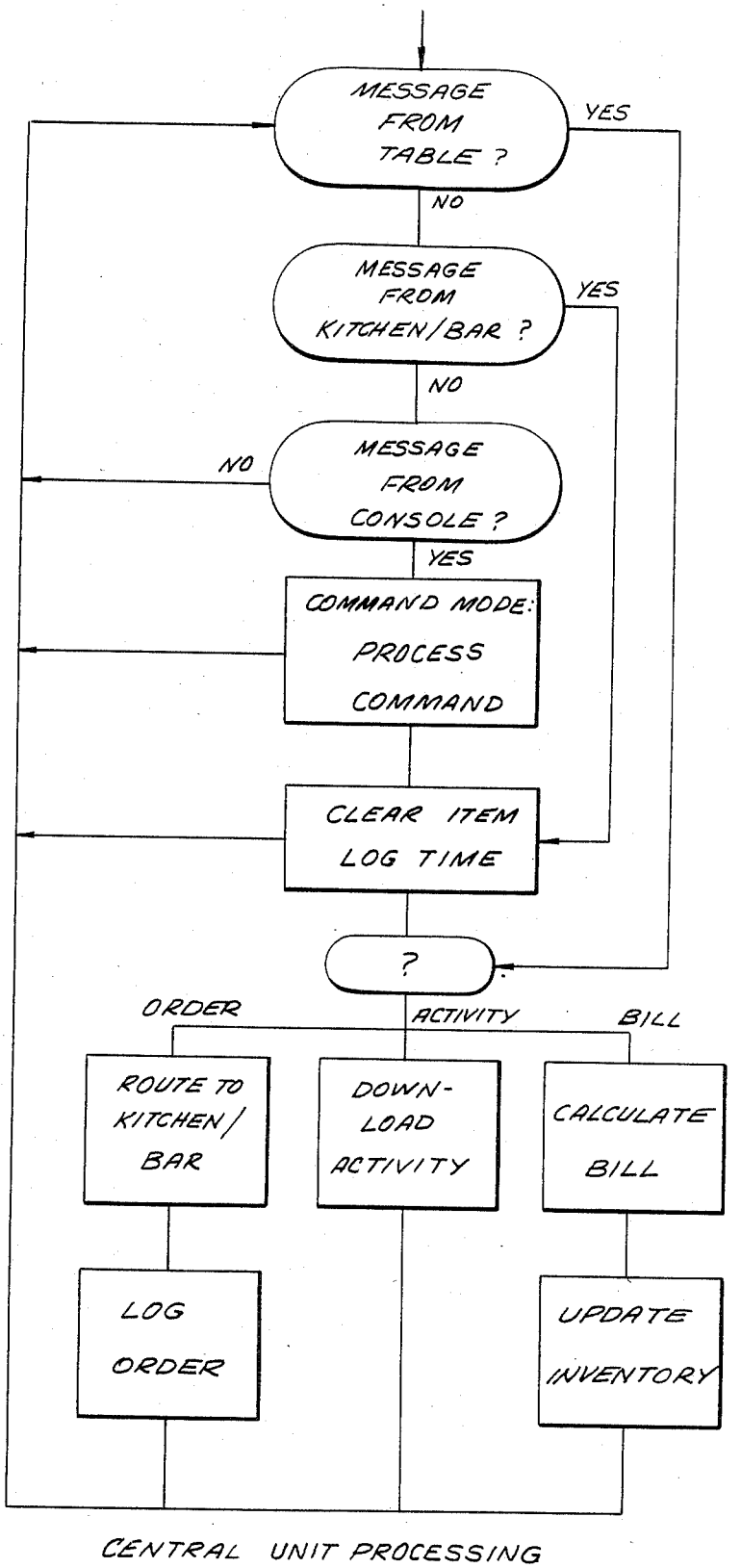

MESSAGE FROM TABLE ?
YES
NO
MESSAGE FROM KITCHEN/BAR ?
YES
NO
MESSAGE FROM CONSOLE ?
NO
YES
COMMAND MODE: PROCESS COMMAND
CLEAR ITEM LOG TIME
?
ORDER
ACTIVITY
BILL
ROUTE TO KITCHEN/ BAR
DOWN-LOAD ACTIVITY
CALCULATE BILL
LOG ORDER
UPDATE INVENTORY
CENTRAL UNIT PROCESSING

FIG. 6

FIG. 7

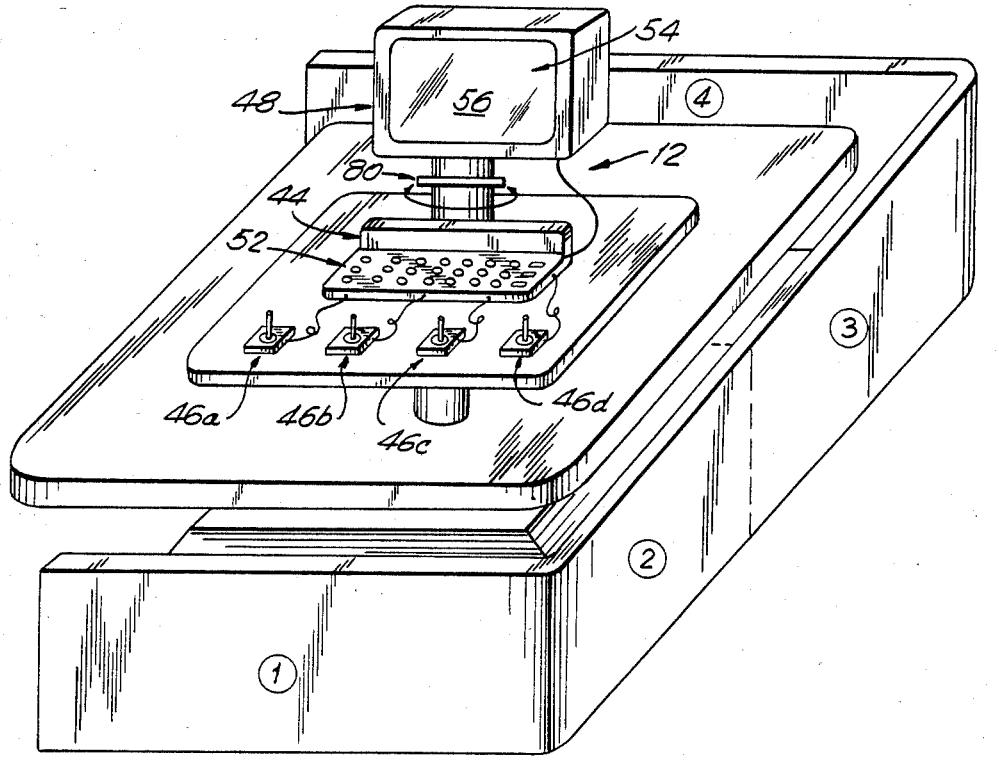


TABLE STATION 1

INTEGRATED INTERACTIVE RESTAURANT COMMUNICATION METHOD FOR FOOD AND ENTERTAINMENT PROCESSING

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to the commonly owned copending U.S. patent application entitled "Integrated Interactive Restaurant Communication System for Food and Entertainment Processing", filed Mar. 14, 1983, bearing U.S. Ser. No. 474,984, and naming Lawrence G. Kurland, the named inventor herein, and Eli Gilbert as joint inventors, the contents of which is specifically incorporated by reference herein in its entirety.

TECHNICAL FIELD

The present invention relates to interactive communication systems and particularly to interactive restaurant communication systems.

BACKGROUND ART

Interactive communication systems, such as those employing video display terminals, are well known in the art, such as disclosed, by way of example, in U.S. Pat. Nos. 4,064,490; 4,054,911; 4,296,476; 4,143,360; 4,191,956; 4,122,519; 3,903,402; 4,001,807; 4,001,785; 4,075,686; 4,084,229; 4,251,291; 3,746,780; 3,668,307; 3,836,888; 3,752,908; 3,691,295; 3,968,327; 4,008,369; 4,028,733; 3,757,225; 3,814,841; 4,117,605; 4,264,925 and 4,164,024. Today, this field is a very active one, particularly with the advent and wide spread use of microprocessors in many business applications, as well as in personal computers, such as disclosed in U.S. Pat. No. 4,296,476. These prior art interactive systems have expanded dramatically in the field of cable television, such as disclosed in many of the aforementioned exemplary patents, both in two-way communication systems, such as disclosed in U.S. Pat. Nos. 3,691,295 and 3,668,307 by way of example, and in one-way cyclical transmission systems such as disclosed in U.S. Pat. Nos. 4,064,490 and 4,054,911 by way of example, as well as in real-time interactive systems such as disclosed in U.S. Pat. No. 4,264,925 by way of example. Moreover, point-of-sale terminals have become almost a total replacement for the cash register in today's businesses. The restaurant business, however, has been very slow to modernize in the area of service to the public except for modern fast-food operations, still relying on separate waitress serice for order taking and separate entertainment features such as music boxes or video games, and the computer revolution has really not caught up. Rather, increased efficiency in restaurant service has primarily been focused on better manual systems, such as the one developed at the Tifflin Inn in Denver, Colo. where a food coordinator person in the kitchen would interface between the waitresses, who never left the dining room, and the cook using busboys to transport the orders which were assembled on carts. This system, however, although sucessful, was still a manual system and did not integrate food and entertainment functions. Thus, although cash register type of point-of-sale systems and automatic order entry systems have made in-roads into the restaurant business, as has distributed interactive video game technology, applicant is not aware of any prior art systems which have sucessfully integrated interactive entertainment and food functions so that a plurality of independent table station terminals, via

down-line loading and two-way communication with a central data base, can accomplish, among other things, transmission of orders to the kitchen, independently selectable down-loading of entertainment modules to the table stations for interactive play at the terminals; automatic computation of composite bills for both food and entertainment, and accumulation of detailed information for restaurant management.

These disadvantages of the prior art are overcome by the system of the present invention.

DISCLOSURE OF THE INVENTION

The present invention relates to an interactive integrated restaurant information communication system for enabling both individualized food and entertainment interactive information communication, such as two-way communication over a common transmission media, between a central remote data base and a plurality of different multipurpose table station terminals located at various table stations throughout the restaurant for use by the restaurant patrons for both food selection, based on down-line loaded food menu modules selectably retrieved from the central data base, and entertainment selection and interactive play, such as video games, also based on down-line loaded entertainment modules selectably retrieved from the central data base. The charges, where applicable, for both the entertainment and food selections can be compositely automatically billed to the table station, with each patron having a unique identification code for billing and/or service purposes, and the bills printed either centrally or at each table station.

The central data base comprises a central main computer which essentially performs the information routing functions, and remote retrievable storage for storing the various food and entertainment programs or sets of control instructions which are retrieved by the various table station terminals in response to selections made by the restaurant patrons, as well as handling coordination or processing and display of food orders in conjunction with kitchen and bar monitors, accumulation of restaurant management information and billing as well as other functions, if desired. Each of the table station terminals comprises a microprocessor and local storage which is down-line loaded with selected sets of control instructions from the central data base, under control of a master control program, in response to patron selection, and a local video display which is utilized to display data for food and entertainment selection as well as to interactively play the game or entertainment selected, with the microprocessor processing incoming data to enable food and entertainments selections to be transmitted to the central data base and to enable the retrieved entertainment to be interactively played at the terminal in response to the retrieved locally stored selected set of control instructions. The food orders, under control of the central computer, are collected from the various table station terminals, and displayed on central kitchen and bar monitors, with the orders being cleared from the monitor screen, such as by using touch-sensitive screens, as they are filled for each table station. If desired, the orders can be assembled by table station, each having its unique patron identification code, and transported to the pertinent table station. Thus, each table station terminal can independently serve to provide both food selection and entertainment functions, interactively with a central data base, in the

integrated restaurant communication system of the present invention, whereby the overall efficiency of the restaurant will be enhanced and better controlled.

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1** is an overall functional block diagram of the presently preferred embodiment of the integrated interactive restaurant communication system of the present invention for providing integrated food and entertainment processing in accordance with the present invention;

FIG. **2** is an overall schematic block diagram of a typical table station terminal for use in the system of FIG. **1**;

FIG. **3** is an illustrative diagram of a typical organization of the memory unit portion of the terminal of FIG. **2**;

FIG. **4** is a condition responsive logic flow diagram of a typical table station terminal processing control program for the terminal of FIG. **2**;

FIG. **5** is a condition responsive logic flow diagram of a typical kitchen or bar monitor processing control program for a typical kitchen or bar monitor in the system of FIG. **1**;

FIG. **6** is a condition responsive logic flow diagram of a typical processing control program for the central computer of the system of FIG. **1**; and

FIG. **7** is a diagrammatic perspective illustration of a typical table station in the system of FIG. **1**.

BEST MODE FOR CARRYING OUT THE INVENTION

Referring now to the drawings in detail, and initially to FIG. **1** thereof, an overall functional block diagram of the presently preferred embodiment of the integrated interactive restaurant communication system, generally referred to by the reference numeral **10**, of the present invention is shown. As will be described in greater detail hereinafter, the interactive restaurant communication system **10** provides integrated food and entertainment processing which enables restaurant patrons to obtain menus for individual food selection on a video monitor at their tables, individually enter their orders into a table station "intelligent" terminal at their tables, select from and interactively play a variety of remotely retrievable interactive entertainment activities using the video monitor while waiting for the food to arrive, and if desired, where applicable, have the food and entertainment charges automatically added to a composite bill which may be printed at the table station terminal or at a remote central location. Moreover, as will be further described herein, the system **10** of the present invention is preferably integrated into the overall restaurant operation so that the food orders placed at the various table station terminals, with five such table station terminals **12**, **14**, **16**, **18** and **20** being shown by way of example in FIG. **1**, can be collected and routed by a central computer **22** to the kitchen and bar for display on video terminals or monitors **24**, **26** and **28**, respectively, for the cook and bartender to enable the orders to be promptly filled, which orders, as illustrated in FIG. **5**, can also be displayed by time logged into the system **10** to enable the oldest orders to be filled first. In addition, restaurant efficiency can also be monitored by comparing the time the order is logged with the time the order is filled.

As shown and preferred in FIG. **1**, each of the table station terminals **12**, **14**, **16**, **18**, **20**, which will be described in greater detail hereinafter with reference to FIGS. **2–4** and **7**, is preferably connected via a conventional two-way communications link, such as conventional RS-232 lines, to the central computer or processor **22** through a conventional multiplexed communication port **30**. The quantity of table station terminals **12**, **14**, **16**, **18**, **20**, which may be serviced or controlled by the central computer **22** is essentially limited primarily by the capacity of the central computer **22** and associated memory **32** which is selected, the desired functions to be performed and the desired access time, all of which are considerations within the ordinary skill of the art based on the system **10** description provided herein. The central processor **22** and associated memory **32**, such as a conventional disk memory, comprise the central or remote data base **22–32** which is remotely accessed by the various table station terminals **12**, **14**, **16**, **18**, **20** in accordance with the present invention. By way of example, the central processor **22**, which is selected to be compatible with the table station terminals **12**, **14**, **16**, **18**, **20** employed, may be one having the capabilities of a PDP 11/23 processor, available from Digital Equipment Corporation, having, by way of example, 128K of memory, with the associated memory **32** being, by way of example a data storage medium of the type such as available from Data Systems Design as its DSD 880 combination flexible and Winchester disk drive. In addition, by way of example, the communications multiplexers **30** may be of the type such as the DLV 11. As further shown and preferred in FIG. **1**, a printer **34**, such as a conventional dot matrix printer, such as an Anadex 9625A, is connected to the central processor **22** for printing bills, if desired, as well as restaurant management information and any other reports desired, such as market survey data. In addition, a conventional master console terminal **36**, such as an ADM-3A, is preferably connected to the central processor **22** to enable, for example, review of restaurant operations and changes to be made in the data base **22–32**. If desired, a conventional type of cash register point-of-sale terminal **38** may also be connected to the central processor **22**.

With respect to the kitchen terminals **24**, **26** and bar terminal **28** employed with the central processor **22**, these terminals **24**, **26**, **28** are preferably conventional and employ touch-sensitive screens, such as the Elographics E270, placed over the video monitor screen, to enable keyboard free data entry, such as, by way of example, to clear an item from the screen after the order has been filled or to indicate order status. The associated processing to be performed by these conventional data terminal **24**, **26**, **28**, which are also selected to be compatible with the central processor **22** employed, if desired, may preferably be limited to requested order display and clearing of filled orders in response to data entry, such as via the aforementioned associated touch sensitive screen, with central computer **22** logging the time that the order is cleared from the screen if desired. Such an exemplary arrangement is illustrated in FIG. **5** which is a self-explanatory condition responsive logic flow diagram of a typical kitchen or bar monitor or terminal processing control program for a typical kitchen or bar terminal **24**, **26**, or **28**, respectively, in the system **10** of the present invention, which may be conventionally programmed by one or ordinary skill in the art based on the description herein. It should be noted that although only two kitchen terminals **24**, **26** and one bar terminal **28** are shown by way of example in the system of FIG. **1**, any desired number of such terminals

may be employed dependent upon the needs of the restaurant and the selected capacity of the system **10**.

Referring now to FIGS. **2–4** and **7**, a typical preferred table station terminal, such as terminal **12** located at table station **1** in the above example, shall now be described in greater detail. As was previously referred to, the table station terminals **12**, **14**, **16**, **18**, **20** are selected to be compatible with the central processor **22** selected. In this regard, preferably the table station terminals are functionally similar to the type marketed, by way of example, by Atari, Inc. as its model 800, or such as the type described in U.S. Pat. No. 4,296,476, the contents of which is specifically incorporated by reference herein in its entirety, conventionally modified to provide a data communications interface or capability with the remote data base **22–32**. In this regard, as shown and preferred in FIG. **2**, which is essentially the type of terminal described in U.S. Pat. No. 4,296,476, which is a terminal employing a microprocessor **40** based data processor having a programmable graphics generator, the peripheral memory devices, such as a local disk drive unit and cassette unit, have been replaced by the down-line loading capability of the terminal **12** from the remote data base **22–32** into the local random access memory or RAM portion **42***a* of the terminal's memory unit **42** under control of a master control program or executive or supervisory program, such as illustrated by way of example in FIG. **4**, stored in the local read only memory or ROM portion **42***b* of the terminal memory unit **42**. The capacity of the ROM and RAM portions of terminal memory unit **42** is preferably conventionally selected so as to permit the desired functions to be performed by the table station terminal **12**.

As shown and preferred in FIGS. **2** and **7**, the table station terminal **12** includes a console **44**, game control apparatus **46**, such as the four conventional video game joystick controllers **46***a–46d* illustrated in FIG. **7**, and a video display unit **48**, which preferably is a conventional television monitor with console **44** providing a suitable radio frequency signal corresponding to a television raster scan signal to the video display unit **48** via line **50**. The table station terminal **12** preferably has two basic modes of operation, the food selection mode and the entertainment mode in which the terminal **12** can function, by way of example, as an interactive video game unit. In the entertainment mode, as will be described in greater detail hereinafter, the table station terminal can be programmed, via down-line loading from the central data base **22–32**, to provide various interactive entertainment activities such as interactive video games or interactive educational materials, such as involving text, diagrams and pictures displayed on video diaplay unit **48**, as well as audio. An interactive dialogue can be conducted between the restaurant patrons seated at the table station terminal **12**, such as at table station **1** in the above example, using a conventional keyboard **52** and/or, if desired, a conventional touch sensitive screen **54**, such as the previously mentioned Elographics E270, which could, if desired, preferably be placed over the normal video screen **56**, to provide keyboard-free data entry. In either mode, the food selection mode or entertainment mode, the table station terminal is utilized by the restaurant patrons to store or retrieve information from the remotely located central data base **22–32**. The video display unit **48** provides the restaurant patrons located at the table station **1** with graphics information, such as an alphanumeric display and/or pictorial graphics, such as for games, that is conventionally formulated and transmitted to the video display unit **48** by the electronics contained in console **44** via the communicating line **50**.

Apart from the master control program or executive or supervisory program for conventionally supervising the overall operation of data manipulation in the table station terminal **12**, which is preferably permanently stored in ROM in memory unit **42**, the desired operating programs or modules for providing the food selection and entertainment functions of the multipurpose table station terminal **12** are down-line loaded from the central data base **22–32** for storage in the RAM section **42***a* of memory unit **42** in response to a request therefor which has been input by the restaurant patron via the keyboard **52**, or touch sensitive screen **54** if desired. Regardless of the mode in which the table station terminal **12** operates, the operation of the associated internal circuitry illustrated in FIG. **2** remains essentially the same. Thus, the operating program or module, whether a food menu module used for food selection or order entry, or an entertainment module used for entertainment selection and interactive game or other entertainment activity, can display portions of the requisite information or data on video display unit **48**. Preferably, the table station terminal **12** includes a conventional programmable object graphics generator **58** which can be called upon to transfer graphics information from memory unit **42** to the video display unit **48**. In this regard, the conventional microprocessor unit **40**, under direction of the down-line loaded operating program or module, transfers a list of display instructions into the RAM section **42***a* (FIG. **3**) of the memory unit **42**, making the display instructions available to the microprocessor **40**. The microprocessor **40** can modify portions of the display instructions, such as the addresses of the various instructions, thereby directing the object graphics generator **58** to those sections of the memory unit **42** containing the graphics information to be displayed, such as alphanumeric characters, lines, heading marks and the like. Alternatively, such as in the entertainment mode, the down-line loaded operating program, such as a down-line loaded entertainment module for an interactive video game, may require information to be displayed in graph-like form, such as a playfield display in the form of a cartesian or other coordinate system displayed to the restaurant patron(s) at the table station **1** via video display unit **48**. Further, the down-line loaded operating program may call for a moveable cursor to be displayed in which instance the operating program would contain a block of graphics information containing the picture data for the vertical column that the movable object generator **58** constructs on the video display unit **48**, including the picture data for the cursor. The microprocessor would then conventionally write into a movable object DMA counter the address of the location within the memory unit **42** of the block of graphics information containing the cursor picture data. The microprocessor **40** would also preferably write a data word to a DMA control register whose contents are used to conventionally inform a DMA control unit that movable object graphics will be displayed. Accordingly, as described in detail in U.S. Pat. No. 4,296,476 specifically incorporated by reference herein, the DMA control unit provides the moveable object DMA counter with signals that cause the DMA counter to sequentially address the memory locations of memory unit **42** containing the graphics information for

the cursor. The data used by the microprocessor **40** in the entertainment mode and/or with respect to cursor movement, in either mode, is preferably provided by the joysticks **46***a*–**46***d* or operation of the keyboard **52**, or the touch-sensitive screen **56**, by the restaurant patrons seated at the table station **1**. Typically, the joysticks **46***a*–**46***d*, assuming by way of example one for each of up to four restaurant patrons to be seated at the table station **1**, provide user generated position information, and possibly other game information, which is communicated to the microprocessor **40** via a conventional data interface unit **60**. The conventional joysticks **46***a*–**46***d* may typically be of the type disclosed in U.S. Pat. No. 4,091,234, the teachings of which are specifically incorporated by reference herein in its entirety. In addition to the above, as further shown and preferred in FIG. **2**, the terminal **12** also includes a conventional audio generator **62** for generating audio signals, a conventional timing unit **64** for generating the various timing signals required such as those illustrated, by way of example, for use in the system of U.S. Pat. No. 4,296,476, a conventional video summer circuit **66** for conventionally summing the color/luminance and composite sync signal output of the object graphics generator **58**, and a conventional RF modulator **68** which provides the displayable video signal to the video display unit **48** via line **50**. Furthermore, an optional conventional printer unit **70** may also be provided at the table station for local printing of bills, or other data if desired, with the printer **70** being connected to the microprocessor **40** via the data interface unit **60** and a serial (I/O) bus through which the terminal **12** is connected to the central data base **22–32**. It should be noted that after the desired food menu module or operating program or entertainment module or operating program is requested by and down-line loaded to the terminal **12**, the microprocessor **40** can then process incoming data provided via the keyboard **52**, player controls **46** or touch sensitive screen **56** in accordance therewith so that food orders may be selected and transmitted back to the central data base **22–32** for billing and filling and/or interactive games may be played. In this regard, FIG. **3** provides an illustrative diagram of a typical organization of the memory unit **42**, such as described in U.S. Pat. No. 4,296,478, and FIG. **4** provides a self-explanatory condition responsive logic flow diagram of a typical processing control program for the table station terminal **12**, which may be conventionally programmed by one of ordinary skill in the art based on the description herein.

As shown and preferred in FIG. **7**, by way of example, preferably each table station has a unique position identification for each restaurant patron to facilitate accurate order filling and billing. Thus, assuming four restaurant patrons per table station by way of example, the various restaurant patrons at table station **1** illustrated in FIG. **7** would identify themselves to the table station terminal **12** as **1-1**, **1-2**, **1-3** and **1-4**. In addition, by way of example, the video display unit **48** could be mounted on a universal swivel **80** so as to be rotatable for viewing by all of the patrons at the table station **1**, with the keyboard **52** and joysticks **46***a*–**46***d* being connected to the terminal **12** via conventional electronic umbilical cords to facilitate use by each patron at the table station **1**. It should be noted that the table station configuration shown in FIG. **7** is merely illustrative and many other such configurations will readily occur to one of ordinary skill in the art.

With respect to the operation of the central data base **22–32** in connection with the processing of food menu and entertainment requests, including the down-line loading of food menu modules or operating programs and entertainment modules or operating programs to the requesting table station terminals **12**, **14**, **16**, **18**, **20**, the logging and routing of orders to the kitchen and bar terminals **24**, **26**, **28**, the calculation of composite bills, the updating of inventory and the overall processing of messages, a condition responsive logic flow diagram of a typical processing control program for the central computer **22** for accomplishing the above is shown in FIG. **6**. The program itself can conventionally readily be written by one of ordinary skill in the art based on the description herein.

Summarizing the overall operation of the preferred system **10** of the present invention, either after or before the diners or restaurant patrons are seated at their respective table station, the table station terminal would preferably display a welcome message and a list of choices for the patrons. The patrons would then select a choice by touching a location on the touch sensitive screen or by pressing keys on the keyboard. The initial choices may be type of food, type of meal, etc. Thereafter, by stepping through various displays, the patrons, using their unique position identification code, could order their meals which would be transmitted to the central data base for billing and filing. The patrons would then be allowed to select entertainment such as games, educational programs, computer generated art displays, etc., and possibly television channels if desired by using a television receiver as the video display unit. Associated charges for both food and entertainment, where applicable, would automatically be added to a composite bill. The central computer downloads the selected activity to the requesting table station terminal, with each terminal being capable of independently requesting and receiving its own operating programs from the central data base. The patrons would then be free to interact with the downloaded operating program, whether it were a food menu module or an entertainment module. If desired, an initialization program could be run daily, or at any time, to set price changes, menu changes, entertainment changes, current time and date, enter deliveries, enter daily specials, etc.

By utilizing the system of the present invention, restaurant efficiency and flexability are enhanced and better control can be achieved.

It is to be understood that the above described embodiments of the invention are merely illustrative of the principles thereof and that numerous modifications and embodiments of the invention may be derived within the spirit and scope thereof.

What is claimed is:

1. A method of interactively integrating food and entertainment patron selection and interaction for a plurality of table stations in a restaurant communication system comprising the steps of remotely retrievably storing and processing information comprising a plurality of displayable food menu modules and entertainment modules at a central data base, providing a plurality of down-line loadable multipurpose table station terminals at a plurality of said table stations, each of said table station terminals comprising microprocessor means, local storage means, selection and control means and local common video type display means capable of providing a video display of patron selected food menu and entertainment modules, each of said retrievably

stored food and entertainment modules comprising a different set of control instructions corresponding to a different patron selectable food menu or entertainment function, said microprocessor means being operable in accordance with a locally stored one of said selected sets of control instructions;

individually patron selecting a displayable remotely stored food menu module via said table station terminal selection means;

down-line loading said patron selected corresponding set of control instructions to the selecting table station terminal for local storage thereof;

locally displaying food menu selection data corresponding to said patron selected food menu module on said common video type display means in response to said down-line loaded patron selected set of control instructions, said video displayable food menu module data comprising menu selection data corresponding to a choice of food items to be ordered; individually patron selecting at least one of said video displayed food items for remotely patron ordering said selected food items; transmitting said at least one patron ordered food selection back to said central data base for central processing of said order; individually patron selecting a displayable remotely stored entertainment module via said food ordering table station terminal for video display on said common video type display means at said food ordering table station terminal; down-line loading said patron selected corresponding set of control instructions to said food ordering table station terminal for local storage thereof; locally video displaying entertainment data corresponding to said patron selected entertainment module on said food ordering table station terminal common video type display means in response to said down-line loaded selected set of control instructions to said food ordering table station terminal for enabling interaction with said video displayed entertainment data while said food order is being processed; whereby each table station terminal may independently provide both interactive entertainment and food selection on a common video type display terminal in cooperation with a central data base.

2. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **1** further comprising the step of compositely billing charges for a given food ordering table station based on said food and entertainment selections of said down-line loaded information.

3. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **2** wherein said composite billing step comprises the step of centrally printing the composite bill for each of said table stations.

4. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **2** wherein said composite billing step comprises the step of locally printing the composite bill for said given table station.

5. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **1** wherein said remote storing step comprises the step of remotely retrievably storing interactive video games as at least a portion of said entertainment modules.

6. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **1** wherein said remote storing step comprises the step of remotely retrievably storing interactive video games as at least a portion of said entertainment modules, said local video display step further comprising the step of locally video displaying a remotely retrieved interactive video game on said food ordering table station terminal common video type display means as said selected entertainment data.

7. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **1** further comprising the step of uniquely identifying each restaurant patron at said food ordering table station and transmitting said unique patron identification to said central data base with at least said food selection information for enabling correlated food distribution based on said central processing thereof.

8. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **7** further comprising the step of compositely billing charges for a given food ordering table station based on said food and entertainment selections of said down-line loaded information.

9. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **1** wherein said central processing step comprises the steps of collecting each food selection from each of said plurality of table station terminals where a selection is made and centrally processing said collected food selections for distribution to said selecting plurality of table stations.

10. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **9** further comprising the step of uniquely identifying each restaurant patron at said food ordering table station and transmitting said unique patron identification to said central data base with at least said food selection information for enabling correlated food distribution based on said central processing thereof.

11. A method of interactively integrating food and entertainment selection for a plurality of table stations in a restaurant communication system in accordance with claim **10** further comprising the step of compositely billing charges for a given food ordering table station based on said food and entertainment selections of said down-line loaded information.

* * * * *

United States Patent [19]

Bennett et al.

[11] **Patent Number: 4,591,983**

[45] **Date of Patent: May 27, 1986**

[54] **HIERARCHICAL KNOWLEDGE SYSTEM**

[75] Inventors: **James S. Bennett; Jay S. Lark,** both of Palo Alto, Calif.

[73] Assignee: **Teknowledge, Inc.,** Palo Alto, Calif.

[21] Appl. No.: **628,817**

[22] Filed: **Jul. 9, 1984**

[51] **Int. Cl.**[4] **G06F 15/24**

[52] **U.S. Cl.** **364/403;** 364/468; 364/478; 235/385; 29/703

[58] **Field of Search** ..... 364/400-401, 364/403, 468-469, 478, 513, 518; 209/1-2, 546, 552; 235/385; 29/33 K, 33 R, 400 R, 400 M, 428-431, 469, 564, 568, 700-703, 711

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,203,204	5/1980	Murphy	29/703
4,310,964	1/1982	Murphy	29/469
4,332,012	5/1982	Sekine et al.	364/468
4,472,783	9/1984	Johnstone et al.	364/478 X
4,484,289	11/1984	Hemond	364/478
4,504,919	3/1985	Fujii et al.	364/478
4,509,123	4/1985	Vereen	364/403 X

OTHER PUBLICATIONS

James Bennett et al., "SACON: A Knowledge-Based Consultant for Structured Analysis," Stanford University Rep. STAN-CS-78-688 (Sep. 1978).

Bennett & Engelmore, "SACON: A Knowledge-Based Consultant for Structured Analysis," *Proc. of the Sixth Int. Joint Conf. on Artificial Intelligence,* Tokyo (Aug. 20-23, 1979) pp. 47-49.

John McDermott, "R1: A Rule-Based Configurer of Computer Systems," *Carnegie-Mellon Univ.,* Rep. CMU-CS-80-119 (Apr. 1980).

W. van Melle et al., *The Emycin Manual,* Stanford University, Rep. STAN-CS-81-855 (Oct. 1981).

Barr & Feigenbaum (eds.), *The Handbook of Artificial Intelligence,* William Kaufmann, Inc. (1982) vol. II, pp. 79-86, 150-154, vol. III, pp. 515-530, 541-556.

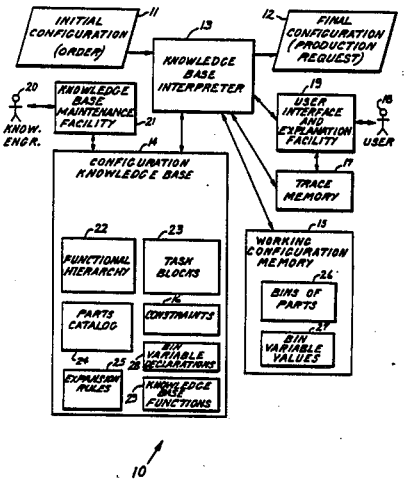
Primary Examiner—Gary V. Harkcom

Attorney, Agent, or Firm—Leydig, Voit & Mayer

[57] **ABSTRACT**

A knowledge system has a hierarchical knowledge base comprising a functional decomposition of a set of elements into subsets over a plurality of hierarchical levels, a plurality of predefined functions or conditions of the elements within the subsets of a plurality of the hierarchical levels, and a predefined set of operations to perform on a user-defined set of elements responsive to the functional knowledge base. Preferably, the knowledge base is defined declaratively by assigning parent sets to offspring subsets to define the hierarchy, by indicating the conditions of the subsets which satisfy the predefined functions and by writing task blocks in an imperative language defining the sequence of operations to perform on the user-defined set of elements. Preferably the operations include matching, configuring and expanding the user-defined set of elements into the defined subsets of individual elements and evaluating the predefined functions, and the operations are executed recursively. In a specific embodiment the elements are available components for a system or item of manufacture, and the subsets of elements are sub-assemblies or functionally related components. The predefined functions define condition-action constraints to insure that the sub-assemblies have compatible components. Such a knowledge system has general applicability, is easy to maintain and incrementally modify, has transparent representation of the functional decomposition and the configuration operations, and provides explanation for an assessment of the configuration.

70 Claims, 13 Drawing Figures



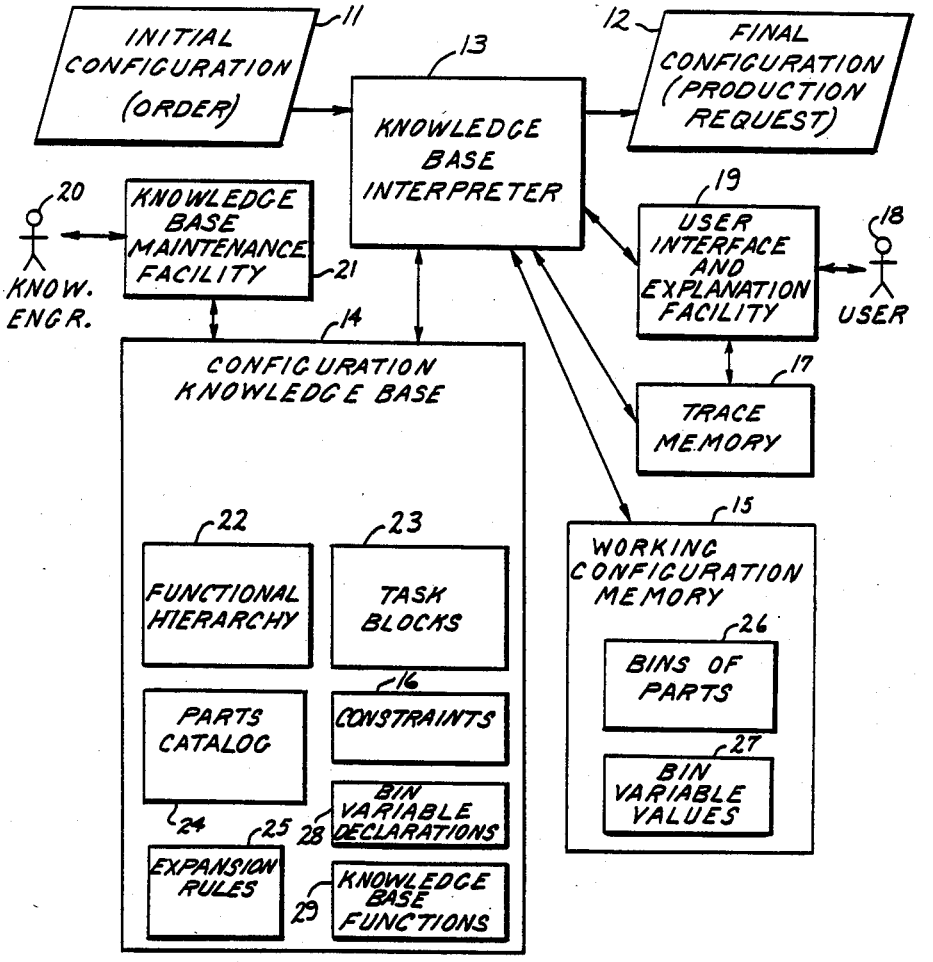

INITIAL CONFIGURATION (ORDER)
11
KNOWLEDGE BASE INTERPRETER
13
FINAL CONFIGURATION (PRODUCTION REQUEST)
12
20
KNOWLEDGE BASE MAINTENANCE FACILITY
21
KNOW. ENGR.
19
USER INTERFACE AND EXPLANATION FACILITY
18
USER
14
CONFIGURATION KNOWLEDGE BASE
17
TRACE MEMORY
15
WORKING CONFIGURATION MEMORY
22
FUNCTIONAL HIERARCHY
23
TASK BLOCKS
26
BINS OF PARTS
16
CONSTRAINTS
PARTS CATALOG
27
BIN VARIABLE VALUES
24
25
28
BIN VARIABLE DECLARATIONS
EXPANSION RULES
29
KNOWLEDGE BASE FUNCTIONS
10

Fig. 1.

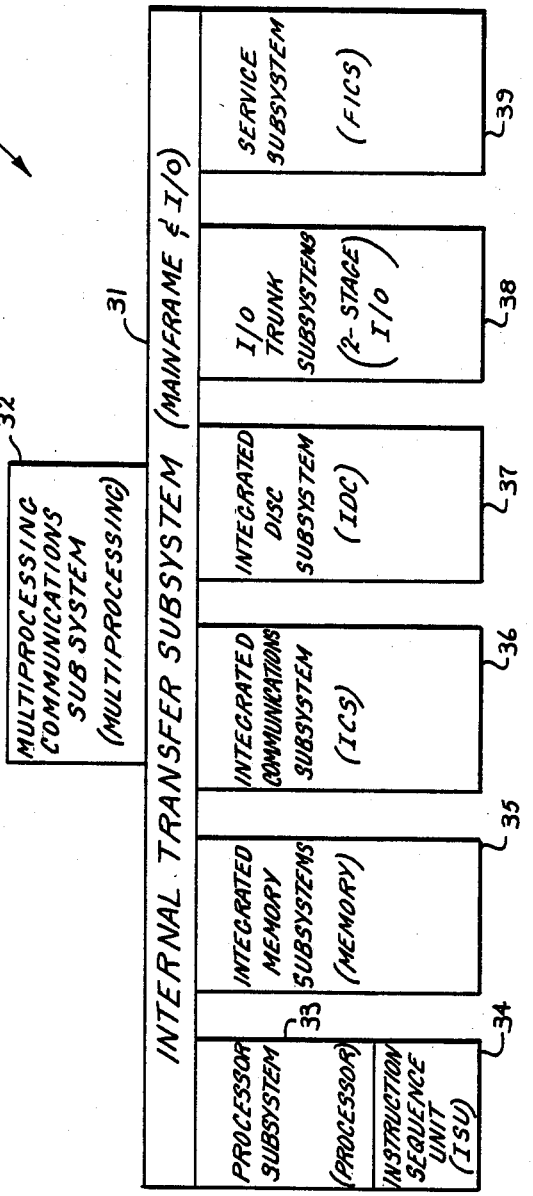

30
31
32
MULTIPROCESSING COMMUNICATIONS SUB SYSTEM (MULTIPROCESSING)
INTERNAL TRANSFER SUBSYSTEM (MAINFRAME & I/O)
PROCESSOR SUBSYSTEM (PROCESSOR)
INSTRUCTION SEQUENCE UNIT (ISU)
33
34
INTEGRATED MEMORY SUBSYSTEMS (MEMORY)
35
INTEGRATED COMMUNICATIONS SUBSYSTEM (ICS)
36
INTEGRATED DISC SUBSYSTEM (IDC)
37
I/O TRUNK SUBSYSTEMS (2- STAGE I/O)
38
SERVICE SUBSYSTEM (FICS)
39

FIG. 2.

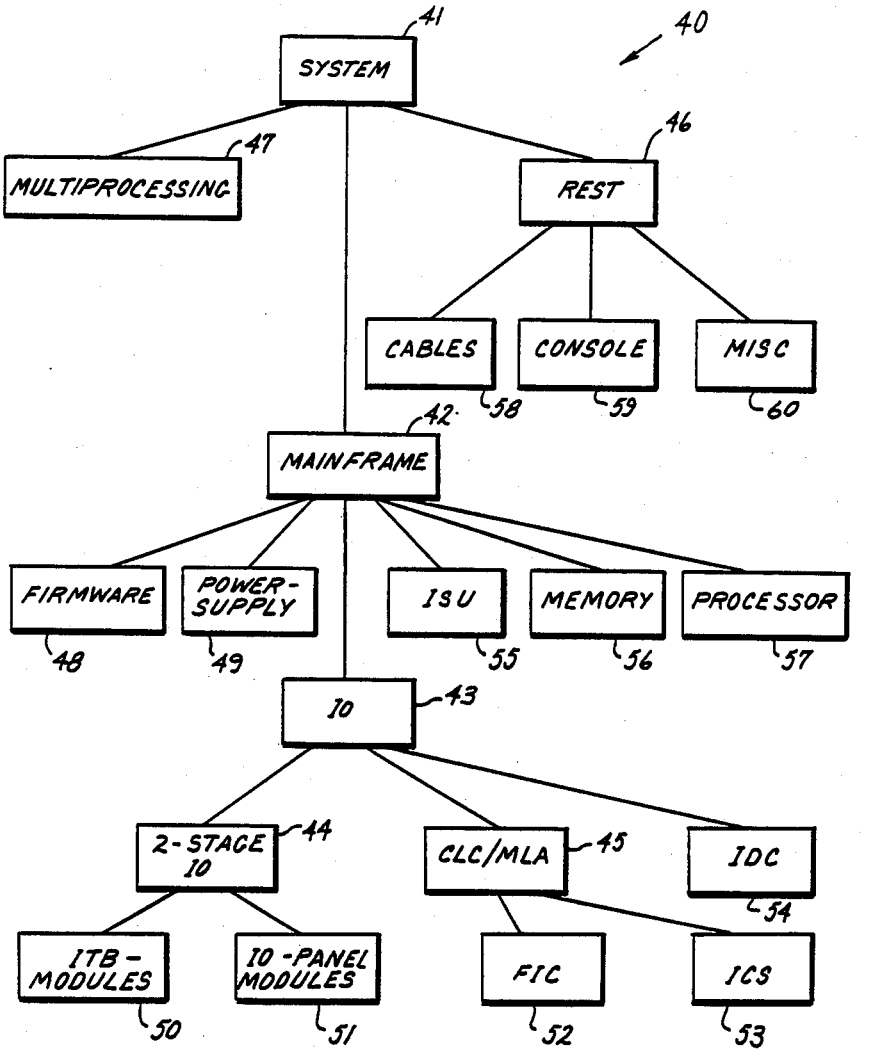

41
40
SYSTEM
47
MULTIPROCESSING
46
REST
CABLES
58
CONSOLE
59
MISC
60
42
MAINFRAME
FIRMWARE
48
POWER-SUPPLY
49
ISU
55
MEMORY
56
PROCESSOR
57
IO
43
2-STAGE IO
44
CLC/MLA
45
IDC
54
ITB-MODULES
50
IO-PANEL MODULES
51
FIC
52
ICS
53

Fig. 3.

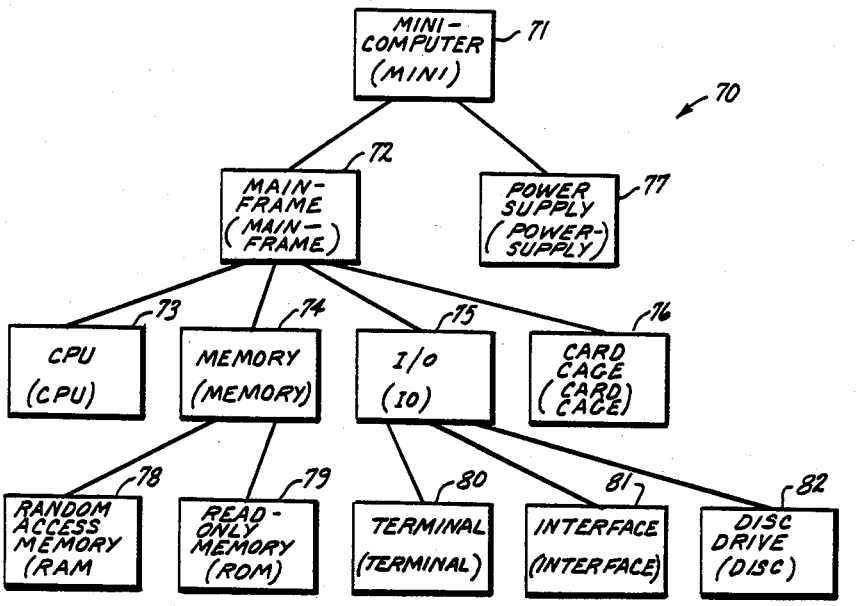

MINI-COMPUTER (MINI)
71
70
MAIN-FRAME (MAIN-FRAME)
72
POWER SUPPLY (POWER-SUPPLY)
77
CPU (CPU)
73
MEMORY (MEMORY)
74
I/O (IO)
75
CARD CAGE (CARD CAGE)
76
RANDOM ACCESS MEMORY (RAM
78
READ-ONLY MEMORY (ROM)
79
TERMINAL (TERMINAL)
80
INTERFACE (INTERFACE)
81
DISC DRIVE (DISC)
82

Fig.4.

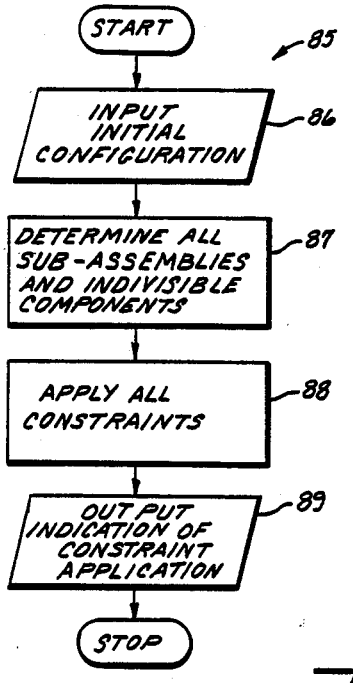

START
85
INPUT INITIAL CONFIGURATION
86
DETERMINE ALL SUB-ASSEMBLIES AND INDIVISIBLE COMPONENTS
87
APPLY ALL CONSTRAINTS
88
OUTPUT INDICATION OF CONSTRAINT APPLICATION
89
STOP

Fig.5.

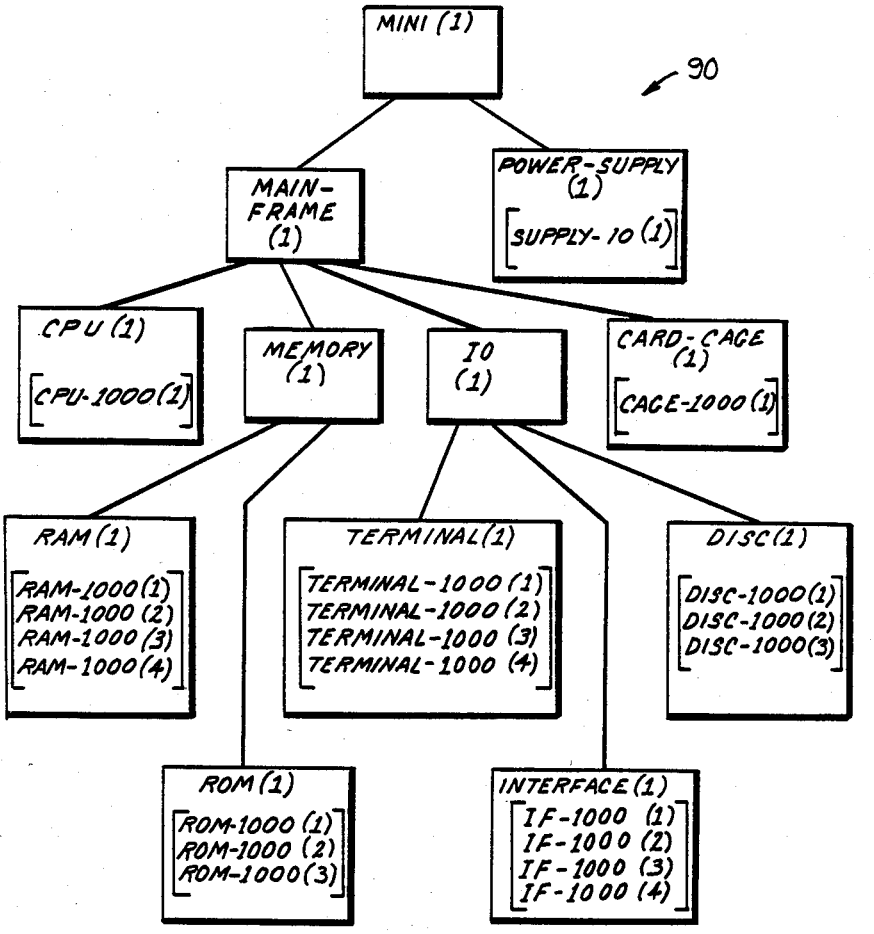

MINI (1)
90
MAIN-FRAME (1)
POWER-SUPPLY (1)
SUPPLY-IO (1)
CPU (1)
CPU-1000 (1)
MEMORY (1)
IO (1)
CARD-CAGE (1)
CAGE-1000 (1)
RAM (1)
RAM-1000 (1)
RAM-1000 (2)
RAM-1000 (3)
RAM-1000 (4)
TERMINAL (1)
TERMINAL-1000 (1)
TERMINAL-1000 (2)
TERMINAL-1000 (3)
TERMINAL-1000 (4)
DISC (1)
DISC-1000 (1)
DISC-1000 (2)
DISC-1000 (3)
ROM (1)
ROM-1000 (1)
ROM-1000 (2)
ROM-1000 (3)
INTERFACE (1)
IF-1000 (1)
IF-1000 (2)
IF-1000 (3)
IF-1000 (4)

Fig. 6.

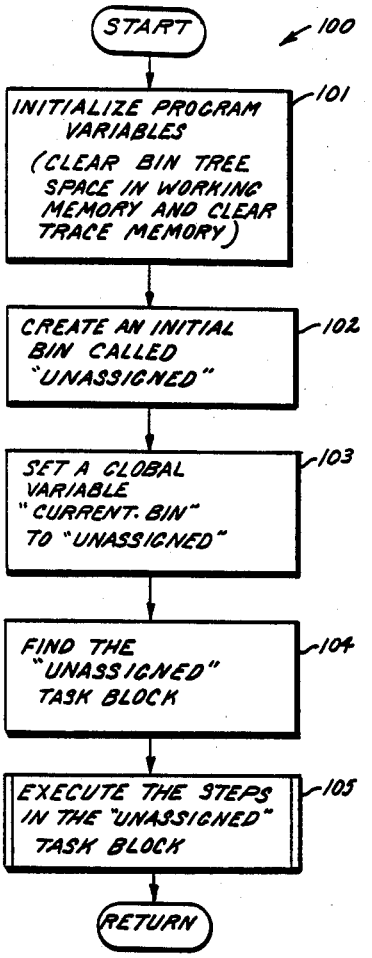

START
100
INITIALIZE PROGRAM VARIABLES (CLEAR BIN TREE SPACE IN WORKING MEMORY AND CLEAR TRACE MEMORY)
101
CREATE AN INITIAL BIN CALLED "UNASSIGNED"
102
SET A GLOBAL VARIABLE "CURRENT-BIN" TO "UNASSIGNED"
103
FIND THE "UNASSIGNED" TASK BLOCK
104
EXECUTE THE STEPS IN THE "UNASSIGNED" TASK BLOCK
105
RETURN

Fig. 7.

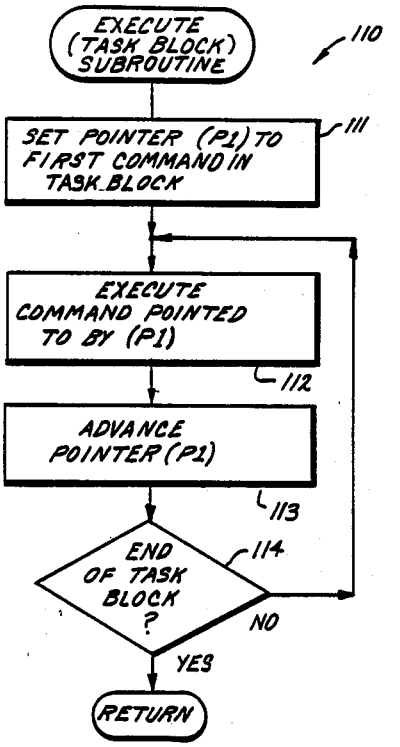

EXECUTE (TASK BLOCK) SUBROUTINE
110
SET POINTER (P1) TO FIRST COMMAND IN TASK BLOCK
111
EXECUTE COMMAND POINTED TO BY (P1)
112
ADVANCE POINTER (P1)
113
END OF TASK BLOCK ?
114
NO
YES
RETURN

Fig. 8.

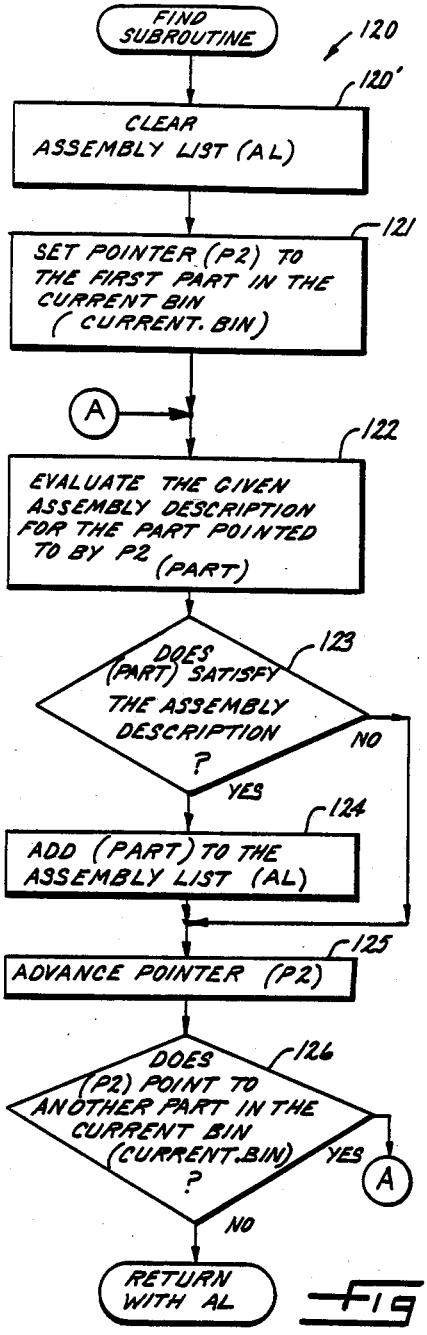

FIND SUBROUTINE
120
120'
CLEAR ASSEMBLY LIST (AL)
121
SET POINTER (P2) TO THE FIRST PART IN THE CURRENT BIN (CURRENT.BIN)
A
122
EVALUATE THE GIVEN ASSEMBLY DESCRIPTION FOR THE PART POINTED TO BY P2 (PART)
123
DOES (PART) SATISFY THE ASSEMBLY DESCRIPTION ?
NO
YES
124
ADD (PART) TO THE ASSEMBLY LIST (AL)
125
ADVANCE POINTER (P2)
126
DOES (P2) POINT TO ANOTHER PART IN THE CURRENT BIN (CURRENT.BIN) ?
YES
A
NO
RETURN WITH AL

Fig. 9.

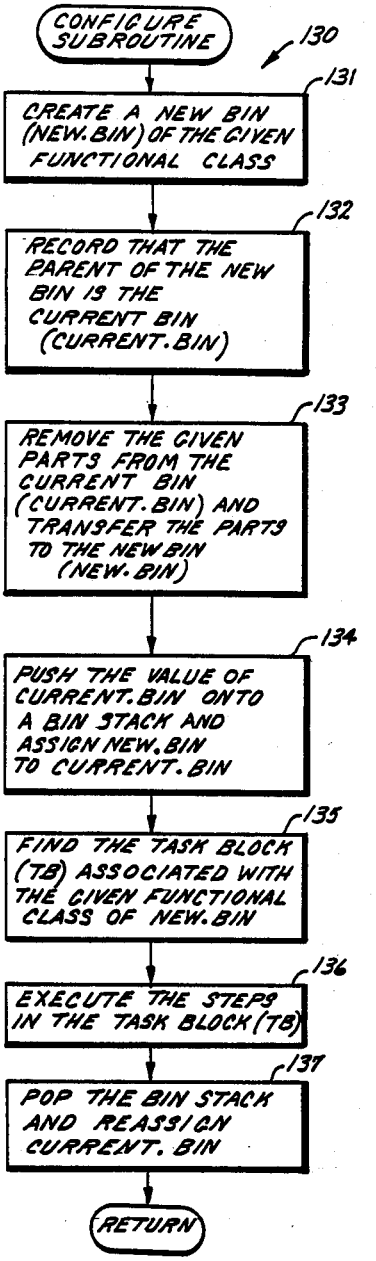

CONFIGURE SUBROUTINE
130
131
CREATE A NEW BIN (NEW.BIN) OF THE GIVEN FUNCTIONAL CLASS
132
RECORD THAT THE PARENT OF THE NEW BIN IS THE CURRENT BIN (CURRENT.BIN)
133
REMOVE THE GIVEN PARTS FROM THE CURRENT BIN (CURRENT.BIN) AND TRANSFER THE PARTS TO THE NEW BIN (NEW.BIN)
134
PUSH THE VALUE OF CURRENT.BIN ONTO A BIN STACK AND ASSIGN NEW.BIN TO CURRENT.BIN
135
FIND THE TASK BLOCK (TB) ASSOCIATED WITH THE GIVEN FUNCTIONAL CLASS OF NEW.BIN
136
EXECUTE THE STEPS IN THE TASK BLOCK (TB)
137
POP THE BIN STACK AND REASSIGN CURRENT.BIN
RETURN

Fig. 10.

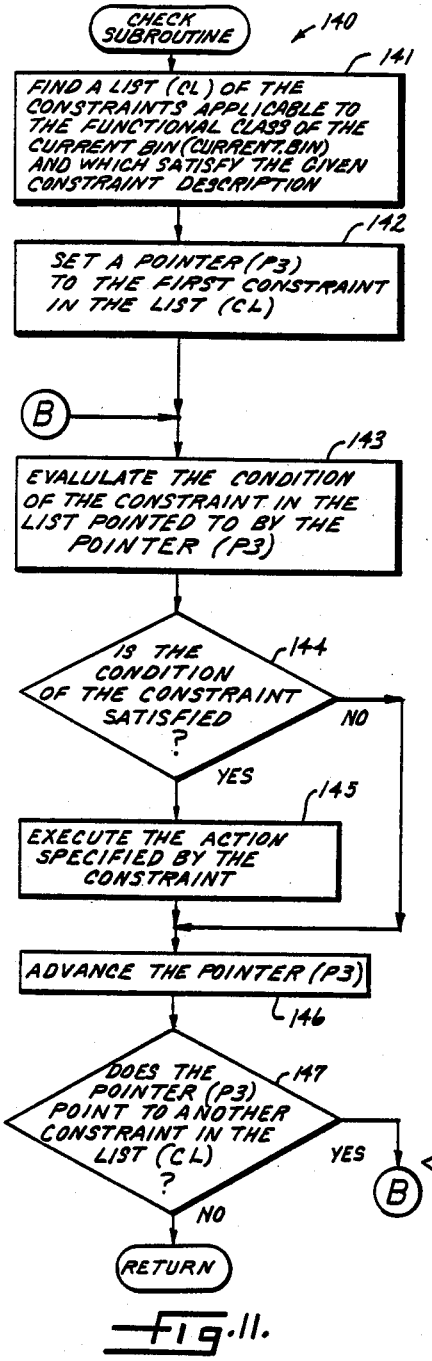

CHECK SUBROUTINE
140
141
FIND A LIST (CL) OF THE CONSTRAINTS APPLICABLE TO THE FUNCTIONAL CLASS OF THE CURRENT BIN (CURRENT.BIN) AND WHICH SATISFY THE GIVEN CONSTRAINT DESCRIPTION
142
SET A POINTER (P3) TO THE FIRST CONSTRAINT IN THE LIST (CL)
B
143
EVALULATE THE CONDITION OF THE CONSTRAINT IN THE LIST POINTED TO BY THE POINTER (P3)
144
IS THE CONDITION OF THE CONSTRAINT SATISFIED ?
NO
YES
145
EXECUTE THE ACTION SPECIFIED BY THE CONSTRAINT
ADVANCE THE POINTER (P3)
146
147
DOES THE POINTER (P3) POINT TO ANOTHER CONSTRAINT IN THE LIST (CL) ?
YES
B
NO
RETURN

Fig. 11.

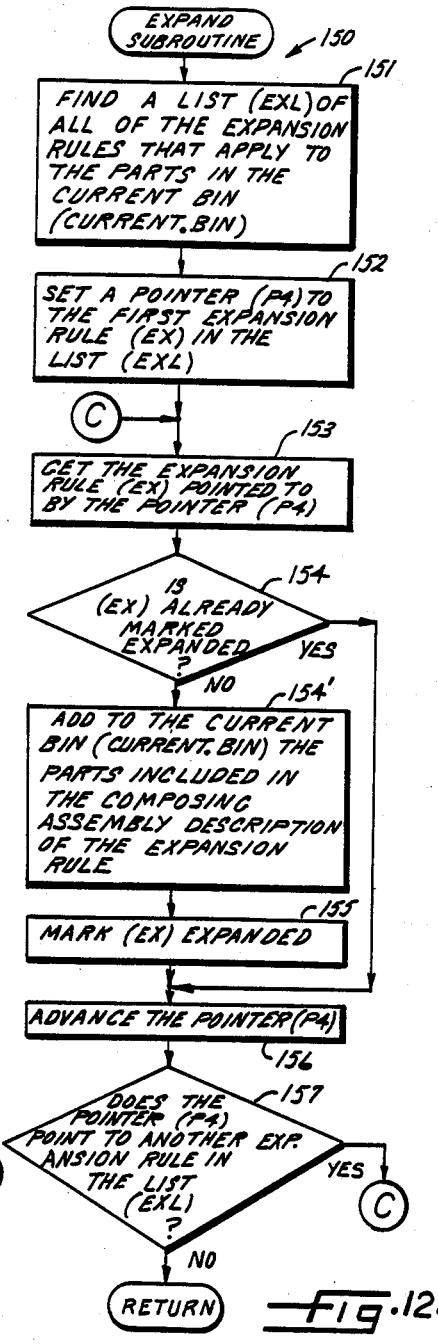

EXPAND SUBROUTINE
150
151
FIND A LIST (EXL) OF ALL OF THE EXPANSION RULES THAT APPLY TO THE PARTS IN THE CURRENT BIN (CURRENT.BIN)
152
SET A POINTER (P4) TO THE FIRST EXPANSION RULE (EX) IN THE LIST (EXL)
C
153
GET THE EXPANSION RULE (EX) POINTED TO BY THE POINTER (P4)
154
IS (EX) ALREADY MARKED EXPANDED ?
YES
NO
154'
ADD TO THE CURRENT BIN (CURRENT.BIN) THE PARTS INCLUDED IN THE COMPOSING ASSEMBLY DESCRIPTION OF THE EXPANSION RULE
155
MARK (EX) EXPANDED
ADVANCE THE POINTER (P4)
156
157
DOES THE POINTER (P4) POINT TO ANOTHER EXPANSION RULE IN THE LIST (EXL) ?
YES
C
NO
RETURN

Fig. 12.

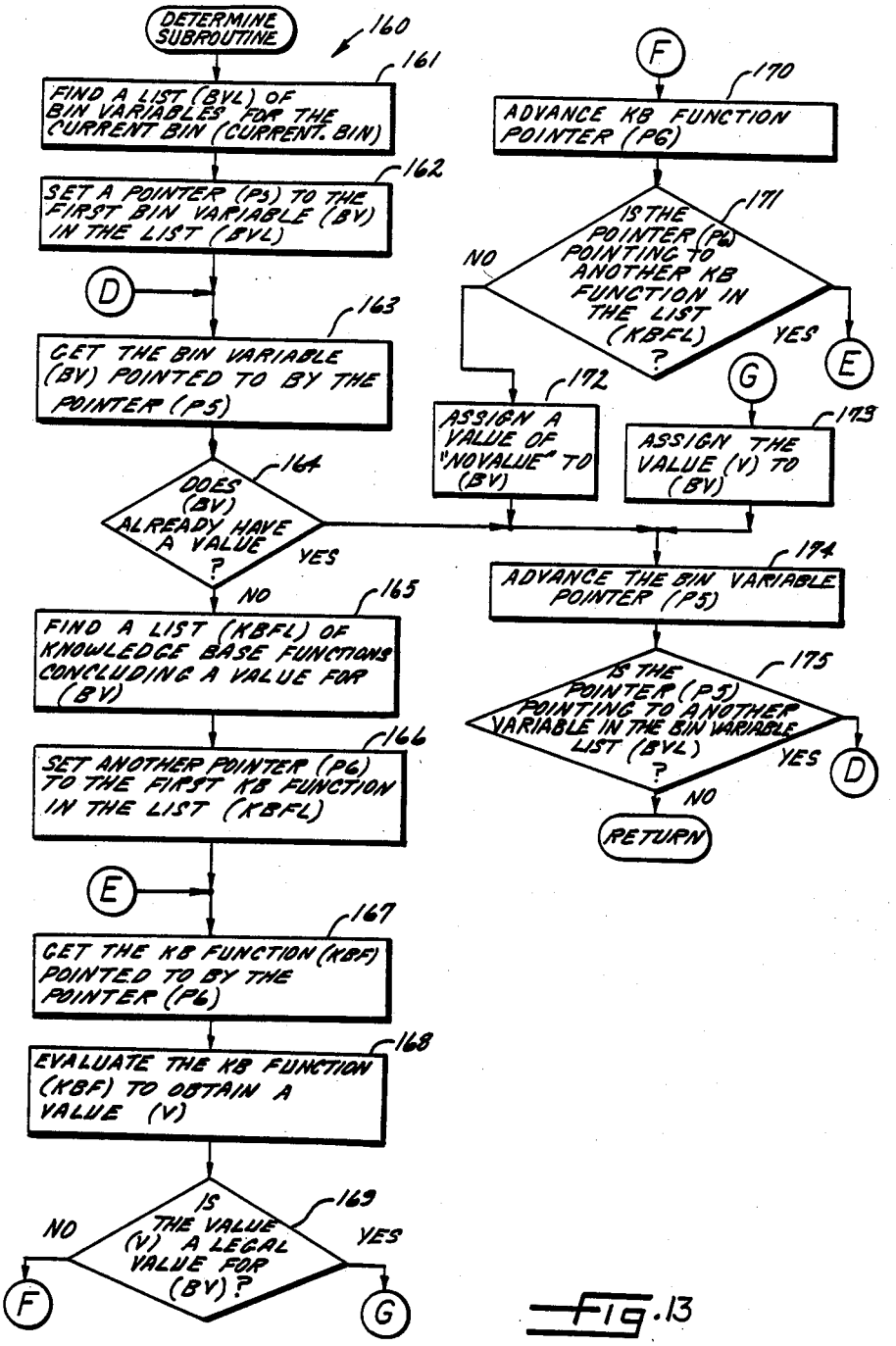

DETERMINE SUBROUTINE
160
FIND A LIST (BVL) OF BIN VARIABLES FOR THE CURRENT BIN (CURRENT. BIN)
161
SET A POINTER (P5) TO THE FIRST BIN VARIABLE (BV) IN THE LIST (BVL)
162
D
GET THE BIN VARIABLE (BV) POINTED TO BY THE POINTER (P5)
163
DOES (BV) ALREADY HAVE A VALUE ?
164
YES
NO
FIND A LIST (KBFL) OF KNOWLEDGE BASE FUNCTIONS CONCLUDING A VALUE FOR (BV)
165
SET ANOTHER POINTER (P6) TO THE FIRST KB FUNCTION IN THE LIST (KBFL)
166
E
GET THE KB FUNCTION (KBF) POINTED TO BY THE POINTER (P6)
167
EVALUATE THE KB FUNCTION (KBF) TO OBTAIN A VALUE (V)
168
IS THE VALUE (V) A LEGAL VALUE FOR (BV)?
169
NO
YES
F
G
F
ADVANCE KB FUNCTION POINTER (P6)
170
IS THE POINTER (P6) POINTING TO ANOTHER KB FUNCTION IN THE LIST (KBFL) ?
171
NO
YES
G
E
ASSIGN A VALUE OF "NOVALUE" TO (BV)
172
ASSIGN THE VALUE (V) TO (BV)
173
ADVANCE THE BIN VARIABLE POINTER (P5)
174
IS THE POINTER (P5) POINTING TO ANOTHER VARIABLE IN THE BIN VARIABLE LIST (BVL) ?
175
YES
NO
D
RETURN

Fig. 13

HIERARCHICAL KNOWLEDGE SYSTEM

FIELD OF THE INVENTION

The present invention relates generally to inventory control and the processing of orders for flexibly assembled systems or items of manufacture, and more specifically to computer systems to aid in the checking of orders for products or systems to be manufactured or assembled.

BACKGROUND OF THE INVENTION

The traditional method of mass production realizes economy of scale due to continuous application of capital and division of labor. In recent years, however, a method of "flexible assembly" has become popular for relatively expensive and complex items such as computers. For these kinds of products, the additional cost of assembling a unique system for a particular customer is offset by the omission of parts that are not desired by the customer. The flexibly assembled product becomes a better match to a specific customer's need than a standard product designed for a particular group of customers.

The typical method of handling an order for a flexibly assembled product is to split the order into secondary orders or "production requests" for groups of standard parts or "kits" that are functionally related or comprise sub-assemblies. Since the production requests are standardized, the flexibly assembled item is sold in terms of features or "packages" which can be made up of the kits. The process of converting the desired packages or features into corresponding production requests is a process of expansion from a list of packages to a list of kits. As this expansion process is carried out, the order is also validated or checked for comformance with marketing and engineering constraints on permissible configurations. In many cases, violations of the constraints are not evident when the order is submitted and are revealed only when the order is partially or totally expanded.

Computer systems have been used previously for expanding and validating orders. The expansion and validation, for example, is part of an inventory control system. Using traditional computer programs, the computer system performs a syntactic check on the list of package numbers comprising an order. The computer system may also use product definition files for separate versions of a generic product as the basis for expanding packages into kits, and may check whether the individual package numbers are legal options within a particular hierarchy under at least some conditions. But global engineering constraints such as power and interconnection are not performed by these computer systems using traditional programming techniques. Although it would be possible to program constraints into the computer program for a specific set of product hierarchies, the computer program would not be easy to update and maintain since the computer program would have to be rewritten for each change in the product hierarchy. Moreover, the constraints are relatively complex and change frequently. It is, therefore, not economical to include global constraint checking in these conventional computer systems.

In order to provide a maintainable computer system for running expansion and validation of an order, knowledge-based systems have been considered wherein the knowledge of what the constraints are and

when the constraints are to be applied is encoded declaratively in a knowledge base. A separate knowledge base interpreter interprets the knowledge base to apply the constraints. Consequently, the knowledge base interpreter need not be modified when the knowledge in the knowledge base is updated or changed. Knowledge systems have been used in general for problems that require diagnosis, recommendation, selection or classification. These problems have traditionally been performed by human experts and are not easily implemented using conventional programming techniques.

Presently there are highly developed commercial tools which may be used to build knowledge systems. The well-known commercial tools (such as KS300 manufactured by Teknowledge Inc., 525 University Avenue, Palo Alto, Calif. 94301) are patterned after a tool called EMYCIN described in *The Emycin Manual* by Van Melle et al., Standford University Report No. STAN-CS-81-885, Standford, Calif. 94305 (October, 1981).

EMYCIN is specifically designed as a domain-independent system for constructing rule-based consultant expert system programs. Domain knowledge is represented in EMYCIN systems primarily as condition-action production rules which are applied according to a goal-directed backward-chaining control procedure. Rules and consultation data are permitted to have associated measures of certainty, and incomplete data entry is allowed. The EMYCIN system includes an explanation facility displaying the line of reasoning followed by the consultation program, and answers questions from the client about the content of its knowledge base. To aid the system designer in producing a knowledge base for a specific domain, EMYCIN provides a terse and stylized language for writing rules; extensive checks to catch common user errors, such as misspellings; and methods for handling all necessary indexing chores.

In addition to production rules, the knowledge base for an EMYCIN system includes a hierarchical structure called a "context tree." The elemental representation of an object or idea is defined as a context-parameter-value triple. The context refers generally to an instance of a particular context type, the parameter refers to an attribute of the context instance and the value refers to the particular value of the parameter for the particular context instance. The context tree is defined by parent and offspring declarations for the context types.

The instantiation of contexts is similar to the invocation of a subroutine for each context, the subroutine in effect being defined by various declarations in the context definition. A consultation is started by instantiating a root context and the branches from this root context define major steps in the consultation during which the offspring contexts of the root node are instantiated. Thus, the context definitions are used to structure the data or evidence required to advise a user about the root context. Besides consultation control, the context tree may be used to organize the distinguished components of some object, or for representing distinguished events or situations that happen to an object.

A rule-based knowledge system has been developed for determining a computer system's configuration from a skeletal specification. A system called XCON has been used by the Digital Equipment Corporation when salesmen take orders and when processing clerks check

and flesh out incoming orders for VAX-11/780 computer systems. Given a customer's order, XCON determines what, if any, modifications have to be made to the order for reasons of system functionality and produces a number of diagrams showing how the various components on the order are to be associated.

A version of XCON called **R1** is described in John McDermott, *R1: A Rule-Based Configurer Of Computer Systems,* Department of Computer Science, Carnegie-Mellon University, (April, 1980). It is said that **R1** has sufficient knowledge of the configuration domain and of the peculiarities of the various configuration constraints that at each step in the configuration process, **R1** simply recognizes what to do. Consequently, it is said that little search is required in order for **R1** to configure a computer system.

Although the XCON rule-based system represents a major improvement over traditional computer techniques in the field of computer configuration, it is rather difficult to change the constraint rules in XCON. In particular, to change a constraint rule, one needs to know what occurs before and after the constraint rule is applied. In general, the constraint rules are interlaced and spread, and they are not readily accessible for maintenance since in general they are not conceptually hierarchical.

SUMMARY OF THE INVENTION

The primary object of the invention is to provide a knowledge system for generalized representation and processing of hierarchical assemblies that is easily maintainable and extensible.

Another object of the invention is to provide an intelligible knowledge base representation for hierarchical assemblies and their functionality.

Still another object of the invention is to provide an intelligible knowledge base representation for configuration strategies and actions.

Still another object of the invention is to provide a knowledge system for configuration checking which clearly separates the definition of configuration constraints from configuration checking strategies and actions.

And yet another object of the invention is to provide an improved control procedure for configuration checking.

And still another object of the invention is to provide an explanation facility especially suited for configuration analysis and adaptation.

Briefly, in accordance with the broadest aspect of the invention, a knowledge system for generalized representation and processing of hierarchical assemblies has a hierarchical knowledge base comprising a decomposition of a set of elements into subsets over a plurality of hierarchical levels, a plurality of respective predefined functions or conditions of the elements within the subsets at a plurality of the hierarchical levels, and a predefined set of operations to perform on a user-defined set of elements responsive to the knowledge base. For ease of maintenance and extensibility, the knowledge base is defined declaratively by assigning parent sets to offspring subsets to define the hierarchy, by indicating the conditions of the subsets which satisfy the predefined functions, and by writing task blocks in an imperative language defining the sequence of operations to perform on the user-defined set of elements. Preferably, the operations include operations for matching, configuring and expanding the user-defined set of elements into the defined subsets of individual elements and for evaluating the predefined functions, and the operations are executed recursively.

In accordance with a preferred embodiment of the invention, a knowledge-based configuration system has separate portions encoding a configuration checking strategy, a description of hierarchical functions of the product to be configured, a catalog of the parts and components which implement those functions, assembly constraints that check whether a given set of components will successfully implement their respective hierarchical functions. The configuration system applies the assembly constraints during configuration checking and if necessary, warns the user or modifies the given set of components to insure compliance with the assembly constraints.

The control knowledge is encoded in an imperative language defining specific configuration control actions to be executed during interpretation of the control knowledge according to a built-in control procedure. Since the configuration control knowledge is explicit and results in modification of the given configuration only in a precisely defined fashion, the configuration system can always explain its conclusions and reasoning, and is intelligible and modifiable.

To provide transparent representation of the control knowledge as well as factual knowledge, the knowledge base is preferably organized into distinct portions including the assembling constraints, task blocks encoding the control strategy knowledge, functional hierarchy classes which become instantiated into bins, bin variables which take on values describing the bins and their contents, the catalog of parts, rules for expanding sub-assemblies into parts, and user-defined functions for computing the values of bin variables. The assembly constraints may be defined in terms of specified bin variables and certain built-in functions responsive to the number of specified parts in specified bins.

A particular task block is provided to be executed at the start of a configuration. The knowledge engineer must provide task blocks which are invoked after instantiation of specified functional hierarchy classes and when the values of the bin variables are to be determined, and may provide explicitly invoked task blocks.

The functional hierarchies are preferably represented by entries defining specific functions that are performed by the configured product, and each entry can be composed of more specific functional entries. Associated with each functional entry is a configuration strategy for checking that the parts in a given configuration will implement the desired function. The system will warn the user when the configuration is incorrect, and can optionally modify the configuration by adding, deleting, and modifying parts in the respective bin for the functional hierarchy. The configuration strategy also describes the order that parts that achieve certain functions should be checked, and describes when to expand sub-assemblies into the same bin of their respective functions.

An explanation facility is provided to allow the user to validate the acceptability of the configuration, to identify unacceptable portions of the configuration, and to identify modifications made to the configuration to satisfy the constraints. A tracing and translation facility is provided to expose the specific steps and the knowledge used in the configuration checking process so that the validity of the configuration strategy is apparent.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects and advantages of the invention will become apparent upon reading the following detailed description and upon reference to the drawings in which:

FIG. 1 is a schematic diagram of a knowledge system incorporating various aspects of the present invention;

FIG. 2 is a block diagram of an exemplary M1234 computer system which is configured by the knowledge system in FIG. 1 using the knowledge base shown in Appendices II (A)-(E) to the present specification;

FIG. 3 is a tree diagram of the functional hierarchy for the M1234 computer system shown in FIG. 2;

FIG. 4 is a tree diagram of a functional hierarchy for a standard mini computer;

FIG. 5 is a simplified flowchart for the configuration process performed by the knowledge system in FIG. 1;

FIG. 6 is a diagram of a bin tree corresponding to the functional hierarchy of FIG. 4 for the mini computer, including the final configuration of parts in the bins for the working example of Appendix III (A);

FIG. 7 is a flowchart for the executive program in the built-in control procedure of the knowledge system shown in FIG. 1;

FIG. 8 is a flowchart of an EXECUTE subroutine for executing task blocks of control procedure steps;

FIG. 9 is a flowchart of a FIND subroutine for obtaining a list of parts included in a specified bin which match or satisfy a specified assembly description;

FIG. 10 is a flowchart of a CONFIGURE subroutine for creating a new bin of a specified functional class, transferring a specified list of parts from a specified current bin to the new bin, and temporarily passing execution to the task block associated with the functional class of the new bin;

FIG. 11 is a flowchart of a CHECK subroutine for applying a specified subset of the constraints applicable to the functional class of the current bin;

FIG. 12 is a flowchart of an EXPAND subroutine for expanding the parts in the current bin into their component parts by applying predefined expansion rules associated with the current bin; and

FIG. 13 is a flowchart of a DETERMINE subroutine for determining values for bin variables associated with the current bin by applying knowledge base functions associated with the current bin.

While the invention is susceptable to various modifications and alternative forms, a specific embodiment thereof has been shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that it is not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the invention as defined by the appended claims.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Turning now to FIG. 1, there is shown a block diagram generally designated **10** of a knowledge system for processing an initial configuration of elements or an order **11** to arrive at a final configuration or production request **12**. The system **10** is recognized as a knowledge system since it has a domain-independent knowledge base interpreter **13** executing a built-in control procedure for interpreting a domain-dependent knowledge base **14**. The knowledge base **14** encodes a generic configuration in a highly structured and transparent format. In general terms, the knowledge base interpreter **13** matches the elements of the initial configuration in the order **11** to the generic configuration in the knowledge base **14** to structure or configure the elements according to the generic configuration. The elements structured in terms of the generic configuration are stored in a working configuration memory **15**.

Once the elements or parts of the order **11** are structured according to the generic configuration, the knowledge base intepreter **13** may apply functions or constraints **16** stored in the knowledge base **14** to the working configuration in the memory **15**. The constraints **16**, for example, indicate particular aspects of the working configuration **15** that are not apparent from the initial configuration or order **11**. The constraints **16** may also indicate desirable changes to make to the working configuration **15**. The knowledge base interpreter **13** may execute these indicated changes in order to change the working configuration. At the end of processing by the knowledge base interpreter **13**, these executed changes are reflected in the final configuration **12**.

The knowledge system **10** has a trace memory **17** for keeping a precise record of the steps performed during the matching of the elements of the initial configuration **11** to the generic configuration in the knowledge base **14** and during the application of the constraints **16** to the working configuration **15**. To enable a user or order analyst **18** to inspect relevant portions of the trace **17**, a user interface and explanation facility **19** is provided. The user **18** may, for example, request a complete typescript of the contents of the trace memory **17** in an easily understandable form. An example typescript is included in Appendix I at the end of the present specification.

The knowledge base **14** is arranged so that it is easily modified and maintained to reflect desired changes in the generic configuration. The modification and maintenance is performed by a knowledge engineer **20** and the knowledge base **14** is accessed through an editor **21** called a knowledge base maintenance facility. The knowledge engineer **20** initially creates the knowledge base **14** collecting and encoding knowledge about the generic configuration according to a precise format and syntax so that the knowledge is transparent to the knowledge engineer and is machine readable by the knowledge base intepreter **13**. For the processing of an order of parts **11** to generate a production request **12** for a system or product, the knowledge engineer **20** collects engineering and marketing data on the generic configuration of the product and its constraints and encodes the information according to the syntax recognized by the knowledge base interpreter **13**.

The specific format of the knowledge base **14** is chosen so that the knowledge base can be easily updated to reflect the addition of new products and changes in current products. The generic configuration is represented as a functional hierarchy **22** which describes the product in various degrees of particularity. When a product is modified, only the particular portions of the functional hierarchy **22** need be changed. Moreover, since the hierarchy is in terms of the function of performance of parts of the generic configuration for the product, the changes required in the functional hierarchy **22** are minimal.

For each function in the functional hierarchy **22**, a separate task block **23** is provided to encode corresponding steps in the control procedure followed by the knowledge base interpreter **13** to match the parts or elements of the initial configuration **11** to the functional hierarchy **22**. A change in the functional hierarchy usually requires only a localized change in the respective task block **23**. Also, the constraints **16** correspond to particular functions in the functional hierarchy **22** so that the constraints **16** are also easily modified. Consequently, the constraints associated with a particular function in the functional hierarchy are applied in a manner specified by the respective task block **23**.

It should be noted that even though the functional hierarchy **22** might not change during the life of a particular product, the parts or elements making up the product may change during the life of the product. These changes are reflected in a parts catalog **24**. Moreover, to simplify the process of taking an order from a customer, an order typically includes packages of parts implementing certain features. To enable the knowledge base interpreter **13** to find the particular parts included in the packages in the order **11**, the knowledge base **14** includes a set of expansion rules **25**.

In addition to providing a framework for clearly intelligible and maintainable representation of knowledge about a product, the functional hierarchy **22** provides the structure or framework for the control procedure executed by the knowledge base interpreter **13** to match the parts in the order **11** to the generic configuration in the knowledge base **14** and to apply the constraints **16** to validate the order **11** and generate a production request **12**. The working configuration memory **15** is organized into respective bins of parts **26** corresponding to the major functional components of the product defined by the functional hierarchy **22**. Each bin of parts **26** corresponds to a structural or functional assembly in the product and the matching process is performed by inputting the initial configuration or list of parts **11** into an initial bin of parts **26** and sequentially matching the parts in parent bins to assembly descriptions for offspring bins and transferring the matching parts to the offspring bins. The task blocks **23** specify the particular assembly descriptions and the specific sequence of matching and transferring the parts. Before matching and transfering the parts from certain bins, however, the expansion rules **25** must sometimes be applied to "break open" certain packages of parts in the order **11** since different parts from the same packages sometimes match the assembly descriptions of different offspring bins. At the end of the transfer process, the bins of parts **26** contain respective parts from the initial configuration **11**. Hence, the parts in the initial configuration **11** have been structured or configured according to the generic product configuration or functional hierarchy **22** and thus the constraints **16** may be applied to their respective bins of parts **26**.

In order to simplify the process of applying the constraints, the constraints **16** may include bin variables defined for particular respective parent bins and which may be referenced in the parent bins and respective offspring bins of the parent bins. The knowledge base **14** includes separate declarations **28** of the bin variables. The respective task blocks **23** include steps for determining the values for the bin variables **27** and the values are stored in the working configuration memory **15**. These steps in the task blocks **23** may include function calls to knowledge base functions **29** in the knowledge base **14**. The knowledge base functions **29** typically count parts and perform numerical computations to determine values for bin variables. In addition to conditioning constraints, bin variables may be used to condition the sequence of execution of the steps in the task blocks in order to perform conditional expansion, matching and transfer of parts and assemblies.

Turning now to FIGS. **2** and **3**, the process of generating a functional hierarchy for a particular product is illustrated. The knowledge engineer obtains a description of the product from product engineers and technical manuals. The description should include an engineering diagram of the product in the form of a functional block diagram. A functional block diagram generally designated **30** for an exemplary M1234 computer system is shown in FIG. **2**. The block diagram **30** shows functional subsystems arranged around an internal transfer subsystem **31** which includes, for example, the mainframe and primary input/output (I/O) bus in the M1234 computer. The internal transfer subsystem **31** exchanges data among the various other subsystems of the computer. These other subsystems include a multiprocessing communications subsystem **32**, a processor subsystem **33** including the central processing unit of the computer, an instruction sequence unit **34** for controlling the sequence of instructions executed by the central processor, integrated memory subsystems **35** including the random access memory for the computer, an integrated communication subsystem **36** for providing a number of data links between the computer and external devices, an integrated disk subsystem **37** having a number of disk drives, I/O trunk subsystems **38** interfacing a number of computer peripherals, and a service subsystem **39** for interfacing printers, flexible disk drives, card readers, and computer terminals.

From the block diagram in FIG. **2** and other technical information about the M1234 computer, the knowledge engineer **20** generates a functional hierarchy description of the M1234 computer as illustrated in FIG. **3**. In FIG. **3** the hierarchy is arranged in the form of a tree diagram generally designated **40**, having an initial or root node **41**, intermediate or branch nodes **42–46**, and several terminal or leaf nodes **47–60**. Each node in the functional hierarchy **40** represents a single functional component. The initial node **41** and the intermediate nodes **42–46** represent functional components that are composed of a number of more specific functional components. Specifically, the functional hierarchy **40** is defined by including declarations of parent functional components in the definitions of the intermediate **42–46** and terminal **47–60** functional components. The functional component MAINFRAME **42**, for example, has the parent functional component SYSTEM **41**. The functional hierarchy **22** of the knowledge base **14** (FIG. **1**) corresponding to the tree diagram **40** in FIG. **3** is listed in Appendix II (A). According to the particular knowledge base syntax of Appendix II (A), it is said that the declared offspring functional component "COMPOSES" its parent functional component.

It should be noted that the consultation typescript of Appendix I was generated by the interpretation of a knowledge base in Appendices II (A)–(E). The task blocks are in Appendix II (B); the parts catalog is in Appendix II (C); the constraints, bin variable declarations, and knowledge base functions are in Appendix II (D); and the expansion rules are in Appendix II (E).

In general the knowledge base is in the form of a list of declarations of task blocks which are invoked and

executed when the system is run, declarations of knowledge base functions which are applied or evaluated during execution of the task blocks to set values of the bin variables, and declarations of knowledge base objects which become instantiated to generate one or more object instances in working memory during execution of the task blocks. Declarations of task blocks and knowledge base functions are preceded by the identifiers DEFTASKBLOCK and DEFKBFUN, respectively, followed by a specified name for the task block or knowledge base function. Declarations of objects are preceded by the identifier DEFCLASS, followed by a specified name for the object and, in parentheses, the class or type of the instantiated object. Object types include BIN (an instance of component or node in the functional hierarchy), PRODUCT.ID (an instance of a particular part), BVI (a bin variable instance), CTI (an instance of applying a constraint or product expansion), and QUEUE. Respective QUEUE instances keep track of the current step of execution in each task block so that the execution of one task block may be interrupted to execute another task block.

The declarations or definitions of object types in the knowledge base include values for instance variables or IVARs associated with instances of the object types. For each offspring BIN, for example, the IVAR "COMPOSES" is set to the name of the BIN's parent BIN. Each BIN and PRODUCT.ID has an IVAR "DESCR" including a description of the respective function or part in English so that the user interface and translation facility **19** may generate comprehensible explanations and traces of the operation of the system **10** (FIG. **1**).

Each part in the parts catalog **24** (FIG. **1**) implements one of the functions described in the functional hierarchy and each part is annotated to indicate the appropriate functional hierarchy element. In the parts catalog of Appendix IIC, each PRODUCT.ID or individual part has an IVAR "IMPLEMENTS.FN" including the name of the functional component which the part implements. These declarations of the functions implemented by the parts are critical to the operation of the system, since they are used to group and manipulate parts of different functional types during the matching operations.

As noted above, each component of the functional hierarchy has associated with it a particular task block **23** specifying when and how the system should focus on the ordered parts that implement the respective functional component. Also, associated with each functional component in the hierarchy is a set of constraints and bin variables **16** which are evaluated or applied during execution of the task block associated with the functional component.

It is apparent that the functional hierarchy serves both to organize the configuration process and to guide the knowledge engineer in designing and maintaining the knowledge base. By understanding the structure of the functional hierarchy and the effects of the configuration strategy used to process an order, the knowledge engineer can understand how the system will configure an order and how the application of constraints, product definitions, and expansion will affect the processing of that order.

The knowledge system **10** of FIG. **1** has its greatest utility for configuring complex systems such as the M1234 computer as described in FIGS. **2** and **3** and the typescript and knowledge base of Appendices I and II (A)–(E). For the sake of illustration, however, the internal operation of the system will be described in connection with a more easily understood minicomputer system corresponding to the functional hierarchy generally designated **70** in FIG. **4** and corresponding to the consultation typescripts and knowledge base in Appendices III (A)–(B) and IV (A)–(D).

In the functional hierarchy **70** of FIG. **4**, each node such as the initial node **71** is labeled by its functional component and has a class name enclosed in parentheses corresponding to the classes declared in the knowledge base functional hierarchy of Appendix III. The minicomputer **71** is presumed to have two major functional components. The main-frame **72** of the minicomputer encompasses all of the active components of the computer such as the central processing unit **73**, a computer memory **74**, and input/output devices **75**. These active devices **73–75** are physically tied together by a card cage **76**. The card cage **76** includes mechanical support for various circuit boards performing the active functions of the computer and also provides a network of electrical connections for the transfer of data among the circuit boards. The second functional component of the minicomputer **71** is a power supply **77** which feeds power to the circuit boards through power supply lines running through the card cage **76**.

The minicomputer **71** also has a number of sub-assemblies performing active functions. Random access memory (RAM) **78** records data generated during operation of the minicomputer. Read-only memory (ROM) **79** contains fixed data used, for example, to initially run the minicomputer when the computer is first turned on, and also to run fixed utility programs for input and output and also to interpret, compile, or load a computer program written in a standard programming language.

The input/output function **75** has three separate functional assemblies. The minicomputer may include a number of terminals **80** for interacting with a human user. A terminal includes, for example, a cathode ray tube for displaying data to the operator, and a keyboard for receiving data from the operator. The terminals **80** and any other input/output devices are connected or interfaced to the computer via interface circuits **81** on circuit boards received by the card cage **76**. The input/output function **75** may also include disk drives **82** for recording and storing relatively large amounts of data.

As a physical object, the minicomputer **71** is comprised of discrete physical components or parts. For a particular model of minicomputer, the minicomputer should be assembled from only a specified set of parts. These specified parts are listed in the parts catalog of Appendix IV (C). These parts include a CAGE-1000 for implementing the card cage **76**, a CPU-1000 central processing unit circuit board for implementing the CPU function, a DISC-1000 disk drive, an IF-1000 interface circuit board, a RAM-1000 random access memory circuit board, a ROM-1000 read-only memory circuit board, a TERMINAL-1000 computer terminal, and three different power supplies, including a five ampere power supply SUPPLY-05, a ten ampere power supply SUPPLY-10, and a fifteen ampere power supply SUPPLY-15. The parts catalog in Appendix IV (C) also assigns a model number MINI-1000 to the minicomputer **71**.

An order for a MINI-1000 minicomputer must at least include a set of parts, but the set may be empty. In general, an order for a product consists of three types of information. The first type is called "order information"

consists of such information as the customer name and an order number. It is primarily used to identify the order and does not effect the configuration process. The second type called "order lines" is a list of parts ordered and the quantity of each part desired. In general the order lines include the product model number and additional or optional features requested by the customer. The list of parts ordered and the quantity of each part desired is checked by the knowledge system **10** to ensure compatibily. The third type called "field service parts" is a list of additional part numbers and are merely appended or said to be shipped "on top" of the order. The field service parts are typically replacement or spare parts for the customer. The field service parts are not intended to be built into the system and are not checked either for compatibility among themselves or with the rest of the order.

For the consultation typescript in Appendix III (A), the order consists merely of order lines. These order lines specify that the product to be configured is one MINI-1000 minicomputer comprising four RAM-1000 random access memory circuit boards, three ROM-1000 read only memory circuit boards, four TERMINAL-1000 computer terminals, and three DISC-1000 disk drives.

Once the order is received by the knowledge system **10**, the parts in the order are checked for consistency by applying the constraints in the knowledge base (Appendix IV (D)). In general, constraints specify engineering or marketing conditions which must be satisfied for the order lines. The constraints also specify the particular action that should be taken in response to whether the conditions are satisfied when the constraints are applied. For the MINI-1000 minicomputer, the order must include at least one RAM-1000 random access memory board, at least one ROM-1000 read only memory board, at least one TERMINAL-1000 computer terminal, and at least one IF-1000 interface board. The action to take when any of these parts is found to be missing is to add one of the respective missing parts, since these parts are essential to the operation of the minicomputer. The minicomputer also has a number of essential components that must be added if missing, but which can only appear with a quantity of one. The minicomputer must have one and only one power supply (SUPPLY-05, SUPPLY-10, or SUPPLY-15), one and only one CPU-1000 central processing unit card, and one and only one CAGE-1000 card cage.

The MINI-1000 minicomputer also has certain functional constraints conditioned on combinations of components. Due to the limited range of addresses addressed by the central processing unit **73**, the memory **74** has a limited size. This memory constraint in turn limits the total number of memory circuit boards to no more than seven. If the number of RAM-1000 cards plus the number of ROM-1000 cards exceeds seven, the user must be told that he has ordered too many memory cards. This kind of constraint is called a warning constraint in contrast to the previous modification constraints which require the working configuration of parts to be changed.

The input/output assembly **75** has a few rather complex constraints. No more than four terminals are permitted for the MINI-1000 computer due to marketing objectives. The customer must, for example, buy a more expensive MINI-2000 computer in order to have a system with more than four terminals. Also, a single interface card can either support four terminals, or one terminal and one disk drive per card since the DISC-1000 disk drive requires three times as many data lines than the TERMINAL-1000 computer terminal. The order is modified by adding interface cards and deleting terminals so that these input/output conditions are satisfied.

The mainframe **72** has a hardware or interconnection constraint since the CAGE-1000 card cage has room for no more than sixteen cards or circuit boards. The user is warned if the total number of CPU-1000, RAM-1000, ROM-1000, and IF-1000 cards exceeds sixteen.

The minicomputer **71** has a global power constraint due to the fact that the customer may order a power supply that is smaller than that required under the worst case conditions. The power required is a function not only of the number of cards, but of the particular cards that are used. The CPU-1000 card requires two amperes of current, the RAM-1000 card requires one ampere of current, the ROM-1000 card requires 0.5 amperes of current, and the IF-1000 interface card requires 0.2 amperes of current. The knowledge base in Appendix IV (D) includes a special knowledge base function called "COMPUTE.POWER.NEEDED" for calculating the total amount of power required. Based on the required power, a five ampere power supply SUPPLY-05, a ten ampere power supply SUPPLY-10, or a fifteen ampere SUPPLY-15 should be included in the order. The knowledge base in Appendix IV (D) includes a set of modification constraints to ensure that the smallest power supply is included consistent with the total power requirement.

Shown in FIG. **5** is a flowchart generally designated **85** describing the basic steps in the order checking process. In the first step **86**, the order or initial configuration **11** is received by the knowledge system **10** (FIG. **1**). Then in step **87** the individual components are determined from the order so that the constraints may be applied in step **88**. Once the constraints are applied, in step **89** the indication of the application of the constraints is transmitted to the user. Application of warning constraints may result in warnings to the user, and application of the modification constraints may lead to a production request **12** that differs from the original order **11**.

For the MINI-1000 minicomputer, the computer's functional hierarchy, parts catalog, and constraints are relatively simple so that the order checking process in FIG. **5** could easily be implemented using a conventional programming language such as the BASIC language. Since an order for a MINI-1000 computer is merely a list of the indivisible components of the computer, the constraints are easily applied. More importantly, the functional class which each part implements is readily apparent from the type of part. For the M1234 computer described in the knowledge base of Appendices II (A)–(E), however, it would be very difficult to use a conventional programming language to implement an order checking computer system and the computer system would be very difficult to maintain.

By using the techniques already described, the components and functionality of a complex product such as a computer can be transparently and explicity represented in the knowledge base. In accordance with another aspect of the present invention, the definition of configuration constraints is clearly separated from the configuration strategies and actions. In particular, a built-in control procedure is provided which permits the constraints to be defined for particular functional components and separately determines which parts are

applicable to the respective constraints. At the most basic level, this separation requires a process which builds an explicit representation of the assemblies of the product from the parts in the order. Shown in FIG. 6, for example, is an explicit representation generally designated 90 for the MINI-1000 minicomputer and the example order given above. The explicit representation 90 is a tree of bins corresponding to the functional hierarchy of FIG. 4. In FIG. 6 the name of each bin corresponds to the respective functional component or bin class in the hierarchy of FIG. 4. Moreover, the particular instance of the class is identified by a numeral enclosed in parentheses. Although the examples in the appendices only use a single instance of each functional class, the knowledge engineer may write task blocks which create a specified number of bins for a specified functional class.

The bins are used to hold a set of product instances or parts. The product instances are obtained from the order lines, by application of modification constraints or by EXPANSION operations upon parts initially placed in the bins. The MINI-1000 computer, however, does not have any expandable parts in its parts catalog. Typically for marketing reasons, an order may include packages of parts that are distinguished from orders of the individual parts included in the package. For the MINI-1000 computer, for example, it could be economically desirable to define a basic package including the minimum components. Thus, the basic package would include one CAGE-1000, one SUPPLY-05, one CPU-1000, one RAM-1000, one ROM-1000, one TERMINAL-1000, and one IF-1000. The basic package would be sold at a price less than the total price of the components of the package if the components were ordered separately. Of course, the intention of the manufacturer is not to sell the package at a lower cost, but to sell the packate at market cost and to increase the price of any additional components purchased on top of the basic package.

To apply the constraints, the packages must be expanded into their component parts since the constraints typically apply to individual parts and not packages. The constraints are preferably independent of the product expansions so that the packages may be redefined merely by changing the expansion rules (25 FIG. 1).

Due to the complexity introduced by packages and modification constraints, it is important that the components of the order are properly associated with corresponding constraints. Preferably, this correspondence is obtained by matching the components in the order to predefined assembly descriptions and then applying the constraints to the components which match the respective assembly descriptions. Also, the preferred method of performing the matching is by transferring or sorting the entire order through the bin tree, starting at the initial bin until the elemental parts end up in the terminal bins. For the MINI-1000 order processed in Appendix III (A), the result of the sorting is illustrated in FIG. 6.

The order in which parts are matched to assembly descriptions, sorted and checked by the constraints is all controlled by the steps in the task blocks. The execution of the steps in the task blocks is started by an executive program 100 shown in FIG. 7. In the first step 101, certain program variables are cleared. By clearing these program variables, space in the working memory is cleared so that a new bin tree may be built. Also, the trace memory is cleared in order to record a new trace.

In step 102 an initial bin is created and is given the name "UNASSIGNED". The name "UNASSIGNED" is given denoting that the particular product being configured is not yet known since the order has not yet been received by the knowledge system.

It should be noted that the knowledge base interpreter 13 is always working on a particular bin at any given time. The name of this particular bin is stored in a global variable called "CURRENT.BIN". In step 103 this global variable "CURRENT.BIN" is set to the value "UNASSIGNED" to record that the configuration process is starting in the "UNASSIGNED" bin. In step 104 "UNASSIGNED" task block is found, and it is excecuted in step 105 so that the rest of the configuration process is controlled solely by task blocks.

The task blocks are executed by calling an EXECUTE subroutine generally designated 110 in FIG. 8. In general terms, the EXECUTE subroutine sequentially executes all of the commands in a specified task block. In the first step 111, a pointer (P1) is set to the position of the first command in the task block. In step 112 the command pointed to by the pointer (P1) is executed. In step 113 the pointer (P1) is advanced to that it either points to another command or is at the end of the task block. In step 114, the end of the task block is detected by testing whether the pointer is out of the range of the specified task block. If it is not, execution jumps to step 12 to execute the next command. Otherwise, the EXECUTE subroutine is finished and execution returns to the calling program.

Preferably, the task blocks are written in a well-known imperative programming language such as LISP described in the text P. Winston & V. Horn, *LISP*, Addison-Wesley Publishing Co., Reading, Mass. (1981). The "UNASSIGNED" task block is as follows:

```
(DEFTASKBLOCK UNASSIGNED.TB UNASSIGNED ( )
(INPUT.ORDER)
(DETERMINE '(TODAYS.DATE ORDERED.PARTS
ORDERED.PRODUCT.LINE))
(IF CHECKING.ORDER THEN
(CONFIGURE (FETCH 'ORDERED.PRODUCT.LINE)
(FIND 'ALL 'CCS))
(CHECK 'EMPTY.BIN))
(DETERMINE '(FINAL.PARTS))
(OUTPUT.ORDER)
)
```

Translated into English, the UNASSIGNED task block above first inputs the order of parts. Then it determines the current date, the ordered parts or order lines of the order, and the product line corresponding to the ordered parts. It is presumed that the knowledge system is currently checking the order. Thus, the task block for the ordered product line is fetched from the knowledge base. This task block is, for example, the MINICOMPUTER task block in Appendix IVB. Then a FIND operation is performed to obtain a list of all the parts in the order lines, and the CONFIGURE operation is performed to create the initial bin in the bin tree, to transfer all the parts in the order lines into the initial bin, and to temporarily transfer execution to the MINICOMPUTER task block. When execution returns, the parts are configured in the bins as shown in the bin tree 90 of FIG. 6. A final set of EMPTY.BIN constraints are applied to ensure that all of the order lines in the order have been properly configured. If so, the FINAL.PARTS are determined and a production request is generated by the OUTPUT.ORDER statement.

The UNASSIGNED task block as well as the task blocks in the knowledge base include a mixture of LISP code written by the knowledge engineer, built-in functions supplied by the LISP compiler, and special functions or subroutines comprising the built-in control procedure of the knowledge base interpreter **13** (FIG. **1**). The primary built-in subroutines include the EXECUTE subroutine already described in conjunction with FIG. **8**, a FIND subroutine shown in FIG. **9**, a CONFIGURE subroutine shown in FIG. **10**, a CHECK subroutine shown in FIG. **11**, an EXPAND subroutine shown in FIG. **12**, and a DETERMINE subroutine shown in FIG. **13**.

The basic matching operation is performed by the FIND subroutine of the FIG. **9**. The FIND subroutine generally designated **120** obtains a list of parts included in a specified bin which match or satisfy a specified assembly description. An assembly description is a predicate or Boolean function defining the desired set of parts in terms of a conjunction of part names, part types, or other functions which describe the desired parts. A collection of parts satisfies an assembly description when the parts fit that description. In other words, the assembly description serves as a template to select the desired parts.

In the first step **120'** of the FIND subroutine, an assembly list (AL) is cleared in order that it may receive the names of the desired parts once the desired parts are found. In step **121** a pointer (P2) is set to the first part in the specified bin. In particular, the FIND subroutine **120** in FIG. **9** is used to find parts in the current bin to transfer them to offspring bins of the current bin. Hence, in step **121** the specified bin is the value of the global variable "CURRENT.BIN". In step **122**, the specified assembly description is evaluated for the part in the current bin pointed to by the pointer (P2). The assembly description, in other words, is a Boolean function of the part pointed to by the pointer (P2). In step **123**, the Boolean value of the assembly description indicates whether the part satisfies the assembly description. If the part satisfies the assembly description, then the part is one of the desired parts so that in step **124** the part is added to the assembly list (AL). If the part does not satisfy the assembly description, it is not added to the assembly list (AL). Then in step **125** the pointer (P2) is advanced so that it may point to another part in the current bin. The pointer (P2) might not, however, point to another part in the current bin since all of the parts in the current bin might have been tested. In step **126** the pointer (P2) is tested to determine whether it is out of the range of the current bin. If it is not out of range, then the pointer (P2) does point to another part in the current bin and execution jumps back to step **122**. Otherwise, all of the parts in the current bin have been checked and the assembly list (AL) will contain the names of all of the parts that satisfy the specified assembly description. Hence, execution returns to the calling task block.

In order to perform the sorting operation described above, a CONFIGURE operation is performed to create a new bin of a specified functional class, to transfer the matched parts from the current bin to the new bin, and to temporarily pass execution to the task block associated with the functional class of the new bin. A flowchart for the CONFIGURE subroutine generally designated **130** is shown in FIG. **10**. In step **131** a new bin of the specified functional class is created by allocating space in the working configuration memory **15** (FIG. **1**) for the new bin. Space is allocated, for example, for instances of all the bin variables declared for the bin and the bin variable instances are marked undetermined to indicate that they do not yet have values. Next, in step **132**, it is asserted that the parent of the new bin is the current bin specified by the global variable "CURRENT.BIN". In step **133**, the specified parts are removed from the current bin and transferred to the new bin. In step **134** the value of the global variable CURRENT.BIN is pushed onto a bin stack and the global variable is assigned the name of the new bin. This is done in anticipation of interrupting the processing for the current bin in order to process the new bin. Specifically, in step **135**, the task block associated with the given functional class of the new bin is found. In step **138** the steps in this task block are executed. Upon completion of executing the steps in this task block, in step **137** the bin stack is popped and the popped value is reassigned to the global variable CURRENT.BIN to continue execution of the task block for the current bin. Execution then returns to the task block which called the CONFIGURE subroutine **130**.

The fact that a stack was used in the CONFIGURE subroutine **130** suggests that the CONFIGURE subroutine is advantageously used in a recursive fashion. In general, each task block associated with an intermediate node in the functional class hierarchy will have CONFIGURE operations in its associated task block for creating offspring bins preceded by a FIND operation to specify the parts to transfer to the offspring bins. This ensures that the parts initially placed into the root bin or added by product expansions become sorted through the bin tree to the terminal or leaf bins. Once the parts have filtered down to the terminal leaf bins, the most basic and elementary constraints may be applied to determine, for example, if specified parts or a specified number of parts are included in the terminal bins. To apply these constraints, a CHECK operation is provided to apply a specified subset of the constraints applicable to the functional class of the current bin.

Shown in FIG. **11** is a flowchart of the check subroutine **140**. In the first step **141** the knowledge base is searched to obtain a list (CL) of the constraints applicable to the functional class of the current bin and which satisfy the given constraint description. Then, in step **142** a pointer (P3) is set to the first constraint in the constraint list (CL). Next, in step **143** the condition portion of the constraint in the list pointed to by the pointer (P3) is evaluated. The condition of the constraint is typically a Boolean function of the conditions of the various parts in the bin. If the condition of the constraint is satisfied, as determined by testing whether the Boolean value of the conditions is true, then in step **145**, the action specified by the constraint is executed. Otherwise, the action specified by the constraint is not executed. In step **146** the pointer (P3) is advanced in order to point to the next constraint in the constraint list (CL). The pointer (P3) must be checked, however, in step **147** to determine whether all of the constraints in the list have been applied. If the pointer (P3) is still within the range of the constraint list (CL), then execution jumps to step **143** to apply the next constraint. Otherwise, the CHECK subroutine **140** is finished and execution returns to the current task block.

It should be noted that during certain steps in the sorting or configuring process, certain kits or groups of parts must be expanded into their component parts. An EXPAND operation is provided for expanding the parts in the current bin into their component parts by

applying predefined expansion rules associated with the current bin. An EXPAND subroutine generally designated **150** is shown in FIG. **12**. In the first step **151**, a list (EXL) of all of the expansion rules that apply to the parts in the current bin are obtained from the knowledge base. Next, in step **152** a pointer (P4) is set to the first expansion rule (EX) in the expansion rule list (EXL). Then in step **153** the particular expansion rule (EX) pointed to by the pointer (P4) is obtained. The expansion rule (EX) is checked in step **154** to determine whether it has already been marked expanded. It should be noted that a particular expansion rule may have already been applied since a particular expansion rule may apply to more than one bin. If the expansion rule has not already been applied, then in step **154'** the expansion is applied by adding to the current bin the parts included in the composing assembly description of the expansion rule. The composing assembly description is a list of the parts included in the respective kit for the respective expansion rule. Once the expansion rule has been applied, it is marked expanded in step **155**. Then, in step **156** the pointer (P4) is advanced in order to point to the next expansion rule in the list. However, in step **157** the value of the pointer (P4) must be checked to determine whether it does point to another expansion rule in the list, or whether all of the expansion rules in the list have been applied. If the point (P4) points to another expansion rule in the list, then execution jumps to step **153**. Otherwise, execution returns to the current task block.

During the execution of task blocks and the application of constraints, it is in many cases desirable to condition the execution of a particular task block step upon the conditions or attributes of the parts included in the bin for the task block. As noted above, bin variables are provided for this purpose and the bin variables may also be included in the condition portions of constraints. A DETERMINE operation is provided for specifying when to determine values for the bin variable instances associated with the current bin by applying the knowledge base functions associated with the current bin. Shown in FIG. **13** is a flowchart of a DETERMINE subroutine generally designated **160**. In the first step **161** a list (BVL) of bin variables for the current bin are obtained from the knowledge base. Next, in step **162** a pointer (P5) is set to the first bin variable in the bin variable list (BVL). Then in step **163**, the particular bin variable (BV) pointed to by the pointer (P5) is obtained. Before determining a value for the bin variable instance (BV), in step **164** the bin variable instance is inspected to determine whether it already has a value. If not, then in step **165** a list (KBFL) of knowledge base functions concluding a value for the bin variable (BV) is obtained from the knowledge base. Then in step **166** another pointer (P6) is set to the first knowledge base function in the function list (KBFL). In step **167** the particular knowledge base function (KBF) pointed to by the pointer (P6) is obtained. This knowledge base function (KBF) is evaluated in step **168** to obtain a respective value (V). This respective value (V) might not, however, be a legal value for the particular bin variable (BV). In step **169** the declared legal values for the particular bin variable (BV) are obtained from the knowledge base and compared to the particular value (V) to determine whether the value is a legal value for the bin variable (BV). If not, then in step **70** the knowledge base function pointer (P6) is advanced. In step **171** the pointer is compared to the end of the range for the function list (KBFL) to determine whether the pointer (P6) is pointing to another knowledge base function. If so, then execution may jump back to step **167** in an attempt to apply this other knowledge base function. Otherwise, there are no more knowledge base functions to try so that in step **172**, a value of "NO VALUE" is assigned to the bin variable instance (BV).

If in step **169** the value (V) was a legal value for the particular bin variable (BV), then in step **173** the particular value (V) is assigned to the bin variable instance (BV).

Now that a value for the particular bin variable instance (BV) has been determined, the bin variable pointer may be advanced in step **174** in an attempt to point to the next bin variable in the bin variable list (BVL). In step **175** the pointer (P5) is first compared to the end of the bin variables list to determine whether it is pointing to another bin variable. If so, execution jumps back to step **163** to determine the value for the next bin variable instance. Otherwise, all of the bin variables in the bin variable list (BVL) have values and execution may return to the current task block.

The primary operations of EXECUTE, FIND, CONFIGURE, CHECK, EXPAND, and DETERMINE are used in the task blocks to define a specific configuration strategy suited for the particular product. In general, however, it can be said that the task blocks for the interior bins have similar characteristics and also the task blocks for the leaf or terminal bins have similar characteristics. A generalized task block for an interior bin is shown below:

```
(DEFTASKBLOCK INTERIOR.TB INTERIOR ( )
(CONFIGURE 'OFFSPRING1 (FIND 'ALL 'CCS
(IMPLEMENTS OFFSPRING1)))
(CONFIGURE 'OFFSPRING2 (FIND 'ALL 'CCS
('IMPLEMENTS OFFSPRING2)))
.
.
.
(CHECK 'REQUIREMENTS)
(EXPAND* '(KITS))
)
```

Translated into English, execution of the interior bin task block first finds all of the components in the interior bin which implements the first offspring function. Then a configure operation creates a bin corresponding to the first offspring function and the parts that are found are transferred to this offspring bin. Execution is then temporarily transferred to the offspring bin. Upon returning from the task block of the first offspring, the components implementing a second offspring function are found and another configuration operation is performed to create a second offspring bin. The parts that are found are transferred to the second offspring bin and execution is passed to the task block for the second offspring function. Upon returning, the parts for any other offspring functions are found and offspring bins are configured. Once all of the offspring bins have been configured, the requirements or constraints for the interior bin are applied. Finally, any kits that may have been added by the application of modification constraints are expanded. The EXPAND* operation is similar to the EXPAND operation except that it specifies a type of part to be expanded (such as any kit) so that the particular kits to be expanded need not be specified.

A generalized task block for leaf bins is shown below:

```
(DEFTASKBLOCK LEAF.TB LEAF( )
(CHECK 'CONTENTS)
(EXPAND* '(KITS))
(CHECK 'REQUIREMENTS)
)
```

First, modification constraints are applied in order to ensure that the leaf bin will not be empty. Then, specified kits in the leaf bin are expanded into their component elemental parts. Finally, warning and modification constraints are applied to check that the requirements for the leaf task block have been satisfied.

Once the knowledge engineer encodes a specific configuration strategy into the knowledge base, a test configuration can be run using a "TRACE" option so that the user interface and explanation facility (19 FIG. 1) generates a comprehensible translation of the configuration strategy. For the MINI-1000 minicomputer, for example, the trace in the typescript of Appendix III (A) explains in detail the specific configuration strategy that is followed. This explanation includes the specific task block statements that are executed and any action taken upon application of the constraints. The configuration strategy for the MINI-1000 minicomputer order in Appendix II (A) first looks at the set of ordered parts to determine that the product line is a MINI-1000 minicomputer. These parts are dumped into the root bin MINI(1) in FIG. 6. Then the parts implementing the MAIN-FRAME function are found and the MAIN-FRAME(1) bin is created and these parts are transferred to the MAIN-FRAME(1) bin. In recursive fashion, the parts in the MAIN-FRAME(1) bin are found which implement the CPU function. A CPU(1) bin is created and these parts are transferred to the CPU(1) bin. The requirements for the CPU(1) bin are checked and it is found that a required CPU board is missing. The knowledge system 10 asks the user for permission to add one CPU-1000 circuit board, thereby completing the CPU(1) bin. Execution returns to the MAIN-FRAME task block. The components implementing the MEMORY function are found and a new MEMORY(1) bin is created to receive these parts. In recursive fashion, the parts in the MEMORY(1) bin implementing the RAM function are found and a new RAM(1) is created to receive these parts. The contents of the RAM(1) bin are checked to ensure that at least one RAM-1000 circuit board is included. Execution then returns to the MEMORY task block which finds the parts implementing the ROM function and creates a new ROM(1) bin to receive these parts. The contents of the ROM(1) bin are checked to ensure that at least one ROM-1000 circuit board is included. Then, execution returns to the MEMORY task block which checks the overall requirements for the MEMORY function. In particular, a MEMORY.CT1 constraint is applied which counts the total number of RAM and ROM boards to ensure that the total number does not exceed seven. When this constraint is applied, it is found that the total number does not exceed seven. Execution then returns to the MAIN-FRAME task block. Next, the MAIN-FRAME task block finds all of the parts in the MAIN-FRAME(1) bin which implement the I/O function, creates a new I/O(1) bin and transfers these parts to the new bin. Execution passes to the I/O task block which finds all of the components in the I/O(1) bin implementing the TERMINAL function, and a new TERMINAL(1) bin is created to receive these parts. Upon execution of the TERMINAL task block, the requirements or constraints for the TERMINAL function are applied. It is found that there is at least one terminal in the TERMINAL(1) bin. It is also found that the number of terminals does not exceed four. Execution then returns to the I/O task block.

The I/O task block next configures the DISC function before it configures the INTERFACE function since the interface constraints need to know how may disk drives have been ordered. The DISC function is configured by first finding the components in the I/O(1) bin which implement the DISC function, and creating a new DISC(1) bin to receive these components. Execution is passed to the DISC task block which first checks the requirements for the DISC function. No constraints are found associated with the DISC function. Execution returns to the I/O task block. Since the INTERFACE task block needs to known how may DISC units are included in the order, a bin variable called "IF-CARDS.NEEDED" is defined for the I/O function. A DETERMINE operation is performed to determine the minimum number of interface cards required for the number of terminals and disk drives in the order. After it is determined that there are four terminals and three disk drives so that four interface cards are required, the components in the I/O(1) bin implementing the INTERFACE function are found and a new INTERFACE(1) bin is created to receive these components. Execution then passes to the INTERFACE task block.

The requirements for the INTERFACE function are checked and it is found that no interface cards were ordered. A modification constraint IF.REQ.CT1 is applied which tells the user that the minimum number of interface cards have not been ordered, and ask permission to add four IF-1000 interface cards. These cards are added after permission is received from the user. Execution then returns to the I/O task block which in turn passes execution back to the MAIN-FRAME task block. In the MAIN-FRAME task block, all of the components in the MAIN-FRAME(1) bin are found which implement the CRAD-CAGE function. A new CARD-CAGE(1) is created to receive these components. Execution is passed to the CARD-CAGE task block which checks the CARD-CAGE requirements. The constraint AT-MOST-ONE.CT ensures that no more than one card cage was ordered. The constraint NO-PARTS.CT checks to ensure that at least one card cage was ordered and finds that a card cage was not ordered, and asks permission from the user to add a card cage. After receiving permission from the user, one CAGE-1000 is added to the order. Execution then returns to the MAIN-FRAME task block.

The MAIN-FRAME task block checks the requirements for the MAIN-FRAME function. The MAIN-FRAME.CT1 constraint is applied which ensures that the total number of ROM cards, RAM cards, and IF cards plus one CPU card is no greater than sixteen, so that no action need be taken. Execution then returns to the MINI task block.

In the MINI task block, all of the components in the MINI(1) bin are found that implement the POWER-SUPPLY function, and a new POWER-SUPPLY(1) is created to receive these components. Execution is transferred to the POWER-SUPPLY task block. The initial contents of the POWER-SUPPLY(1) bin are first checked to determine the number of power supplies included in the order. It is found that no power supplies

were ordered. Next, the requirements for the POWER-SUPPLY function are checked and a set of modification constraints are applied in order to determine the particular power supply to be added to the order. Thus, several power supply modification constraints are applied which use a bin variable called "POWER.NEEDED".

It should be noted that the task blocks for the minicomputer do not explicity have a DETERMINE operation to determine the value of the "POWER.-NEEDED" bin variable. A DETERMINE operation is implicity performed when a constraint is applied that references a bin variable. Upon determining the value of the bin variable "POWER.NEEDED" it is found that the required power to operate the minicomputer is 8.3 amperes. The application of the power supply modification constraints determines that a 10 ampere power supply is needed and one SUPPLY-10 is automatically added to the order. Execution then returns to the MINI task block which in turn passes execution back to the unassigned task block which is listed above in the specification. The final contents of the UNASSIGNED bin are checked to ensure that all of the original components have been considered during the configuration of the minicomputer. It is possible, for example, that some superfluous parts were included in the order, and at this point in the UNASSIGNED task block the superfluous components would be left in the UNASSIGNED bin. It is found, however, that the UNASSIGNED bin is empty, so that no superfluous components were ordered. Next, the list of final parts are determined for the production request. Finally, the final parts are transmitted to the user.

Inspection of the constraints, bin variable declarations and knowledge base functions for the M1234 computer and the MINI-1000 minicomputer knowledge bases in Appendices II (D) and IV (D) reveal that there are many built-in functions available to the knowledge engineer which simplify the knowledge engineer's task of creating the task blocks, constraints, and knowledge base functions. Moreover, the knowledge base is organized so that the knowledge engineer has ready access to the conditions of the parts upon which the constraints and knowledge base functions are responsive and can manipulate the working configuration in any desired fashion.

An operation (ASSIGN parts bin) moves the parts specified by a list of PRODUCT.IDs from the current bin into the specified bin.

If the operation (BVSET binvar newvalue noerror) assigns (newvalue) as the value of (binvar) after checking the (newvalue) correspondence to the format indicated by the LEGAL.VALUE field of the (binvar). The value (newvalue) is typically computed by a KBFUNCTION defined by the knowledge engineer. If (noerror) is true, the checking that the value (binvar) matches the format prescribed in its LEGAL.VALUES field is not performed.

The function (FETCH binvar noerror) returns the value of (binvar) if (binvar) already has a value. If (binvar) does not have a value, FETCH determines that value using the KBFUNCTIONS specified on its HOW.TO.DETERMINE field. If (noerror) is false, the FETCH function checks to make sure that the value of the (binvar) matches the format prescribed in its LEGAL.VALUE field.

The function (MEMBER element list) tests whether (element) is a member of (list). The function MEMBER returns true (T) if the element is a member of the list, and false (NIL) if not.

The function (FIND number bins description) searches the CONTENTS of (bins) for a certain (number) of parts (PRODUCT.IDs) that satisfy the (description). The FIND function returns a list of PRODUCT.IDs. The (number) of parts defined may be an integer, the word for an integer between ZERO and TEN, or the word ALL; any words must be quoted. The set of (bins) to search are specified by the following set of key words:

CCS	The current bin;
SCS	The immediate parent bin;
CSCS	The current and immediate parent bins;
ACS	The current and all ancestor bins; or
ICS	The current and all decendant bins.

The (description) is a Boolean expression made up of one or more logical clauses and connectors AND and OR. The clause (IMPLEMENT<FUNCTION.-CLASS>) succeeds if the part has an IMPLEMENTS.FN that COMPOSES the <FUNCTION.CLASS> specified. The clause (ISA<-PART>) succeeds if the part matches this <PART> name. The clause (ISA.SPEC<FUNCTION.-CLASS>) succeeds if the part is a PROD.GEN of the <FUNCTION.CLASS> specified.

Many of the built-in functions recognized by the knowledge base interpreter **13** of the knowledge system **10** (FIG. **1**) have particular utility in conjunction with constraints. Constraints always apply to a specified set of parts. The parts are specified by a list of BIN.-TYPES, which are the FUNCTION.CLASS entries in the functional hierarchy that the constraint applies to. The constraint also includes a STATEMENT of conditions to test against the parts in those bins. The STATEMENTs often test relationships between values of different bin variables, which must be declared in the BIN.VARS.REQUIRED field of the constraint declaration. Also, a list of CHECK.KEYWORDS may be specified to control when certain constraints are applied. These keywords are used by the CHECK operations in the task block steps to select relevant subsets of constraints to apply at different times in a task block.

The knowledge system provides a number of important specializations of the CONSTRAINT knowledge base type, which are further described below. All product type and specific constraints are defined in terms of these specializations. The system also considers product expansions as a particular kind of CONSTRAINT since they add parts to a bin if certain conditions, such as whether a particular product was ordered, are met. In general, the types of constraints include constraints on specific parts, warning constraints, and modification constraints.

There are five types of constraints on specific parts. The type EXPANSION.CT expands a composite part into its component parts if certain conditions are met. The type ALWAYS.EXPAND.CT unconditionally expands a composite part into its component parts. The type FULLY.EXPAND.CT expands a composite part into its component parts, and then expands all of those parts that are expandable. The type EFFECTIVELY.CT allows the knowledge engineer to specify a date after which the specified part is available for the specified product line. The type PHASE-OUT.CT al-

lows the knowledge engineer to specify a date after which the specified part is no longer available for the specified product line.

There are two types of warning constraints. The type WARN-ON-SUCCESS.CT warns the user if the conditions in the STATEMENT for the constraint succeed. The type WARN-ON-FAILURE.CT warns the user if the conditions in the STATEMENT fail.

There are two types of modification constraints. The type MODIFY-ON-SUCCESS.CT outputs a warning message and modifies the contents of the current bin if the conditions in the STATEMENT succeed. The type MODIFY-ON-FAILURE.CT outputs a warning message and modifies the contents of the current bin if the conditions in the STATEMENT fail.

A constraint STATEMENT expresses a set of conditions that the order must meet to be valid. The STATEMENT of a constraint is a Boolean expression composed of "clauses", separated by the usual logical connectives AND and OR. Individual clauses (or the entire STATEMENT expression) may be negated by using the form (NOT<EXPRESSION>).

In general the STATEMENT clauses test various relationships between expressions of bin variables. The expressions may include the typical numerical relationships <, >, =, >=, <=, and <> (not equal to).

Clauses can also evaluate arbitrary predicates written by the knowledge engineer as KBFUNCTIONs. The system provides several predefined functions such as COUNT.PARTS, ORDERED?, and SYSTEM that enhance the transparency of the constraint statements. Moreover, the system can evaluate clauses expressed in a limited, English-like format. These expressions test whether a specified quantity IS (or IS NOT) ZERO, NULL, ATOMIC, NUMERIC, REAL, or a STRING.

Constraint STATEMENTs typically reference the values of bin variables. The bin variables may be referenced directly by the FETCH function in a STATEMENT. Alternatively, the system can directly interpret references to bin variables when the bin variables are specifically declared in the BIN.VARS.REQUIRED field of the constraint declaration.

The function (COUNT.PARTS part search) returns the number of (parts) in the current bin. If (search) is true (T), the function COUNT.PARTS will search the bin tree below and including the current bin returning the total number of (parts) in these bins.

The function (ORDERED? parts search) returns the value true (T) if there is at least one instance of any of the (parts) in the current bin. If the value of (search) is true (T), ORDERED? will search the bin tree below (and including) the current bin, returning true (T) if it finds instances of (parts) in these bins. The function (SYSTEM system-name1 system-name2 . . .) tests whether the system type or product model of the order is one of the (system-names) indicated in the argument list. Each (system-name) should be a FUNCTION.-CLASS having a PRODUCT.GEN declaration including SYSTEM.

The function (COUNT list) returns the number of elements in the (list).

For constraints that modify the contents of the current bin, the system provides a special form for specifying a list of parts to add to, delete from, or replace the contents of a bin. This special form is called a PRODUCT.QTY or "product quantity". PRODUCT.QTYs are specified in the TO field of the different EXPANSION.CTs, and they are computed by the ADD-PARTS, DEL-PARTS, and SET-PARTS fields of the MODIFY-ON-SUCCESS.CTs and MODIFY-ON-FAILURE.CTs. Special functions described further below can be used to compute these PRODUCT.QTY description for these constraints.

A PRODUCT.QTY consists of a list of lists where each sublist specifies a quantity and a part type.

The function (FIND* number bins description) searches a set of (bins) for a certain (number) of parts that satisfy the (description). FIND* returns a PRODUCT.QTY that describes the quantities of the parts found.

The function (CHOOSE one parts) is given a specified PRODUCT.QTY description of several different parts, and asks the user to select one part from (parts) and returns a PRODUCT.QTY describing a single <SELECTED-PART>.

As noted above, an expansion rule is a kind of constraint that applies to specific parts. Expansion rules are specified by EXPANSION.CT objects in the knowledge base. The FROM field of an EXPANSION.CT specifies the composite part to be expanded; the TO field specifies the list of component parts to add to the bin; the WHEN field specifies the conditions under which this expansion should occur; and the MESSAGE field specifies the messages that should be printed when the expansion succeeds or fails.

Modification constraints are defined in the knowledge base either as MODIFY-ON-SUCCESS.CT objects or a MODIFY-ON-FAILURE.CT objects. A warning message is given to the user and the contents of the current bin are modified if the conditions in the STATEMENT field succeed or fail, respectively. The knowledge engineer may specify a MESSAGE that will warn the user that a modification must be made. The modifications are specified by providing values for one or more of the fields SET-PARTS, ADD-PARTS, and DEL-PARTS. If a list of parts is specified for the field SET-PARTS, the current bin is emptied and the list of parts (in the quantities specified) is placed in the bin. Then, if a list of parts is specified in the ADD-PARTS field, new parts are added to the bin in the quantities specified. If a list of parts is provided in the DEL-PARTS field, existing parts are deleted from the bin in the quantities specified.

In the knowledge base, the BIN.VAR declarations must include a LEGAL.VALUE field indicating the legal value for the bin variable (e.g., integer, string, etc.), a HOW.TO.DETERMINE field including a list of KBFUNCTIONs that can be used to determine the value of the bin variable, and a BIN.TYPES field including the bin types or function classes with which the variable is associated. It should be noted that values of BIN.VAR instances can be referenced by constraints applied by task blocks that are lower in the bin tree than where the BIN.VAR instance is created. Thus, the BIN.TYPES should associate the bin variable with bins high enough in the bin tree so that the bin variables can be referenced by all of the appropriate contraints.

The system provides two built-in KBFUNCTIONS that can be used to ask the user a question to determine the value of specified BIN.VARs. The (ASK.QUESTION) function uses the DESCR field of the BIN.VAR to form the question "What is <DESCR>?" The ASK.QUESTION function uses the optional help field of the BIN.VAR declaration to provide help to the user when the user types a question mark (?) in response to the question. Also, the response to the question is

checked against the LEGAL.VALUES expression of the BIN.VAR declaration. The second function (ASK-.BOOLEAN.QUESTION) uses the DESCR field of the BIN.VAR declaration to form the question "Is it the case that <DESCR>?" The ASK-.BOOLEAN.QUESTION function accepts only responses of YES or NO and prints, "Answer YES or NO please" if any other response is given.

It should be noted that the knowledge base itself may be divided into sections so that the sections my be identified and referenced by the task blocks and constraints. The name for each portion of the knowledge base is preferably composed of a knowledge base prefix and a unique suffix identifying the particular portion or file of the knowlege base. In the preferred embodiment of Appendix II (A)-(E) the suffix FH identifies the file containing the functional hierarchy and the product line task blocks. The suffix NCT identifies the file containing the product line constraints, the bin variables, and the knowledge base functions. The suffix BX contains product line descriptions. The suffix AK identifies the file containing product line packages. The suffix FT identifies the file containing product line features. The suffix KT identifies the product line kits. The suffix KV identifies the file containing the product line kit versions. The suffix XT identifies a file containing miscellaneous parts. The suffix EX identifies the file containing product expansions. A knowledge base file with the suffix MODS is used to record all modifications or changes made to the knowledge base since the last modification by the knowledge engineer using the knowledge base maintenance facility **21** (FIG. **1**).

The knowledge base maintenance facility **21** preferably has a number of commands used by the knowledge engineer to create, maintain, and augment the knowledge base **14**. These commands include, for example, eight commands for knowledge base data and control functions, three commands for knowledge base editing functions, and four commands for knowledge base file maintenance.

The eight knowledge base data and control commands include SELECT, PP, CHANGES, CONTENTS, CONTROL, IGNORE.CHANGES, REMEMBER.CHANGES, and LIST.KBSETS.

The command SELECT selects a knowledge base to be modified and loads the knowledge base into the host computer system so that it may be further manipulated by the knowledge base maintenance facility **21**.

The command PP displays the status of the current knowledge base, including whether changes are being recorded, whether the changes will be automatically saved, how many modifications have been made since the last time the knowledge base was saved, (i.e., written back from the working memory of the knowledge base maintenance facility **21** to the more permanent memory from which the knowledge base was loaded), and how many objects are currently in the current knowledge base.

The CHANGES command displays the modifications made to the knowledge base since the most recent migration, (see MIGRATE below) and indicates which objects have been added, changed, or deleted from the knowledge base.

The command CONTENTS displays the names of all of the objects in the current knowledge base.

The command CONTROL controls whether the system automatically saves modifications after a certain number of additions, edits, and deletions have been made to the current knowledge base.

The command IGNORE.CHANGES tells the knowledge base maintenance facility **21** to ignore any subsequent additions, modifications, or deletions.

The command REMEMBER.CHANGES turns off the effect of a previous IGNORE.CHANGES command so that subsequent changes will be recorded.

A command LIST.KBSETS displays the contents of specified main knowledge base sets after asking the user to specify the names of the sets.

The three knowledge base editing commands include ADD, EDIT, and DELETE. The command ADD adds new objects to the current knowledge base and prompts the knowledge engineer for the knowledge base type and the name of the new knowledge base object. The command EDIT modifies existing objects in the current knowledge base and prompts for the name of the knowledge base object. The command DELETE removes objects from the current knowledge base and prompts for the name of an existing knowledge base object.

The four knowledge base file maintenance commands include SAVE, MIGRATE, BUILD, and CREATE. The command SAVE saves the copy of the current knowledge base changes in a modification file. The command MIGRATE distributes the current knowledge base modification into the main knowledge base files and writes out a new, empty copy of the modifications file. The command BUILD saves a copy of the current executable system on a file and prompts the knowledge engineer for the name of the file. The command CREATE creates a new knowledge base for a new product line, and automatically prompts for a knowledge base prefix to identify the new product line.

The preferred embodiment of the user interface and explanation facility **19** has several commands to enable the user to obtain an explanation of the operation of the system **10**. The user can select options for either obtaining an explanation as a configuration is being run, or after the completion of a configuration. As a configuration is being run, the user can select one or more of six options. The command SHOW-STEPS instructs the explanation facility **19** to print the task block steps as they are executed. The command SHOW-DETERMINATION causes the explanation facility to print the values of bin variables as they are determined. The command SHOW-EXPANSIONS tells the explanation facility to print a description of the expansions applied to parts in the order. The command SHOW-PARTS causes the explanation facillity to indicate when parts are added to or deleted from a bin's contents. The command SHOW-FULL-TRACE or ALL causes the explanation facility to indicate all of the events described above.

After an order has been configured and the complete trace has been recorded in the trace memory **17**, the user may enter commands to access the trace memory **17**. The command SHOW-TRACE causes the explanation facility to replay the trace of the most recent order. The command SHOW-VALUE causes the explanation facility to display the values of the bin variables used in the most recent order. The command SHOW-PART-HISTORY displays the parts history of the most recent order. The command FORGET causes the explanation facility to erase the trace memory **17** so that there is no longer a history of the prior consultation.

The knowledge system **10** has been described above primarily for checking and modifying the order lines in an order for a flexibly assembled product. It should be noted, however, that the modification constraints in part redesign the ordered product. The system **10** is equally useful as a design tool. The design procedure is illustrated by the typescript in Appendix III (B) wherein an order for a MINI-1000 minicomputer merely includes the model number of the desired product. The set of modification constraints in the knowledge base for the MINI-1000 product line (Appendices IV (A)–(D)) designs a minimal system consistent with the modification constraints including one CPU-1000 card, one RAM-1000 card, one ROM-1000 card, one TERMINAL-1000 computer terminal, one IF-1000 card, one CAGE-1000 card cage, and one SUPPLY-05 five ampere power supply.

To use the knowledge system **10** as shown in FIG. **1** as an expert design tool, the initial configuration **11** should include not only an initial set of parts but an initial set of values for bin variables to specify the desired parameters of the final configuration. The initial values for bin variables, for example, may specify the desired conditions and relations between the parts being assembled, and may also specify what modification constraints are to be applied as the product is being built in the working configuration memory **15**.

It should also be noted that the knowledge engineer has great flexibility in defining the control procedure for the knowledge base interpreter **13**. The knowledge engineer, for example, may write specialized knowledge base functions and task blocks to perform specialized configuration operations. The knowledge engineer may also modify the built-in control procedure of the knowledge base interpreter **13** to modify the control procedure in any desired fashion.

In view of the above, a knowledge system has been described for generalized representation and processing of hierarchical assemblies. The configuration strategies and actions are defined in task blocks which are completely separate from the functional hierarchy which describes the generic configuration of the product and the parts catalog which defines the individual elements available for configuration. Moreover, the constraints are completely separate from the task blocks, functional hierarchy, and parts catalog so that the definition of configuration constraints are clearly separated from configuration checking strategies and actions. The knowledge base interpreter has a built-in control procedure and built-in functions which enable the knowledge engineer to design and implement readily a desired configuration checking strategy. Since the imperative language of the task blocks clearly defines the configuration checking strategies and actions, a trace of a configuration operation is easily stored in a trace memory including both the actual steps executed and the resulting actions such as warnings or modifications to the configuration. Thus, an explanation facility can be provided which generates an intelligible and comprehensible explanation of the configuration based on the record in the trace memory.

APPENDICES I–V

TABLE OF CONTENTS

APPENDIX I
CONSULTATION TYPESCRIPT FOR THE PROCESSING OF AN ORDER FOR AN M1234 COMPUTER
APPENDIX II (A)
KNOWLEDGE BASE FOR M1234 COMPUTER FUNCTIONAL HIERARCHY
APPENDIX II (B)
KNOWLEDGE BASE FOR M1234 COMPUTER TASK BLOCKS
APPENDIX II (C)
KNOWLEDGE BASE FOR M1234 COMPUTER PARTS CATALOG
APPENDIX II (D)
KNOWLEDGE BASE FOR M1234 COMPUTER CONSTRAINTS & BIN VARIABLES & KNOWLEDGE BASE FUNCTIONS
APPENDIX II (E)
KNOWLEDGE BASE FOR M1234 COMPUTER EXPANSION RULES
APPENDIX III (A)
MINICOMPUTER ORDER CHECKING TYPESCRIPT
APPENDIX III (B)
MINICOMPUTER DESIGN TYPESCRIPT
APPENDIX IV (A)
KNOWLEDGE BASE FOR MINICOMPUTER FUNCTIONAL HIERARCHY
APPENDIX IV (B)
KNOWLEDGE BASE FOR MINICOMPUTER TASK BLOCKS
APPENDIX IV (C)
KNOWLEDGE BASE FOR MINICOMPUTER PARTS CATALOG
APPENDIX IV (D)
KNOWLEDGE BASE FOR MINICOMPUTER CONSTRAINTS & BIN VARIABLES & KNOWLEDGE BASE FUNCTIONS
APPENDIX V
GLOSSARY

APPENDIX I

CONSULTATION TYPESCRIPT FOR THE PROCESSING OF AN ORDER FOR AN M1234 COMPUTER

```
The following is an example M1234 consultation with OCEAN.

Welcome to OCEAN

OCEAN executive> RUN

Processing new order...
Enter name of order file: T
```

```
Reading order information...
Order information> ORDER NUMBER: 6330-82-0261
Order information> BRANCH: 1815
Order information> METHOD: SURFACE
Order information>
Reading parts to check...
Order line> M1234
=>  1 M1234                    The M1234 product line
Order line> 1234-3003-0690
=>  1 1234-3003-0690           Class model for M1234 system
Order line> 1234-F010
=>  1 1234-F010                Firmware
Order line> 1234-F290
=>  1 1234-F290                Memory 2M, 2 way
Order line> 1234-F361
=>  1 1234-F361                Common trunk
Order line> 1234-F691
=>  1 1234-F691                BSLA
Order line> 1234-F680
=>  1 1234-F680                MFG.PLANT BYTE MULTIPLEX TRUNK
Order line>
Reading parts to be shipped F/S...
F/S order line>
Enter order processing options: ?
Please select from among CHECK, DON'T-CHECK, EDIT-ORDER,
QUIETLY, SHOW-COMPILATION, SHOW-CONSTRAINTS, SHOW-
DETERMINATION, SHOW-EXPANSIONS, SHOW-FULL-TRACE, SHOW-PARTS,
SHOW-STEPS, or other
Enter order processing options: SHOW-FULL-TRACE

    Adding the following parts: 1 M1234, 1 1234-3003-0690, 1
    1234-F010, 1   1234-F290, 1 1234-F361, 1 1234-F691 and 1
    1234-F680

Evaluating: (DETERMINE '(TODAYS.DATE ORDERED.PARTS
ORDERED.PRODUCT.LINE))  Today's date of original
components is set to: 15-Mar-84 The ordered set of parts
of original components is set to: M1234, 1234-3003-0690,
1234-F010, 1234-F290, 1234-F361, 1234-F691 and 1234-F680
The ordered product line of original components is set
to: M1234 product line components

Evaluating: (IF CHECKING.ORDER
                THEN   **COMMENT**
                     (CONFIGURE (FETCH
                          'ORDERED.PRODUCT.LINE)
                               (FIND 'ALL 'CCS))
                     (CHECK 'EMPTY.BIN))

Configuring M1234 product line components
Initial parts are: 1 M1234, 1 1234-3003-0690, 1 1234-
F010, 1 1234-F290, 1 1234-F361, 1 1234-F691 and 1 1234-
F680

Evaluating: (BVSET 'SYSTEM.TYPE
(COMPUTE.M1234.SYSTEM.TYPE))
The M1234 system type of original components is set to:
M1234 system components
```

```
Evaluating: (CONFIGURE (FETCH 'SYSTEM.TYPE)
                       (FIND 'ALL 'CCS))

Configuring M1234 system components
     Initial parts are: 1 M1234, 1 1234-3003-0690, 1
     1234-F010, 1 1234-F290, 1 1234-F361, 1 1234-F691
     and 1 1234-F680

     Evaluating: (EXPAND* '(BG BU.M12))
     Checking constraint: 1234-3003-0690.ECT.1
       (a ALWAYS.EXPAND.CT)
     Expanding 1234-3003-0690 to 1 1234-F262, 1 1234-
     F113, 1 1234-K915, 1 1234-K450, 1 1234-K917, 1
     1234-K416 and 1 1234-K941 in M1234 system
     components.
     Adding the following parts: 1 1234-F262, 1 1234-
     F113, 1 1234-K915, 1 1234-K450, 1 1234-K917, 1
     1234-K416 and 1 1234-K941

     Evaluating: (DETERMINE '(CLASS.MODEL.EXPANSION
     BASIC.TO.PRICED
                                    MP.ORDER))
     The class model parts of M1234 system components is
     set to: 1 1234-F262, 1 1234-F113, 1 1234-K915, 1
     1234-K450, 1 1234-K917, 1 1234-K416 and 1 1234-K941
     The basic-to-priced parts of M1234 system
     components is set to: 1 1234-K416, 1 1234-K920, 2
     1234-K924, 1 1234-K940 and 1 1234-K941
     This is a multi-processing order of M1234 system
     components is set to: False

     Evaluating: [CONFIGURE 'MULTIPROCESSING
                    (FIND 'ALL 'CCS '(IMPLEMENTS
                                        MULTIPROCESSING]

     Configuring multi-processing components
       No initial parts

        Evaluating: (CHECK 'CONTENTS)
        Checking CONTENTS constraints in multi-
         processing components
          No CONTENTS constraints were found.
        Evaluating: (EXPAND* '(AK FT KT))

        Evaluating: (DETERMINE '(MP.PIBS))
        PIBS required for MP kits of M1234 system
           components is set to: 0

     Evaluating: [CONFIGURE 'MAINFRAME (FIND 'ALL 'CCS
                                    '(IMPLEMENTS
                                        MAINFRAME]
     Configuring mainframe components
       Initial parts are: 1 1234-F010, 1 1234-F290, 1
     1234-F361, 1 1234-F691, 1 1234-F680, 1 1234-F262, 1
     1234-F113, 1 1234-K915, 1 1234-K450, 1 1234-K917
     and 1 1234-K941
```

```
Evaluating: [CONFIGURE 'PROCESSOR (FIND 'ALL 'CCS
                                  '(IMPLEMENTS
                                      PROCESSOR]

Configuring processor components
  Initial parts are: 1 1234-K941

   Evaluating: (EXPAND* '(AK FT KT))

   Evaluating: [CONFIGURE 'MEMORY (FIND 'ALL 'CCS
                                  '(IMPLEMENTS
                                      MEMORY]

Configuring main memory components
  Initial parts are: 1 1234-F290 and 1 1234-F262
  Evaluating: (CHECK 'CONTENTS)
  Checking CONTENTS constraints in main memory
  components
    The following constraints apply: AT-MOST-
  ONE-MEMORY-FT.CT
  Checking constraint: AT-MOST-ONE-MEMORY-FT.CT
  (a MODIFY-ON-SUCCESS.CT)

    The statement: ((SYSTEM M1234.SYSTEM)
                    AND
                    [COUNT (FIND 'ALL 'CCS
                           '(ISA FT]
                    > 1)
    succeeded with value T when applied to main
  memory components
   Removing the following parts: 1 1234-F262

  Evaluating: (EXPAND* '(AK FT KT))
  Checking constraint: 1234-F290.ECT.1 (a
    ALWAYS.EXPAND.CT)
  Expanding 1234-F290 to 4 1234-K920, 2 1234-
  K921, 8 1234-K924 and 2 1234-K940 in main
  memory components.
  Adding the following parts: 4 1234-K920, 2
  1234-K921, 8 1234-K924 and 2 1234-K940

  Evaluating: (DETERMINE '(MAIN-MEMORY.SUPPLIED
                           MEMORY.INTERLEAVING
                              MEMORY.PIBS))
  The main memory supplied with this order (in
  kb) of M1234 system components is set to: 2048
  The memory interleaving for this order of
  M1234 system components is set to: 2-WAY
  PIBS required for main memory w/o extra
  plenums of M1234 system components is set to:
  16

  Evaluating: (CHECK 'INTERLEAVING)
  Checking INTERLEAVING constraints in main
  memory components
    The following constraints apply:  2-WAY-
      INTERLEAVING
```

```
Checking constraint: 2-WAY-INTERLEAVING (a
  INTERLEAVING.CT)
  The statement: [(SYSTEM M1234.SYSTEM)
                  AND
                  (NOT (MEMORY.INTERLEAVING =
                      '2-WAY]
failed with value NIL when applied to main
  memory components
Evaluating: (CHECK 'MEMORY-CAPACITY)

Checking MEMORY-CAPACITY constraints in main
memory components
   No MEMORY-CAPACITY constraints were found.
Evaluating: (CHECK 'MEMORY-INCREMENT)
Checking MEMORY-INCREMENT constraints in main
memory components

   The following constraints apply: 1024-
INCREMENT
Checking constraint: 1024-INCREMENT (a
MEMORY-INCREMENT.CT)

The statement: ((SYSTEM M1234.SYSTEM)
                AND
               (NOT (INCREMENT MAIN-
                  MEMORY.SUPPLIED 1024)))
     failed with value NIL when applied to
     main memory components

Evaluating: [CONFIGURE 'ISU (FIND 'ALL 'CCS
                      '(IMPLEMENTS ISU]

Configuring ISU memory components
  Initial parts are: 1 1234-F113, 1 1234-K915
    and 1 1234-K917

Evaluating: (CHECK 'CONTENTS)
Checking CONTENTS constraints in ISU memory
components
  The following constraints apply: AT-LEAST-
     ONE-FT.CT AT-MOST-ONE-FT.CT
Checking constraint: AT-LEAST-ONE-FT.CT (a
     WARN-ON-SUCCESS.CT)

  The statement: ([COUNT (FIND 'ALL 'CCS '(ISA
    FT]
                  IS ZERO)
   failed with value NIL when applied to ISU
      mmory components
 Checking constraint: AT-MOST-ONE-FT.CT (a
      MODIFY-ON-SUCCESS.CT)

   The statement: ([COUNT (FIND 'ALL 'CCS
                     '(ISA FT]
                   > 1)
   failed with value NIL when applied to ISU
      memory components
```

```
Evaluating: (EXPAND* '(AK FT KT))
Checking constraint: 1234-F113.ECT.1 (a
     ALWAYS.EXPAND.CT)

  Expanding 1234-F113 to 2 1234-K940 and 3
       1234-K913 in ISU memory components.
  Adding the following parts: 2 1234-K940 and 3
       1234-K913

  Evaluating: (DETERMINE '(ISU-
       MEMORY.SUPPLIED))
  The isu memory supplied with this order (in
       kb) of M1234
  system components is set to: 24

  Evaluating: (CHECK 'REQUIREMENTS)
  Checking REQUIREMENTS constraints in ISU
       memory components
    The following constraints apply: MP.ISU.CT
  Checking constraint: MP.ISU.CT (a WARN-ON-
       SUCCESS.CT)

    The statement: ((SYSTEM M1234.SYSTEM)
                    AND MP.ORDER AND (ISU-
                   MEMORY.SUPPLIED < 40))
    failed with value NIL when applied to ISU
       memory components

Evaluating: [CONFIGURE 'FIRMWARE (FIND 'ALL 'CCS
                                  '(IMPLEMENTS
                                      FIRMWARE]


Configuring firmware components
  Initial parts are: 1 1234-F010

  Evaluating: (CHECK 'REQUIREMENTS)
  Checking REQUIREMENTS constraints in firmware
       components
    The following constraints apply:
       I9050.FIRMWARE.CT1
    MP.FIRMWARE.CT FIRMWARE.CT
  Checking constraint: I9050.FIRMWARE.CT1 (a
  MODIFY-ON-SUCCESS.CT)

    The statement: [(SYSTEM I9050-1.SYSTEM
       I9050-2.SYSTEM I9050-3.SYSTEM)
                    AND
                    (NOT (ORDERED? '(1234-F018
                       1234-K018]
    failed with value NIL when applied to
       firmware components
  Checking constraint: MP.FIRMWARE.CT (a
  MODIFY-ON-SUCCESS.CT)
```

```
    The statement: [(SYSTEM M1234:SYSTEM)
                    AND MP.ORDER AND (NOT
       (ORDERED? '5640-F400]
    failed with value NIL when applied to
       firmware components
  Checking constraint: FIRMWARE.CT (a MODIFY-
       ON-SUCCESS.CT)

    The statement: [(SYSTEM M1234.SYSTEM)
                    AND
                    (NOT MP.ORDER)
                    AND
                    (NOT (ORDERED? '(1234-F010
                       1234-K010 1234-F011
                       1234-K011]
    failed with value NIL when applied to
       firmware components

  Evaluating: (CHECK 'CONTENTS)
  Checking CONTENTS constraints in firmware
       components
    The following constraints apply: AT-LEAST-
       ONE-FT.CT
    AT-MOST-ONE-FT.CT
  Checking constraint: AT-LEAST-ONE-FT.CT (a
       WARN-ON-SUCCESS.CT)

    The statement: ([COUNT (FIND 'ALL 'CCS
       '(ISA FT]
                    IS ZERO)
    failed with value NIL when applied to
       firmware components
  Checking constraint: AT-MOST-ONE-FT.CT (a
       MODIFY-ON-SUCCESS.CT)

    The statement: ([COUNT (FIND 'ALL 'CCS
       '(ISA FT]
                    > 1)
    failed with value NIL when applied to
       firmware components

  Evaluating: (EXPAND* '(AK FT KT))
  Expanding 1234-F010 to 1234-K010 in firmware
       components.
  Adding the following parts: 1 1234-K010

Evaluating: [CONFIGURE 'IO (FIND 'ALL 'CCS
                                '(IMPLEMENTS IO]
Configuring IO components
  Initial parts are: 1 1234-F361, 1 1234-F691
       and 1 1234-F680
    Evaluating: [CONFIGURE 'IDC (FIND 'ALL 'CCS
                                '(IMPLEMENTS IDC]

    Configuring IDC components
      No initial parts
```

```
Evaluating: (CHECK 'CONTENTS)
Checking CONTENTS constraints in IDC
   components
  The following constraints apply: NO-
      FTS.CT
Checking constraint: NO-FTS.CT (a MODIFY-
  ON-SUCCESS.CT)

  The statement: ([COUNT (FIND 'ALL 'CCS
      '(ISA FT]
                  IS NOT ZERO)
  failed with value NIL when applied to
      IDC components

Evaluating: (EXPAND* '(AK FT KT))

Evaluating: [CONFIGURE 'CLC/MLA
                       (FIND 'ALL 'CCS
                             '(IMPLEMENTS
                                  CLC/MLA]

Configuring CLC/MLA components
  No initial parts

Evaluating: [CONFIGURE 'FICS
                       (FIND 'ALL 'CCS
                             '(IMPLEMENTS
                                  FICS]

Configuring fast ICS components
  No initial parts

   Evaluating: (CHECK 'CONTENTS)
   Checking CONTENTS constraints in fast
      ICS components
     No CONTENTS constraints were found.
   Evaluating: (EXPAND* '(AK FT KT))

Evaluating: [CONFIGURE 'ICS (FIND 'ALL
                  'CCS
                             '(IMPLEMENTS
                                  ICS]

Configuring ICS components
  No initial parts

   Evaluating: (CHECK 'CONTENTS)
   Checking CONTENTS constraints in ICS
       components
     No CONTENTS constraints were found.

   Evaluating: (EXPAND* '(AK FT KT))

Evaluating: (DETERMINE '(#CLC.MLA.LINES
  ICS.PIBS))
The number of clc's of M1234 system
  components is set to: 0
```

```
The number of mla's of M1234 system
  components is set to: 0
The number of clc/mla lines for this order
  of M1234 system
components is set to: 0
PIBS required for CLC/MLA of M1234 system
  components is set to: 0

Evaluating: (CHECK 'IO-CAPACITY)
Checking IO-CAPACITY constraints in
   CLC/MLA components
  The following constraints apply:
       #CLC.MLA-LINES
Checking constraint: #CLC.MLA-LINES (a
WARN-ON-SUCCESS.CT)

  The statement: ((SYSTEM M1234.SYSTEM)
                  AND
                  (#CLC.MLA.LINES > 20))
  failed with value NIL when applied to
    CLC/MLA components

Evaluating: [CONFIGURE '2STAGE-IO
                       (FIND 'ALL 'CCS
                             '(IMPLEMENTS
                                2STAGE-IO]
Configuring two-stage IO components
  Initial parts are: 1 1234-F361, 1 1234-F691
    and 1 1234-F680

   Evaluating: [CONFIGURE 'ITB-MODULES
                          (FIND 'ALL 'CCS
                             '(IMPLEMENTS
                                ITB-MODULES]

   Configuring ITB components
     Initial parts are: 1 1234-F361

     Evaluating: (CHECK 'CONTENTS)
     Checking CONTENTS constraints in ITB
         components
       No CONTENTS constraints were found.

     Evaluating: (EXPAND* '(AK FT KT))
     Checking constraint: 1234-F361.ECT1 (a
         ALWAYS.EXPAND.CT)
     Expanding 1234-F361 to 2 1234-K931 in
         ITB components.
     Adding the following parts: 2 1234-K931

     Evaluating: (DETERMINE '(#TRUNK.LINES
         IO.PIBS))
     The number of standard io lines for
         this order of M1234
     system components is set to: 2
     PIBS required for IOLC and common trunk
        of M1234 system
     components is set to: 4
```

```
Evaluating: [CONFIGURE 'IO-PANEL-MODULES
              (FIND 'ALL 'CCS
                  (IMPLEMENTS
                      IO-PANEL-MODULES]

Configuring IO panel components
  Initial parts are: 1 1234-F691 and 1
      1234-F680

   Evaluating: (CHECK 'CONTENTS)
   Checking CONTENTS constraints in IO
       panel components
     No CONTENTS constraints were found.

   Evaluating: (EXPAND* '(AK FT KT))
   Checking constraint: 1234-F691.ECT1 (a
       ALWAYS.EXPAND.CT)
   Expanding 1234-F691 to 2 1234-K935, 2
       1234-K936 and 2 1234-K937 in IO
         panel components.
   Adding the following parts: 2 1234-
       K935, 2 1234-K936 and 2 1234-K937
   Checking constraint: 1234-F680.ECT.1 (a
       ALWAYS.EXPAND.CT)
   Expanding 1234-F680 to 1 1234-K930 in
       IO panel components.
   Adding the following parts: 1 1234-K930

   Evaluating: (DETERMINE '(#CT.LINES
       #IOLC.LINES))

                   The number of common trunk interface
                       lines of M1234
                   system components is set to: 1
                   The number of iolc interface lines of
                       M1234 system
                   components is set to: 2

                 Evaluating: (CHECK 'IO-CAPACITY)
                 Checking IO-CAPACITY constraints in two-
                   stage IO components
                   The following constraints apply: CT.IO
                   CT.IOLC.IO CT-LINES IOLC-LINES
                 Checking constraint: CT.IO (a WARN-ON-
                   FAILURE.CT)

                   The statement: (#CT.LINES <=
                         TRUNK.LINES)
                   succeeded with value T when applied to
                         two-stage IO components
                 Checking constraint: CT.IOLC.IO (a WARN-
                         ON-FAILURE.CT)

                   The statement: (#TRUNK.LINES =
                         #IOLC.LINES + #CT.LINES)
```

```
        failed with value NIL when applied to
             two-stage IO components
*** WARNING - The number of IOLC and common trunk I/F lines
does not match the number of HS, MS, and LS IO lines.

      Checking constraint: CT-LINES (a IO-
        LINES.CT)

        The statement: ((SYSTEM M1234.SYSTEM)
                        AND
                        (NOT (BETWEEN #CT.LINES
                             0 6)))
        failed with value NIL when applied to
         two-stage IO components
      Checking constraint: IOLC-LINES (a IO-
        LINES.CT)

        The statement: ((SYSTEM STSYSTEM )
                        AND
                        (NOT (BETWEEN
                          #IOLC.LINES 0 6)))
        failed with value NIL when applied to
         two-stage IO components
Evaluating: [CONFIGURE 'PLENUM (FIND 'ALL 'CCS
                                  '(IMPLEMENTS
                                     PLENUM]

Configuring plenum components
  No initial parts
   Evaluating: (CHECK 'CONTENTS)
   Checking CONTENTS constraints in plenum
        components
     The following constraints apply: AT-MOST-
        ONE-FT.CT
   Checking constraint: AT-MOST-ONE-FT.CT (a
        MODIFY-ON-SUCCESS.CT)

     The statement: ([COUNT (FIND 'ALL 'CCS
        '(ISA FT]
                    > 1)
     failed with value NIL when applied to
        plenum components

   Evaluating: (EXPAND* '(AK FT KT))

Evaluating: [CONFIGURE 'POWER-SUPPLY
                  (FIND 'ALL 'CCS
                        '(IMPLEMENTS
                           POWER-SUPPLY]
Configuring power supply components
  No initial parts

   Evaluating: (CHECK 'REQUIREMENTS)
   Checking REQUIREMENTS constraints in power
        supply components
```

```
    The following constraints apply: MP.POWER-
        SUPPLY.CT
  Checking constraint: MP.POWER-SUPPLY.CT (a
  MODIFY-ON-SUCCESS.CT)
    The statement: [(SYSTEM M1234.SYSTEM)
                    AND MP.ORDER AND
                    (NOT (ORDERED? '(1234-F400
                                     1234-K400]
    failed with value NIL when applied to power
       supply components
  Evaluating: (EXPAND* '(AK FT KT))
Evaluating: [CONFIGURE 'REST (FIND 'ALL 'CCS
                                '(IMPLEMENTS REST]
Configuring remaining components
  Initial parts are: 1 1234-K416
  Evaluating: [CONFIGURE 'CABLE (FIND 'ALL 'CCS
                                 '(IMPLEMENTS CABLE]
  Configuring cable components
    No initial parts
    Evaluating: (EXPAND* '(AK FT KT))
  Evaluating: [CONFIGURE 'CONSOLE (FIND 'ALL 'CCS
                                    '(IMPLEMENTS
                                       CONSOLE]
  Configuring console components
    Initial parts are: 1 1234-K416
    Evaluating: (EXPAND* '(AK FT KT))
  Evaluating: [CONFIGURE 'MISC (FIND 'ALL 'CCS
                                '(IMPLEMENTS MISC]
  Configuring miscellaneous components
    No initial parts
    Evaluating: (EXPAND* '(AK FT KT))

    Evaluating: (CHECK 'REQUIREMENTS)
    Checking REQUIREMENTS constraints in
        miscellaneous components
      The following constraints apply: CLC/MLA-
          DISPLAY
    Checking constraint: CLC/MLA-DISPLAY (a
    MODIFY-ON-SUCCESS.CT)

      The statement: ((SYSTEM M1234.SYSTEM)
                      AND
                      ((COUNT.PARTS '1234-K545 T)
                       = 1)
                      AND
                      (BETWEEN #CLC.MLA.LINES 11
                         20))
      failed with value NIL when applied to
          miscellaneous components
    Evaluating: (EXPAND* '(AK FT KT))
```

```
Evaluating: (CHECK 'POWER)
Checking POWER constraints in M1234 system
  components
  The following constraints apply: EXTRA.MINUS5
  EXTRA.PLUS5 MINUS5 PLUS5
 Checking constraint: EXTRA.MINUS5 (a WARN-ON-
                                      SUCCESS.CT)
 The power used by this order of M1234 system
   components is set to:  +5V: 33 -2V: 0 -5V: 226
   The statement: ((SYSTEM M1234.SYSTEM )
                   AND
                   (NOT MP.ORDER)
                   AND
                   (ORDERED? '1234-F400 T)
                   AND
                   (POWER.USED:MINUS.5 <= 250))
   failed with value NIL when applied to M1234
     system components
 Checking constraint: EXTRA.PLUS5 (a WARN-ON-
                                     SUCCESS.CT)
   The statement: ((SYSTEM M1234.SYSTEM )
                   AND
                   (ORDERED? '1234-F413 T)
                   AND
                   (POWER.USED:PLUS.5 <= 60))
   failed with value NIL when applied to M1234
     system components
 Checking constraint: MINUS5 (a MODIFY-ON-
                                   SUCCESS.CT)
 The power supplied for this order of M1234 system
   components is set to:  +5V: 0 -2V: 0 -5V: 0

   The statement: ((SYSTEM M1234.SYSTEM)
                   AND
                   (POWER.USED:MINUS.5 -
                      POWER.SUPPLIED:MINUS.5)
                   > 250)
   failed with value NIL when applied to M1234
     system components
 Checking constraint: PLUS5 (a MODIFY-ON-SUCCESS.CT)

   The statement: ((SYSTEM M1234.SYSTEM )
                   AND
                   (POWER.USED:PLUS.5 -
                    POWER.SUPPLIED:PLUS.5)
                   > 60)
   failed with value NIL when applied to M1234
       system components
 Evaluating: (CHECK 'PIBS)
 Checking PIBS constraints in M1234 system
       components
   The following constraints apply: ICS.PIBS.CT
   IO.PIBS.CT PIBS.CT1 PIBS.CT2
 Checking constraint: ICS.PIBS.CT (a WARN-ON-
                                      SUCCESS.CT)
```

```
  The statement: ((SYSTEM M1234.SYSTEM )
                  AND
                  (ICS.PIBS > 8))
  failed with value NIL when applied to M1234
      system components
Checking constraint: IO.PIBS.CT (a WARN-ON-
                                    SUCCESS.CT)

  The statement: ((SYSTEM  M1234.SYSTEM)
                  AND
                  (IO.PIBS + ICS.PIBS > 12))
  failed with value NIL when applied to M1234
      system components
Checking constraint: PIBS.CT1 (a WARN-ON-
                                    SUCCESS.CT)

  The statement: ((SYSTEM M1234.SYSTEM)
                  AND
                  (ORDERED? '1234-K926)
                  AND
                  (TOTAL.PIBS > 38))
  failed with value NIL when applied to M1234
      system components
Checking constraint: PIBS.CT2 (a MODIFY-ON-
                                    SUCCESS.CT)

  The statement: ((SYSTEM M1234.SYSTEM)
                  AND
                  (NOT (ORDERED? '1234-K926))
                  AND
                  (TOTAL.PIBS > 38))
  failed with value NIL when applied to M1234
      system components
Evaluating: (EXPAND* '(AK FT KT))

Evaluating: (CHECK 'EFFECTIVITY)
Checking EFFECTIVITY constraints in M1234 system
      components
  No EFFECTIVITY constraints were found.

Evaluating: (CHECK 'PHASE-OUT)
Checking PHASE-OUT constraints in M1234 system
      components
  No PHASE-OUT constraints were found.
Checking EMPTY.BIN constraints in M1234 system
      components
  The following constraints apply:
        UNASSIGNED.EMPTY.CT
Checking constraint: UNASSIGNED.EMPTY.CT (a WARN-
                                    ON-SUCCESS.CT)
          The statement: (BIN:CONTENTS IS NOT NULL)
          failed with value NIL when applied to original
              components
```

```
Evaluating: (DETERMINE '(FINAL.PARTS))
      The final set of parts of original components is
 set to: M1234, 1234-3003-0690, 1234-K450, 1234-K941,
 1234-F290, 1234-K920, 1234-K921, 1234-K924, 1234-K940,
 1234-F113, 1234-K915, 1234-K917, 1234-K913, 1234-F010,
 1234-K010, 1234-F361, 1234-K931, 1234-F691, 1234-F680,
 1234-K935, 1234-K936, 1234-K937, 1234-K930 and 1234-K416

Evaluating: (IF (MEMBER (FETCH 'ORDERED.PRODUCT.LINE)
                        '(PRODUCT.LINE
                           I9050.PRODUCT.LINE))
                THEN (OUTPUT.ORDER))
What type of output form would you like:
PRODUCTION.WORKSHEET
Order number: 6330-82-0261
Branch: 1815
Method: SURFACE
Remarks:
*** WARNING - The number of IOLC and common trunk I/F lines
does not match the
number of HS, MS, and LS IO lines.

1   1234-3003-0690    Class model for M1234 system      F/S

Basic To Priced
1   1234-K416         CONSOLE INDICATOR INTERFACE       F/A
1   1234-K920         MSU EXPANSION MODULE              F/A
2   1234-K924         MSU 256K ARRAY                    F/A
1   1234-K940         ISU/MSU RESISTOR                  F/A
1   1234-K941         CRYSTAL 84/56 NS                  F/A

Customer BTO
1   1234-K450         VS HARDWARE                       F/A
3   1234-K920         MSU EXPANSION MODULE              F/A
2   1234-K921         MSU, INTERFACE                    F/A
6   1234-K924         MSU 256K ARRAY                    F/A
3   1234-K940         ISU/MSU RESISTOR                  F/A
1   1234-K915         READ-ONLY MEMORY AND TERMINATOR   F/A
1   1234-K917         SERVICE PROCESSOR, ISU NO.1       F/A
3   1234-K913         ISU, 8K FAST                      F/A
1   1234-K010         Firmware                          F/A
2   1234-K931         DIRECT MEMORY ACCESS TRUNK        F/A
2   1234-K935         BIT SERIAL LINK CABLES            F/A
2   1234-K936         CONNECTOR MODULE                  F/A
2   1234-K937         CT BSL ADAPTER                    F/A
1   1234-K930         COMMON TRUNK DATA INTERFACE       F/A

What type of output form would you like:

OCEAN executive>
```

APPENDIX II (A)

KNOWLEDGE BASE FOR M1234 COMPUTER
FUNCTIONAL HIERARCHY

```
(* XTEST-FHS definition)
(DEFINSTANCE XTEST-FHS FHS
  SET.OF XTESTKB
  FILED.ON XTESTFH
  ELEMENTS          (2STAGE-IO CABLE CLC/MLA CONSOLE FICS
                    FIRMWARE ICS IDC IO IO-PANEL-MODULES ISU
                    ITB-MODULES MAINFRAME MEMORY MISC MULTI-
                    PROCESSING PLENUM POWER-SUPPLY PROCESSOR
                    PRODUCT.LINE REST SP M1234.SYSTEM))
(DEFCLASS 2STAGE-IO (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "two-stage IO"
   COMPOSES (IO))
  )

(DEFCLASS CABLE (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "cable"
   COMPOSES (REST))
  )

(DEFCLASS CLC/MLA (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "CLC/MLA"
   COMPOSES (IO))
  )

(DEFCLASS CONSOLE (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "console"
   COMPOSES (REST))
  )

(DEFCLASS FICS (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "fast ICS"
   COMPOSES (CLC/MLA))
  )
(DEFCLASS FIRMWARE (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "firmware"
   COMPOSES (MAINFRAME))
  )
```

```
(DEFCLASS ICS (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "ICS"
   COMPOSES (CLC/MLA))
  )

(DEFCLASS IDC (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "IDC"
   COMPOSES (IO))
  )

(DEFCLASS IO (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "IO"
   COMPOSES (MAINFRAME))
  )

(DEFCLASS IO-PANEL-MODULES (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "IO panel"
   COMPOSES (2STAGE-IO))
  )

(DEFCLASS ISU (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "ISU memory"
   COMPOSES (MAINFRAME))
  )

(DEFCLASS ITB-MODULES (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "ITB"
   COMPOSES (2STAGE-IO))
  )
(DEFCLASS MAINFRAME (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "mainframe"
   COMPOSES (SYSTEM))
  )

(DEFCLASS MEMORY (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "main memory"
   COMPOSES (MAINFRAME))
  )

(DEFCLASS MISC (BIN)
```

```
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "miscellaneous"
   COMPOSES (REST))
  )

(DEFCLASS MULTIPROCESSING (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "multi-processing"
   COMPOSES (SYSTEM))
  )

(DEFCLASS PLENUM (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "plenum"
   COMPOSES (MAINFRAME))
  )

(DEFCLASS POWER-SUPPLY (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "power supply"
   COMPOSES (MAINFRAME))
  )

(DEFCLASS PROCESSOR (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "processor"
   COMPOSES (MAINFRAME))
  )

(DEFCLASS PRODUCT.LINE (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "M1234 product line"
   PROD.GEN (PRODUCT.LINE))
  )

(DEFCLASS REST (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "remaining"
   COMPOSES (SYSTEM))
  )

(DEFCLASS SP (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   DESCR "SP"
   COMPOSES (MAINFRAME))
  )
```

```
(DEFCLASS M1234.SYSTEM (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   PROD.GEN (SYSTEM))
  )

(* XTEST-TBS definition)
(DEFINSTANCE XTEST-TBS TBS _
  SET.OF XTESTKB
  FILED.ON XTESTFH
  ELEMENTS          (2STAGE-IO.TB CABLE.TB CLC/MLA.TB
                     CONSOLE.TB FICS.TB FIRMWARE.TB ICS.TB
                    IDC.TB IO-PANEL-MODULES.TB IO.TB ISU.TB
                    ITB-MODULES.TB MAINFRAME.TB MEMORY.TB
                    MISC.TB MULTIPROCESSING.TB PLENUM.TB
                    POWER-SUPPLY.TB PROCESSOR.TB
                    PRODUCT.LINE.TB REST.TB SP.TB SYSTEM.TB))
```

APPENDIX II (B)

KNOWLEDGE BASE FOR M1234 COMPUTER
TASK BLOCKS

```
(DEFTASKBLOCK 2STAGE-IO.TB 2STAGE-IO ()
                                    (* jsb: " 4-Aug-83 23:17")
  (CONFIGURE 'ITB-MODULES (FIND 'ALL 'CCS '(IMPLEMENTS ITB-
MODULES)))
  (CONFIGURE 'IO-PANEL-MODULES (FIND 'ALL 'CCS '(IMPLEMENTS
IO-PANEL-MODULES)))
  (CHECK 'IO-CAPACITY))

(DEFTASKBLOCK CABLE.TB CABLE ()    (* jsb: " 4-Aug-83 23:13")
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK CLC/MLA.TB CLC/MLA ()
                                    (* jsb: " 7-AUG-83 20:32")
  (CONFIGURE 'FICS (FIND 'ALL 'CCS '(IMPLEMENTS FICS)))
  (CONFIGURE 'ICS (FIND 'ALL 'CCS '(IMPLEMENTS ICS)))
  (DETERMINE '(#CLC.MLA.LINES ICS.PIBS))
  (CHECK 'IO-CAPACITY))

(DEFTASKBLOCK CONSOLE.TB CONSOLE ()
                                    (* jsb: " 4-Aug-83 23:13")
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK FICS.TB FICS ()      (* jsb: " 4-Aug-83 23:16")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK FIRMWARE.TB FIRMWARE ()
                                    (* jsb: " 7-AUG-83 17:24")
  (CHECK 'REQUIREMENTS)
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT)))
```

```
(DEFTASKBLOCK ICS.TB ICS ()          (* jsb: " 4-Aug-83 23:17")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK IDC.TB IDC ()          (* jsb: " 4-Aug-83 23:14")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK IO-PANEL-MODULES.TB IO-PANEL-MODULES ()
                                     (* jsb: " 7-AUG-83 20:30")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT))
  (DETERMINE '(#CT.LINES #IOLC.LINES)))
(DEFTASKBLOCK IO.TB IO ()            (* jsb: " 4-Aug-83 23:12")
  (CONFIGURE 'IDC (FIND 'ALL 'CCS '(IMPLEMENTS IDC)))
  (CONFIGURE 'CLC/MLA (FIND 'ALL 'CCS '(IMPLEMENTS
CLC/MLA)))
  (CONFIGURE '2STAGE-IO (FIND 'ALL 'CCS '(IMPLEMENTS 2STAGE-
IO))))

(DEFTASKBLOCK ISU.TB ISU ()          (* jst: "11-AUG-83 08:49")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT))
  (DETERMINE '(ISU-MEMORY.SUPPLIED))
  (CHECK 'REQUIREMENTS))

(DEFTASKBLOCK ITB-MODULES.TB ITB-MODULES ()
                                     (* jsb: " 7-AUG-83 20:29")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT))
  (DETERMINE '(#TRUNK.LINES IO.PIBS)))

(DEFTASKBLOCK MAINFRAME.TB MAINFRAME ()
                                     (* jsb: " 4-Aug-83 23:07")
  (CONFIGURE 'PROCESSOR (FIND 'ALL 'CCS '(IMPLEMENTS
PROCESSOR)))
  (CONFIGURE 'MEMORY (FIND 'ALL 'CCS '(IMPLEMENTS MEMORY)))
  (CONFIGURE 'ISU (FIND 'ALL 'CCS '(IMPLEMENTS ISU)))
  (CONFIGURE 'FIRMWARE (FIND 'ALL 'CCS '(IMPLEMENTS
FIRMWARE)))
  (CONFIGURE 'IO (FIND 'ALL 'CCS '(IMPLEMENTS IO)))
  (CONFIGURE 'PLENUM (FIND 'ALL 'CCS '(IMPLEMENTS PLENUM)))
  (CONFIGURE 'POWER-SUPPLY (FIND 'ALL 'CCS '(IMPLEMENTS
POWER-SUPPLY))))

(DEFTASKBLOCK MEMORY.TB MEMORY ()
                                     (* jsb: " 7-AUG-83 20:28")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT))
  (DETERMINE '(MAIN-MEMORY.SUPPLIED MEMORY.INTERLEAVING
MEMORY.PIBS))
  (CHECK 'INTERLEAVING)
  (CHECK 'MEMORY-CAPACITY)
  (CHECK 'MEMORY-IEMENT))
```

```
(DEFTASKBLOCK MISC.TB MISC ()      (* jsb: " 9-AUG-83 14:45")
  (EXPAND* '(AK FT KT))
  (CHECK 'REQUIREMENTS)
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK MULTIPROCESSING.TB MULTIPROCESSING ()
                                   (* jsb: " 7-AUG-83 20:31")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT))
  (DETERMINE '(MP.PIBS)))
(DEFTASKBLOCK PLENUM.TB PLENUM ()
                                   (* jsb: " 4-Aug-83 23:13")
  (CHECK 'CONTENTS)
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK POWER-SUPPLY.TB POWER-SUPPLY ()
                                   (* jsb: " 7-AUG-83 17:24")
  (CHECK 'REQUIREMENTS)
  (EXPAND* '(AK_FT KT)))

(DEFTASKBLOCK PROCESSOR.TB PROCESSOR ()
                                   (* jsb: " 4-Aug-83 23:08")
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK PRODUCT.LINE.TB PRODUCT.LINE ()
                                   (* jsb: " 3-AUG-83 18:17")
  (BVSET 'SYSTEM.TYPE (COMPUTE.SYSTEM.TYPE))
  (CONFIGURE (FETCH 'SYSTEM.TYPE)
         (FIND 'ALL 'CCS)))

(DEFTASKBLOCK REST.TB REST ()      (* jsb: " 4-Aug-83 23:14")
  (CONFIGURE 'CABLE (FIND 'ALL 'CCS '(IMPLEMENTS CABLE)))
  (CONFIGURE 'CONSOLE (FIND 'ALL 'CCS '(IMPLEMENTS
CONSOLE)))
  (CONFIGURE 'MISC (FIND 'ALL 'CCS '(IMPLEMENTS MISC))))

(DEFTASKBLOCK SP.TB SP ()          (* jsb: " 4-Aug-83 23:08")
  (EXPAND* '(AK FT KT)))

(DEFTASKBLOCK SYSTEM.TB SYSTEM ()
                                   (* jst: "11-AUG-83 09:11")
  (EXPAND* '(BG BU.V85))
  (DETERMINE '(CLASS.MODEL.EXPANSION BASIC.TO.PRICED
MP.ORDER))
  (CONFIGURE 'MULTIPROCESSING (FIND 'ALL 'CCS '(IMPLEMENTS
MULTIPROCESSING)))
  (CONFIGURE 'MAINFRAME (FIND 'ALL 'CCS '(IMPLEMENTS
MAINFRAME)))
  (CONFIGURE 'REST (FIND 'ALL 'CCS '(IMPLEMENTS REST)))
  (CHECK 'POWER)
  (CHECK 'PIBS)
  (EXPAND* '(AK FT KT))
  (CHECK 'EFFECTIVITY)
  (CHECK 'PHASE-OUT))
```

APPENDIX II (C)

KNOWLEDGE BASE FOR M1234 COMPUTER PARTS CATALOG

```
(* XTEST-BSS definition)
(DEFINSTANCE XTEST-BSS BSS
  SET.OF XTESTKB
  FILED.ON XTESTBX
  ELEMENTS (M1234))

(DEFCLASS M1234 (PRODUCT.ID)
  ()
  (METACLASS BS
   DESCR "The M1234 product line"
   MFG.PLANT 1804
   IMPLEMENTS.FN PRODUCT.LINE
   BASIC.GROUPS NIL)
  )

(* XTEST-BGS definition)
(DEFINSTANCE XTEST-BGS BGS
  SET.OF XTESTKB
  FILED.ON XTESTBX
  ELEMENTS (M1234-II))

(DEFCLASS M1234-II (PRODUCT.ID)
  ()
  (METACLASS BG
   DESCR "M1234-II SYSTEM "
   IMPLEMENTS.FN M1234.SYSTEM
   BASIC.UNITS NIL)
  )

(* XTEST-BUS definition)
(DEFINSTANCE XTEST-BUS BUS
  SET.OF XTESTKB
  FILED.ON XTESTBX
  ELEMENTS (1234-0066-0690 1234-3003-0690))

(DEFCLASS 1234-0066-0690 (PRODUCT.ID)
  ()
  (METACLASS BU
   DESCR "Vanilla model for M1234 system"
   MFG.PLANT 1804
   IMPLEMENTS.FN SYSTEM
   ACCESSORIES NIL
   REQUIRED.IVARS NIL)
  )
(DEFCLASS 1234-3003-0690 (PRODUCT.ID)
  ()
  (METACLASS BU.M12
   DESCR "Class model for M1234 system"
```

```
    IMPLEMENTS.FN M1234.SYSTEM
    BASIC.TO.PRICED ((1 . 1234-K416)
                     (1 . 1234-K920)
                     (2 . 1234-K924)
                     (1 . 1234-K940)
                     (1 . 1234-K941))
    VANILLA.MODEL 1234-0066-0690)
   )

(* XTEST-AKS definition)
(DEFINSTANCE XTEST-AKS AKS
  SET.OF XTESTKB
  FILED.ON XTESTAK
  ELEMENTS (1234-P413 1234-P723 1234-P724 1234-P742 1234-
            P743 1234-P954 1234-P955))

(DEFCLASS 1234-P413 (PRODUCT.ID)
  ()
  (METACLASS AK
   MFG.PLANT 1804
   DESCR "Additional Power Supply"
   IMPLEMENTS.FN POWER-SUPPLY)
  )

(DEFCLASS 1234-P723 (PRODUCT.ID)
  ()
  (METACLASS AK
   DESCR "1024K TO 2048K MEM ADD (CONT  TWO 1234-K920, FOUR
                                  1234-K924)"
   MFG.PLANT 1804
   IMPLEMENTS.FN MEMORY)
  )

(DEFCLASS 1234-P724 (PRODUCT.ID)
  ()
  (METACLASS AK
   DESCR "2048K TO 3072 MEMORY ADDITION (CONTAINS FOUR 1234-
                                         K924)"
   MFG.PLANT 1804
   IMPLEMENTS.FN MEMORY)
  )

(DEFCLASS 1234-P742 (PRODUCT.ID)
  ()
  (METACLASS AK
   DESCR "VHST (INCL. 1234-K930 & K931) "
   MFG.PLANT 1804
   IMPLEMENTS.FN 2STAGE-IO)
  )

(DEFCLASS 1234-P743 (PRODUCT.ID)
  ()
  (METACLASS AK
   DESCR "I/O LINK CONTROL     "
   MFG.PLANT 1804
```

```
   COMMENT (NOTE CRIT-KITS-0006-NOTE-09)
   IMPLEMENTS.FN 2STAGE-IO)
  )

(DEFCLASS 1234-P954 (PRODUCT.ID)
  ()
  (METACLASS AK
   DESCR "FIRST CLC/MLA COMB (CONTAINS  1234-K905, K901 AND
                              K903)"
   MFG.PLANT 1804
   IMPLEMENTS.FN CLC/MLA)
  )

(DEFCLASS 1234-P955 (PRODUCT.ID)
  ()
  (METACLASS AK
   DESCR "SECOND CLC/MLA COMB (CONTAINS 1234-K905, K901 AND
                               K903)"
   MFG.PLANT 1804
   IMPLEMENTS.FN CLC/MLA)
  )

(* XTEST-FTS definition)
(DEFINSTANCE XTEST-FTS FTS
  SET.OF XTESTKB
  FILED.ON XTESTFT
  ELEMENTS     (1234-F010 1234-F113 1234-F262 1234-F290 1234-
               F360 1234-F361 1234-F362 1234-F400 1234-F413
               1234-F504 1234-F510 1234-F545 1234-F680 1234-
               F690 1234-F691 1234-K010))

(DEFCLASS 1234-F010 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "Firmware"
   IMPLEMENTS.FN FIRMWARE
   UNIQUE.EXPANSION 1234-K010)
  )
(DEFCLASS 1234-F113 (PRODUCT.ID)
  ()
  (METACLASS FT
   IMPLEMENTS.FN ISU
   DESCR "ISU 113K"
   PIBS (0 0)
   POWER (0.0 8.0 33.0))
  )

(DEFCLASS 1234-F262 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "Memory 1M, 2 way"
   IMPLEMENTS.FN MEMORY
   POWER (6.0 0.0 34.0)
   PIBS (4 0))
  )
```

```
(DEFCLASS 1234-F290 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "Memory 2M, 2 way"
   IMPLEMENTS.FN MEMORY
   POWER (44.0 0.0 12.0)
   PIBS (8 0))
  )

(DEFCLASS 1234-F360 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "MFG.PLANT COMMON TRUNK  "
   IMPLEMENTS.FN ITB-MODULES
   POWER (0.0 8.6 13.5)
   PIBS (2 2))
  )

(DEFCLASS 1234-F361 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "Common trunk"
   IMPLEMENTS.FN ITB-MODULES
   POWER (0.0 17.2 27.0))
  )

(DEFCLASS 1234-F362 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "Common trunk"
   IMPLEMENTS.FN ITB-MODULES
   POWER (0.0 25.8 40.5))
  )

(DEFCLASS 1234-F400 (PRODUCT.ID)
  ()
  (METACLASS FT
   IMPLEMENTS.FN POWER-SUPPLY
   MFG.PLANT 1804
   DESCR "-5V power supply")
  )

(DEFCLASS 1234-F413 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "MFG.PLANT POWER SUPPLY  "
   IMPLEMENTS.FN POWER-SUPPLY
   POWER (0 0 -62)
   PIBS (0 0))
  )

(DEFCLASS 1234-F504 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "MFG.PLANT FIRST CLC/MLA "
```

```
    IMPLEMENTS.FN CLC/MLA
    POWER (2 1 5.5)
    PIBS (2 2))
   )

(DEFCLASS 1234-F510 (PRODUCT.ID)
   ()
   (METACLASS FT
    DESCR "Fast ICS, 2CLC, 2MLA"
    IMPLEMENTS.FN FICS
    POWER (11.0 2.0 4.0)
    PIBS (4 4))
   )

(DEFCLASS 1234-F545 (PRODUCT.ID)
   ()
   (METACLASS FT
    DESCR "MFG.PLANT ICS LIGHT DISPLAY  "
    IMPLEMENTS.FN MISC)
   )

(DEFCLASS 1234-F680 (PRODUCT.ID)
   ()
   (METACLASS FT
    DESCR "MFG.PLANT BYTE MULTIPLEX TRUNK    "
    IMPLEMENTS.FN IO-PANEL-MODULES
    POWER (7 .4 .5)
    PIBS (0 0))
   )

(DEFCLASS 1234-F690 (PRODUCT.ID)
   ()
   (METACLASS FT
    DESCR "MFG.PLANT BSLA"
    IMPLEMENTS.FN IO-PANEL-MODULES
    POWER (7.8 .4 .5)
    PIBS (0 0))
   )

(DEFCLASS 1234-F691 (PRODUCT.ID)
   ()
   (METACLASS FT
    DESCR "BSLA"
    IMPLEMENTS.FN IO-PANEL-MODULES
    POWER (15.6 .8 1.0))
   )

(DEFCLASS 1234-K010 (PRODUCT.ID)
   ()
   (METACLASS FT
    DESCR "Firmware"
    IMPLEMENTS.FN FIRMWARE)
   )
```

```
(* XTEST-KTS definition)
(DEFINSTANCE XTEST-KTS KTS
  SET.OF XTESTKB
  FILED.ON XTESTKT
  ELEMENTS    (1234-K010 1234-K400 1234-K413 1234-K416 1234-
              K450 1234-K901 1234-K903 1234-K905 1234-K913
              1234-K915 1234-K917 1234-K920 1234-K921 1234-
              K924 1234-K930 1234-K931 1234-K935 1234-K936
              1234-K937 1234-K939 1234-K940 1234-K941))

(DEFCLASS 1234-K010 (PRODUCT.ID)
  ()
  (METACLASS FT
   DESCR "Firmware"
   IMPLEMENTS.FN FIRMWARE)
  )

(DEFCLASS 1234-K400 (PRODUCT.ID)
  ()
  (METACLASS KT
   IMPLEMENTS.FN POWER-SUPPLY
   MFG.PLANT 1804
   DESCR "-5V power supply")
  )
(DEFCLASS 1234-K413 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "+5V ADD. POWER SUPPLY    "
   MFG.PLANT 1804
   IMPLEMENTS.FN POWER-SUPPLY)
  )

(DEFCLASS 1234-K416 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "CONSOLE INDICATOR INTERFACE   "
   MFG.PLANT 1804
   IMPLEMENTS.FN CONSOLE)
  )

(DEFCLASS 1234-K450 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "VS HARDWARE    "
   MFG.PLANT 1804
   IMPLEMENTS.FN MAINFRAME)
  )

(DEFCLASS 1234-K901 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "MULTILINE ADAPTER   "
   MFG.PLANT 1804
   IMPLEMENTS.FN CLC/MLA)
  )
```

```
(DEFCLASS 1234-K903 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "CLC EXPANSION MODULE"
   MFG.PLANT 1804
   IMPLEMENTS.FN CLC/MLA)
  )

(DEFCLASS 1234-K905 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "FAST CLC  "
   MFG.PLANT 1804
   IMPLEMENTS.FN CLC/MLA)
  )
(DEFCLASS 1234-K913 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "ISU, 8K FAST   "
   MFG.PLANT 1804
   IMPLEMENTS.FN ISU)
  )

(DEFCLASS 1234-K915 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "?? Mystery kit ?? "
   MFG.PLANT 1804
   IMPLEMENTS.FN ISU)
  )

(DEFCLASS 1234-K917 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "SERVICE PROCESSOR, ISU NO.1   "
   MFG.PLANT 1804
   IMPLEMENTS.FN ISU)
  )

(DEFCLASS 1234-K920 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "MSU EXPANSION MODULE"
   MFG.PLANT 1804
   IMPLEMENTS.FN MEMORY)
  )

(DEFCLASS 1234-K921 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "MSU, INTERFACE "
   MFG.PLANT 1804
   IMPLEMENTS.FN MEMORY)
  )
```

```
(DEFCLASS 1234-K924 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "MSU 256K ARRAY "
   MFG.PLANT 1804
   IMPLEMENTS.FN MEMORY)
  )
(DEFCLASS 1234-K930 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "COMMON TRUNK DATA INTERFACE   "
   MFG.PLANT 1804
   IMPLEMENTS.FN IO-PANEL-MODULES)
  )

(DEFCLASS 1234-K931 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "DIRECT MEMORY ACCESS TRUNK    "
   MFG.PLANT 1804
   IMPLEMENTS.FN ITB-MODULES)
  )

(DEFCLASS 1234-K935 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "BIT SERIAL LINK CABLES   "
   MFG.PLANT 1804
   IMPLEMENTS.FN IO-PANEL-MODULES)
  )

(DEFCLASS 1234-K936 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "CONNECTOR MODULE    "
   MFG.PLANT 1804
   IMPLEMENTS.FN IO-PANEL-MODULES)
  )

(DEFCLASS 1234-K937 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "CT BSL ADAPTER "
   MFG.PLANT 1804
   IMPLEMENTS.FN IO-PANEL-MODULES)
  )

(DEFCLASS 1234-K939 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "BSL ENTRY MODULE    "
   MFG.PLANT 1804
   IMPLEMENTS.FN IO-PANEL-MODULES)
  )
```

```
(DEFCLASS 1234-K940 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "ISU/MSU RESISTOR     "
   MFG.PLANT 1804
   IMPLEMENTS.FN MEMORY)
  )

(DEFCLASS 1234-K941 (PRODUCT.ID)
  ()
  (METACLASS KT
   DESCR "CRYSTAL 84/56 NS     "
   MFG.PLANT 1804
   IMPLEMENTS.FN PROCESSOR)
  )

(* XTEST-KVS definition)
(DEFINSTANCE XTEST-KVS KVS
  SET.OF XTESTKB
  FILED.ON XTESTKV
  ELEMENTS (1234-K400-V001))

(DEFCLASS 1234-K400-V001 (PRODUCT.ID)
  ()
  (METACLASS KV
   IMPLEMENTS.FN POWER-SUPPLY
   MFG.PLANT 1804
   DESCR "-5V power supply")
  )
```

APPENDIX II (D)

KNOWLEDGE BASE FOR M1234 COMPUTER
CONSTRAINTS & BIN VARIABLES
KNOWLEDGE BASE FUNCTIONS

```
(* XTEST-CTS definition)
(DEFINSTANCE XTEST-CTS CTS
  SET.OF XTESTKB
  FILED.ON XTESTNCT
  ELEMENTS (AT-LEAST-ONE-FT.CT AT-MOST-ONE-FT.CT AT-MOST-
                ONE-MEMORY-FT.CT NO-FTS.CT NO-MP-
                FTS.CT #CLC.MLA-LINES CLC/MLA-
                DISPLAY CT.IO CT.IOLC.IO
                EXTRA.MINUS5 EXTRA.PLUS5 FIRMWARE.CT
                ICS.PIBS.CT IO.PIBS.CT MINUS5
                MP.FIRMWARE.CT MP.ISU.CT MP.POWER-
                SUPPLY.CT PLUS5 M1234.PIBS.CT1
                M1234.PIBS.CT2
                ))

(DEFCLASS AT-LEAST-ONE-FT.CT (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   BIN.TYPES (ISU FIRMWARE)
```

```
   STATEMENT ((COUNT (FIND 'ALL 'CCS '(ISA FT)))
          IS ZERO)
   MESSAGE ("The customer must order some part that
             implements                    " (SAND CURRENT.BIN
                                                 'GET.TRANSLATION)

                                                   "; this order is
        incorrect.")
   CHECK.KEYWORDS (CONTENTS))
  )
(DEFCLASS AT-MOST-ONE-FT.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (CLC/MLA PLENUM ISU FIRMWARE)
   CHECK.KEYWORDS (CONTENTS)
   STATEMENT ((COUNT (FIND 'ALL 'CCS '(ISA FT)))
              > 1)
   DEL-PARTS (ELIMINATE.BTP (FIND* 'ALL 'CCS '(ISA FT))))
  )
(DEFCLASS AT-MOST-ONE-MEMORY-FT.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   CHECK.KEYWORDS (CONTENTS)
   BIN.TYPES (MEMORY)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (COUNT (FIND 'ALL 'CCS '(ISA FT)))
          > 1)
   DEL-PARTS (ELIMINATE.BTP (FIND* 'ALL 'CCS '(ISA FT))))
  )
(DEFCLASS NO-FTS.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (IDC)
   CHECK.KEYWORDS (CONTENTS)
   STATEMENT ((COUNT (FIND 'ALL 'CCS '(ISA FT)))
          IS NOT ZERO)
   MESSAGE ("The customer is not allowed to order products
             which implement "
               (SAND CURRENT.BIN 'GET.TRANSLATION)
        ". This order is not correct.")
   DEL-PARTS (FIND* 'ALL 'CCS '(ISA FT)))
  )
(DEFCLASS NO-MP-FTS.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   CHECK.KEYWORDS (CONTENTS)
   BIN.TYPES (MULTIPROCESSING)
   STATEMENT ((SYSTEM)
          AND
          (COUNT (FIND 'ALL 'CCS '(ISA FT)))
          IS NOT ZERO)
   DEL-PARTS (FIND* 'ALL 'CCS '(ISA FT)))
  )
```

```
(DEFCLASS #CLC.MLA-LINES (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   BIN.TYPES (CLC/MLA)
   BIN.VARS.REQUIRED (#CLC.MLA.LINES)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (#CLC.MLA.LINES > 20))
   MESSAGE ("The order specifies too many CLC/MLA lines.")
   CHECK.KEYWORDS (IO-CAPACITY))
  )
(DEFCLASS CLC/MLA-DISPLAY (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (MISC)
   BIN.VARS.REQUIRED (#CLC.MLA.LINES)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          ((COUNT.PARTS '1234-K545 T)
           = 1)
          AND
          (BETWEEN #CLC.MLA.LINES 11 20))
   MESSAGE (
"This order uses more than 10 CLC/MLA lines. However, the
customer only ordered 1 Remote ICS display (1234-F545)."
         )
   ADD-PARTS (QUOTE ((1 . 1234-F545)))
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS CT.IO (CTI)
  ()
  (METACLASS WARN-ON-FAILURE.CT
   BIN.VARS.REQUIRED (#TRUNK.LINES #CT.LINES)
   CHECK.KEYWORDS (IO-CAPACITY)
   BIN.TYPES (2STAGE-IO)
   STATEMENT (#CT.LINES <= #TRUNK.LINES)
   MESSAGE ("There are not enough common trunk I/F lines for
the HS, MS, and LS lines ordered.")
)
  )

(DEFCLASS CT.IOLC.IO (CTI)
  ()
  (METACLASS WARN-ON-FAILURE.CT
   BIN.VARS.REQUIRED (#TRUNK.LINES #CT.LINES #IOLC.LINES)
   CHECK.KEYWORDS (IO-CAPACITY)
   BIN.TYPES (2STAGE-IO)
   STATEMENT (#TRUNK.LINES = #IOLC.LINES + #CT.LINES)
   MESSAGE (
"The number of IOLC and common trunk I/F lines does not
match the number of HS, MS, and LS IO lines."))
  )
```

```
(DEFCLASS EXTRA.MINUS5 (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   BIN.TYPES (SYSTEM)
   BIN.VARS.REQUIRED (POWER.USED MP.ORDER)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (NOT MP.ORDER)
          AND
          (ORDERED? '1234-F400 T)
          AND
          (POWER.USED:MINUS.5 <= 250))
   MESSAGE ("The customer ordered a -5V power supply which
is not required to run this system.")
   CHECK.KEYWORDS (POWER))
  )

(DEFCLASS EXTRA.PLUS5 (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   BIN.TYPES (SYSTEM)
   BIN.VARS.REQUIRED (POWER.USED)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (ORDERED? '1234-F413 T)
          AND
          (POWER.USED:PLUS.5 <= 60))
   MESSAGE ("The customer ordered a +5V power supply which
is not required to run this system.")
   CHECK.KEYWORDS (POWER))
  )

(DEFCLASS FIRMWARE.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (FIRMWARE)
   BIN.VARS.REQUIRED (MP.ORDER)
   CHECK.KEYWORDS (REQUIREMENTS)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (NOT MP.ORDER)
          AND
          (NOT (ORDERED? '(1234-F010 1234-K010 1234-F011
1234-K011))))
   MESSAGE (
"No firmware has been ordered for this non-MP system. Please
check with the customer and determine what operating system
will be used with this system. If it is VS-1, they should
order 1 1234-F011; otherwise if it is RS-1, they should
order 1 1234-F010. Please select one of these features for
this order.")
   SET-PARTS (CHOOSE.ONE '((1. 1234-F010) (1 . 1234-F011))))
  )
```

```
(DEFCLASS ICS.PIBS.CT (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   BIN.TYPES (SYSTEM)
   BIN.VARS.REQUIRED (ICS.PIBS)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (ICS.PIBS > 8))
   MESSAGE ("The order uses more ICS PIBS than is allowed
for this system.")
   CHECK.KEYWORDS (PIBS))
  )

(DEFCLASS IO.PIBS.CT (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   BIN.TYPES (SYSTEM)
   BIN.VARS.REQUIRED (IO.PIBS ICS.PIBS)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (IO.PIBS + ICS.PIBS > 12))
   MESSAGE ("The order uses more IO and ICS PIBS combined
than is allowed for this system.")
   CHECK.KEYWORDS (PIBS))
  )

(DEFCLASS MINUS5 (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (SYSTEM)
   BIN.VARS.REQUIRED (POWER.SUPPLIED POWER.USED)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (POWER.USED:MINUS.5 - POWER.SUPPLIED:MINUS.5)
          > 250)
   MESSAGE ("The order uses more -5V power than is
supplied.")
   ADD-PARTS (QUOTE ((1 . 1234-F400)))
   CHECK.KEYWORDS (POWER))
  )
(DEFCLASS MP.FIRMWARE.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (FIRMWARE)
   BIN.VARS.REQUIRED (MP.ORDER)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND MP.ORDER AND (NOT (ORDERED? '5640-F400)))
   MESSAGE ("All MP orders require MP firmware.")
   SET-PARTS (QUOTE ((1 . 5640-F400)))
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS MP.ISU.CT (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
```

```
   BIN.TYPES (ISU)
   CHECK.KEYWORDS (REQUIREMENTS)
   STATEMENT ((SYSTEM M1234.SYSTEM )
          AND MP.ORDER AND (ISU-MEMORY.SUPPLIED < 40))
   BIN.VARS.REQUIRED (MP.ORDER ISU-MEMORY.SUPPLIED)
   MESSAGE (
"This MP order requires at least 40KB of ISU memory.
If this system will be running VS2 V03, please select 1234-
F122 or F132; otherwise, if it will be running VS2 V04,
please select 1234-F123 or F128."))
  )

(DEFCLASS MP.POWER-SUPPLY.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (POWER-SUPPLY)
   CHECK.KEYWORDS (REQUIREMENTS)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND MP.ORDER AND (NOT (ORDERED? '(1234-F400 1234-
            K400))))
   BIN.VARS.REQUIRED (MP.ORDER)
   ADD-PARTS (QUOTE ((1 . 1234-F400)))
   MESSAGE ("This MP order requires an additional +5V power
supply."))
  )

(DEFCLASS PLUS5 (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (SYSTEM)
   BIN.VARS.REQUIRED (POWER.SUPPLIED POWER.USED)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (POWER.USED:PLUS.5 - POWER.SUPPLIED:PLUS.5)
          > 60)
   MESSAGE ("The order uses more +5V power than is
supplied.")
   ADD-PARTS (QUOTE ((1 . 1234-F413)))
   CHECK.KEYWORDS (POWER))
  )

(DEFCLASS M1234.PIBS.CT1 (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   BIN.TYPES (SYSTEM)
   BIN.VARS.REQUIRED (TOTAL.PIBS)
   CHECK.KEYWORDS (PIBS)
   STATEMENT ((SYSTEM M1234.SYSTEM)
          AND
          (ORDERED? '1234-K926)
          AND
          (TOTAL.PIBS > 38))
   MESSAGE (
"The order includes a LH plenum and uses more than 38 PIBS;
this is not a legal order."))
  )
```

```
(DEFCLASS M1234.PIBS.CT2 (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   BIN.TYPES (SYSTEM)
   BIN.VARS.REQUIRED (TOTAL.PIBS)
   CHECK.KEYWORDS (PIBS)
   STATEMENT ((SYSTEM M1234.SYSTEM)
              AND
              (NOT (ORDERED? '1234-K926))
              AND
              (TOTAL.PIBS > 38))
   MESSAGE ("The order uses more than 38 PIBS; must add LH
plenum.")
   ADD-PARTS (QUOTE ((1 . 1234-F441))))
  )
(* XTEST-NCTS definition)
(DEFINSTANCE XTEST-NCTS NCTS
  SET.OF XTESTKB
  FILED.ON XTESTNCT
  ELEMENTS (1024-INCREMENT 2-WAY-INTERLEAVING CT-LINES IOLC-
                LINES))
(DEFCLASS 1024-INCREMENT (CTI)
  ()
  (METACLASS MEMORY-INCREMENT.CT
   BIN.TYPES (MEMORY)
   STATEMENT ((SYSTEM M1234.SYSTEM)
              AND
              (NOT (INCREMENT MAIN-MEMORY.SUPPLIED 1024))))
  )
(DEFCLASS 2-WAY-INTERLEAVING (CTI)
  ()
  (METACLASS INTERLEAVING.CT
   BIN.TYPES (MEMORY)
   STATEMENT ((SYSTEM M1234.SYSTEM)
              AND
              (NOT (MEMORY.INTERLEAVING = '2-WAY))))
  )
(DEFCLASS CT-LINES (CTI)
  ()
  (METACLASS IO-LINES.CT
   BIN.TYPES (2STAGE-IO)
   STATEMENT ((SYSTEM M1234.SYSTEM)
              AND
              (NOT (BETWEEN #CT.LINES 0 6))))
  )
(DEFCLASS IOLC-LINES (CTI)
  ()
  (METACLASS IO-LINES.CT
   BIN.TYPES (2STAGE-IO)
   STATEMENT ((SYSTEM M1234.SYSTEM )
              AND
              (NOT (BETWEEN #IOLC.LINES 0 6))))
  )
```

```
(* XTEST-BVS definition)
(DEFINSTANCE XTEST-BVS BVS
  SET.OF XTESTKB
  FILED.ON XTESTNCT
  ELEMENTS (#CLC #CLC.MLA.LINES #CT.LINES #IOLC.LINES #MLA
                #TRUNK.LINES 59.POSITION.BACKPANEL
                BASIC.TO.PRICED CAP.PROCESSOR
                CLASS.MODEL.EXPANSION ICS.PIBS IO.PIBS
                ISU-MEMORY.SUPPLIED MAIN-MEMORY.SUPPLIED
                MEMORY.INTERLEAVING MEMORY.PIBS
                MP.ORDER MP.PIBS OPERATING.SYSTEM
                POWER.SUPPLIED POWER.USED TOTAL.PIBS))

(DEFCLASS #CLC (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the number of CLC's"
   HOW.TO.DETERMINE (COMPUTE.#CLC)
   LEGAL.VALUES INTEGER)
  )

(DEFCLASS #CLC.MLA.LINES (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the number of CLC/MLA lines for this order"
   HOW.TO.DETERMINE (COMPUTE.#CLC.MLA.LINES)
   LEGAL.VALUES (AND INTEGER (SATISFIES (LAMBDA (value)
                                (BETWEEN value 0 20)))))
  )

(DEFCLASS #CT.LINES (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the number of common trunk interface lines"
   HOW.TO.DETERMINE (COMPUTE.#CT.LINES)
   LEGAL.VALUES (AND INTEGER (SATISFIES (LAMBDA (value)
                                (BETWEEN value 0 8)))))
  )

(DEFCLASS #IOLC.LINES (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the number of IOLC interface lines"
   HOW.TO.DETERMINE (COMPUTE.#IOLC.LINES)
   LEGAL.VALUES (AND INTEGER (SATISFIES (LAMBDA (value)
                                (BETWEEN value 0 8)))))
  )

(DEFCLASS #MLA (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
```

```
   DESCR "the number of MLA's"
   HOW.TO.DETERMINE (COMPUTE.#MLA)
   LEGAL.VALUES INTEGER)
  )

(DEFCLASS #TRUNK.LINES (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the number of standard IO lines for this order"
   HOW.TO.DETERMINE (COMPUTE.#TRUNK.LINES)
   LEGAL.VALUES (AND INTEGER (SATISFIES (LAMBDA (value)
                                        (BETWEEN value 0 8)))))
  )

(DEFCLASS 59.POSITION.BACKPANEL (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "this order will use a 59-position backpanel (T/N >
        1762)"
   HOW.TO.DETERMINE (COMPUTE.59.POSITION.BACKPANEL)
   LEGAL.VALUES BOOLEAN)
  )


(DEFCLASS BASIC.TO.PRICED (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the basic-to-priced parts"
   HOW.TO.DETERMINE (COMPUTE.BASIC.TO.PRICED)
   LEGAL.VALUES PRODUCT.QTYS)
  )


(DEFCLASS CAP.PROCESSOR (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "this is a CAP processor"
   HOW.TO.DETERMINE (ASK.BOOLEAN.QUESTION)
   LEGAL.VALUES BOOLEAN)
  )


(DEFCLASS CLASS.MODEL.EXPANSION (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the class model parts"
   HOW.TO.DETERMINE (COMPUTE.CLASS.MODEL.EXPANSION)
   LEGAL.VALUES PRODUCT.QTYS)
  )
```

```
(DEFCLASS ICS.PIBS (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "PIBS required for CLC/MLA"
   HOW.TO.DETERMINE (COMPUTE.ICS.PIBS)
   LEGAL.VALUES INTEGER)
  )

(DEFCLASS IO.PIBS (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "PIBS required for IOLC and common trunk"
   HOW.TO.DETERMINE (COMPUTE.IO.PIBS)
   LEGAL.VALUES INTEGER)
  )


(DEFCLASS ISU-MEMORY.SUPPLIED (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the ISU memory supplied with this order (in KB)"
   HOW.TO.DETERMINE (COMPUTE.ISU-MEMORY.SUPPLIED)
   LEGAL.VALUES INTEGER)
  )

(DEFCLASS MAIN-MEMORY.SUPPLIED (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the main memory supplied with this order (in KB)"
   HOW.TO.DETERMINE (COMPUTE.MAIN-MEMORY.SUPPLIED)
   LEGAL.VALUES INTEGER)
  )

(DEFCLASS MEMORY.INTERLEAVING (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the memory interleaving for this order"
   HOW.TO.DETERMINE (COMPUTE.MEMORY.INTERLEAVING)
   LEGAL.VALUES (AND ATOM (MEMBEROF (1-WAY 2-WAY 4-WAY))))
  )

(DEFCLASS MEMORY.PIBS (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "PIBS required for main memory w/o extra plenums"
   HOW.TO.DETERMINE (COMPUTE.MEMORY.PIBS)
   LEGAL.VALUES INTEGER)
  )
```

```
(DEFCLASS MP.ORDER (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "this is a multi-processing order"
   HOW.TO.DETERMINE (COMPUTE.MP.ORDER)
   LEGAL.VALUES BOOLEAN)
  )
(DEFCLASS MP.PIBS (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "PIBS required for MP kits"
   HOW.TO.DETERMINE (COMPUTE.MP.PIBS)
   LEGAL.VALUES INTEGER)
  )

(DEFCLASS OPERATING.SYSTEM (BVI)
  ()
  (METACLASS ENUMERATED.BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the operating system for this order"
   ENUMERATED.SET (RS-1 VS-1 MP))
  )


(DEFCLASS POWER.SUPPLIED (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the power supplied for this order"
   HOW.TO.DETERMINE (COMPUTE.POWER.SUPPLIED)
   LEGAL.VALUES POWER.SPEC)
  )


(DEFCLASS POWER.USED (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the power used by this order"
   HOW.TO.DETERMINE (COMPUTE.POWER.USED)
   LEGAL.VALUES POWER.SPEC)
  )


(DEFCLASS TOTAL.PIBS (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (SYSTEM)
   DESCR "the total PIBS required for this order"
   HOW.TO.DETERMINE (COMPUTE.TOTAL.PIBS)
   LEGAL.VALUES INTEGER)
  )
```

```
(* XTEST-BVMS definition)
(DEFINSTANCE XTEST-BVMS BVMS
  SET.OF XTESTKB
  FILED.ON XTESTNCT
  ELEMENTS (COMPUTE.#CLC COMPUTE.#CLC.MLA.LINES
              COMPUTE.#CT.LINES COMPUTE.#IOLC.LINES
            COMPUTE.#MLA COMPUTE.#TRUNK.LINES
              COMPUTE.59.POSITION.BACKPANEL
            COMPUTE.BASIC.TO.PRICED
              COMPUTE.CLASS.MODEL.EXPANSION
                COMPUTE.ICS.PIBS
            COMPUTE.IO.PIBS COMPUTE.ISU-MEMORY.SUPPLIED
            COMPUTE.MAIN-MEMORY.SUPPLIED
              COMPUTE.MEMORY.INTERLEAVING
            COMPUTE.MEMORY.PIBS COMPUTE.MP.ORDER
              COMPUTE.MP.PIBS
            COMPUTE.POWER.SUPPLIED COMPUTE.POWER.USED
              COMPUTE.TOTAL.PIBS
            COMPUTE.SYSTEM.TYPE ELIMINATE.BTP INCREMENT))

(DEFKBFUN COMPUTE.#CLC ()              (* edited: "10-AUG-83
                                                  20:34")
  ((COUNT.PARTS '1234-K905 T)+(COUNT.PARTS '1234-K900 T)))

(DEFKBFUN COMPUTE.#CLC.MLA.LINES ()  (* edited: "10-AUG-83
                                                  20:41")
  ((FETCH '#CLC)+ 4*(FETCH '#MLA)))

(DEFKBFUN COMPUTE.#CT.LINES ()         (* jst: " 3-AUG-83
                                                  10:57")
  (COUNT.PARTS '1234-K930))

(DEFKBFUN COMPUTE.#IOLC.LINES ()       (* jst: " 3-AUG-83
                                                  10:57")
  (COUNT.PARTS '1234-K935))

(DEFKBFUN COMPUTE.#MLA ()              (* edited: "10-AUG-83
                                                  20:34")
  (COUNT.PARTS '1234-K901 T))

(DEFKBFUN COMPUTE.#TRUNK.LINES ()      (* jst: " 3-AUG-83
                                                  10:57")
  ((COUNT.PARTS '1234-K931)+(COUNT.PARTS '1234-
        K932)+(COUNT.PARTS '1234-K933)))

(DEFKBFUN COMPUTE.59.POSITION.BACKPANEL (BVI)   (* jsb: "31-
                                                  DEC-00 17:14")
  (IF (FETCH 'TRACER.NUMBER T) IS NUMERIC
      THEN (IF (FETCH 'TRACER.NUMBER) > 1762
               THEN T
           ELSE NIL)
    ELSE (ASK.BOOLEAN.QUESTION BVI)))

(DEFKBFUN COMPUTE.BASIC.TO.PRICED ()            (* jsb: "31-
                                                  DEC-00 17:14")
```

```
  (SANDC (FIND 'ONE 'CCS '(ISA BU.M12))
     'GET.BASIC.TO.PRICED))

(DEFKBFUN COMPUTE.CLASS.MODEL.EXPANSION ()      (* edited:
                                                  "11-AUG-83 06:02")
  (PROG (CTS:(LISTOF CONSTRAINT)
      CT:CONSTRAINT)
        (CTS :=(SANDC (FIND 'ONE 'CCS '(ISA BU.M12))
          'GET.EXPANSION.CTS))
        (IF CTS
        THEN (CT :=(SAND (CAR CTS)
         'GET.TO)))
        (RETURN CT)))
(DEFKBFUN COMPUTE.ICS.PIBS ()  (* edited: "10-AUG-83 20:39")
  (PROG (ANS:INTEGER)
        (ANS :=(FETCH '#CLC)+(FETCH '#MLA))
        (IF (IREMAINDER ANS 2)= 1
        THEN (ANS := ANS + 1))
        (RETURN ANS)))

(DEFKBFUN COMPUTE.IO.PIBS ()   (* edited: "11-AUG-83 02:12")
  (2*(FETCH '#TRUNK.LINES)))

(DEFKBFUN COMPUTE.ISU-MEMORY.SUPPLIED ()    (* jst: " 3-AUG-
                                                  83 10:57")
  (2*(COUNT.PARTS '1234-K912)+ 8*(COUNT.PARTS '1234-K913)+
                               32*(COUNT.PARTS '1234-K909)))

(DEFKBFUN COMPUTE.MAIN-MEMORY.SUPPLIED ()   (* jst: " 3-AUG-
                                                  83 10:57")
  (256*(COUNT.PARTS '1234-K924)+ 64*(COUNT.PARTS '1234-
                                                 K922)))

(DEFKBFUN COMPUTE.MEMORY.INTERLEAVING ()     (* edited: "11-
                                                  AUG-83 06:19")
  (SELECTQ (COUNT.PARTS '1234-K921)
       (1 '1-WAY)
       (2 '2-WAY)
       ((4 0)
         '4-WAY)
       (SHOULDNT)))

(DEFKBFUN COMPUTE.MEMORY.PIBS ()  (* jst: "26-JUL-83 02:02")
  (2*(COUNT.PARTS '1234-K924)))

(DEFKBFUN COMPUTE.MP.ORDER ()     (* jst: "11-AUG-83 07:39")
  (IF  (FIND 'ONE 'CCS '(IMPLEMENTS MULTIPROCESSING)) OR
(ORDERED? '5640-F400 T)
      THEN T
    ELSE NIL))

(DEFKBFUN COMPUTE.MP.PIBS ()   (* edited: "11-AUG-83 01:11")
  ((COUNT.PARTS '1234-K917)+(COUNT.PARTS '1212-C003-0091)+
2*(COUNT.PARTS '1401-C058-0090)))
```

```
(DEFKBFUN COMPUTE.POWER.SUPPLIED ()   (* edited: "10-AUG-83
                                                  07:54")
  (PROG (POWER.PART:POWER.SPEC SUM:POWER.SPEC)
        (SUM :=(SEND SUM NEW.SELF))
        (FOR POWER.SUPPLY IN '(1234-F400 1234-F413 1234-
              F396) WHEN (ORDERED? POWER.SUPPLY T)
            DO (SUM _+(SAND POWER.SUPPLY 'GET.POWER)))
        (RETURN SUM:INVERTED)))

(DEFKBFUN COMPUTE.POWER.USED ()   (* jst: "11-AUG-83 07:42")
  (PROG (SUM:POWER.SPEC IO:NUMBER MEM:NUMBER CLC:NUMBER
         MLA:NUMBER FUDGE:NUMBER)
        (SUM :=(SEND SUM NEW.SELF))
        (MEM :=(FETCH 'MAIN-MEMORY.SUPPLIED)/ 256)
        (IO :=(FETCH '#TRUNK.LINES))
        (CLC :=(FETCH '#CLC))
        (MLA :=(FETCH '#MLA))

        (FUDGE :=(SELECTQ (FETCH 'SYSTEM.TYPE)
                          (M1234.SYSTEM 180.0)
                          (SHOULDNT)))
        (SUM:MINUS.5 := FUDGE + 2.0*MEM + 15.0*IO + 2.0*CLC
                        + 0.0*MLA)
        (SUM:PLUS.5 := 6.0 + 1.5*MEM + 7.5*IO + 2.0*CLC +
                       4.0*MLA)
        (RETURN SUM)))

(DEFKBFUN COMPUTE.TOTAL.PIBS ()   (* edited: "11-AUG-83
                                                  05:42")
  (PROG (BASIC:NUMBER MEM:NUMBER PERF:BOOLEAN LH:BOOLEAN
                      RH:BOOLEAN)
        (MEM :=(COUNT.PARTS '1234-K924 T))
        (BASIC :=(FETCH 'MP.PIBS)+(FETCH 'IO.PIBS)+(FETCH
                       'ICS.PIBS))
        (PERF :=(ORDERED? '1234-K941 T))
        (LH :=(ORDERED? '1234-K926 T))
        (RH :=(ORDERED? '1234-K928 T))
        (RETURN (SELECTQ (FETCH 'SYSTEM.TYPE)
                    ((M1234.SYSTEM)
                      (10 + BASIC +(IF LH
                                       THEN 0
                                     ELSE MEM)))
                    (SHOULDNT)))))

(DEFKBFUN COMPUTE.SYSTEM.TYPE ()  (* jsb: " 7-AUG-83 20:04")
  (PROG (SYSTEM:FUNCTION!CLASS)
        (IF (SYSTEM :=(FIND 'ONE 'CCS '(AND (ISA BG)
                                            (IMPLEMENTS SYSTEM))))
          THEN (SYSTEM :=(SANDC SYSTEM 'GET.IMPLEMENTS.FN))
        ELSEIF (SYSTEM :=(FIND 'ONE 'CCS '(ISA BU.M12)))
          THEN (SYSTEM :=(SANDC SYSTEM 'GET.IMPLEMENTS.FN)))
        (IF SYSTEM IS NULL
        THEN (printout T
```

```
"I can't find either a basic group or a class model.
I will configure this order as a generic M1234 system."
                        T)
            (SYSTEM := 'SYSTEM))
    (RETURN SYSTEM)))

(DEFKBFUN ELIMINATE.BTP (CURRENT.CONTENTS:PRODUCT.QTYS)
                                    (* jsb: "26-JAN-84 09:34")
  (PROG (CMPARTS:PRODUCT.QTYS REMOVE.PARTS:PRODUCT.QTYS
              REMAINING.PARTS:PRODUCT.QTYS
          PART:PRODUCT.QTY CMPART:PRODUCT.QTY WORKING.PARTS:
              PRODUCT.QTYS)
        (CMPARTS :=(FETCH 'CLASS.MODEL.EXPANSION T))
        (REMOVE.PARTS :=(FOR PART IN CURRENT.CONTENTS WHEN
              (CMPART :=(SEND CMPARTS MEMBER PART))
       COLLECT (A PRODUCT.QTY WITH NAME = PART:NAME QTY
              =(MAX PART:QTY
              CMPART:QTY))))
        (WORKING.PARTS :=(COPY CURRENT.CONTENTS))
        (REMAINING.PARTS :=(SEND WORKING.PARTS SUBTRACT
              REMOVE.PARTS))
        (PART :=(CAR REMAINING.PARTS))
        (IF (CDR REMAINING.PARTS) IS NULL AND PART:QTY = 1
             THEN (RETURN REMOVE.PARTS)
      ELSE (WORKING.PARTS :=(COPY CURRENT.CONTENTS))
           (RETURN (SEND WORKING.PARTS SUBTRACT (CHOOSE.ONE
              REMAINING.PARTS))))))

(DEFKBFUN INCREMENT (VALUE:INTEGER INCREMENT:INTEGER)
                                    (* jsb: " 3-AUG-83 10:57")
  ((IREMAINDER VALUE INCREMENT)
   IS ZERO))
```

APPENDIX II (E)

KNOWLEDGE BASE FOR M1234 COMPUTER
EXPANSION RULES

```
(* XTEST-ECTS definition)
(DEFINSTANCE XTEST-ECTS ECTS
  SET.OF XTESTKB
  FILED.ON XTESTEX
  ELEMENTS (1234-0066-0690.ECT1 1234-3003-0690.ECT.1 1234-
              F113.ECT.1 1234-F262.E CT.1 1234-F290.ECT.1
              1234-F360.ECT.1 1234-F361.ECT1 1234-F362.ECT1
              1234-F400.ECT 1234-F413.ECT.1 1234-F504.ECT.1
              1234-F510.ECT1 1234-F545.ECT1 1234-F680.ECT.1
              1234-F690.ECT.1 1234-F690.ECT.2 1234-F691.ECT1
              1234-K400.ECT1 1234-P413.ECT 1234-P723.ECT.1
              1234-P724.ECT.1 1234-P742.ECT.1 1234-
              P743.ECT.1 1234-P954.ECT.1 1234-P955.ECT.1
              M1234-II.ECT.1))
```

```
(DEFCLASS 1234-0066-0690.ECT1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-0066-0690
   TO ((3 . 1234-K913-V001)
       (1 . 1234-K920-V002)
       (1 . 1234-K450-V001)
       (1 . 1234-K915-V001)
       (1 . 1234-K917-V001)
       (1 . 1234-K921-V003)
       (2 . 1234-K924-V001)
       (3 . 1234-K940-V001)))
  )

(DEFCLASS 1234-3003-0690.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-3003-0690
   TO ((1 . 1234-F262)
       (1 . 1234-F113)
       (1 . 1234-K915)
       (1 . 1234-K450)
       (1 . 1234-K917)
       (1 . 1234-K416)
       (1 . 1234-K941)))
  )

(DEFCLASS 1234-F113.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F113
   TO ((2 . 1234-K940)
       (3 . 1234-K913)))
  )

(DEFCLASS 1234-F262.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F262
   TO ((2 . 1234-K920)
       (2 . 1234-K921)
       (4 . 1234-K924)
       (2 . 1234-K940)))
  )

(DEFCLASS 1234-F290.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F290
   TO ((4 . 1234-K920)
       (2 . 1234-K921)
       (8 . 1234-K924)
       (2 . 1234-K940)))
  )
```

```
(DEFCLASS 1234-F360.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F360
   TO ((1 . 1234-K931)))
  )

(DEFCLASS 1234-F361.ECT1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F361
   TO ((2 . 1234-K931)))
  )

(DEFCLASS 1234-F362.ECT1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F362
   TO ((3 . 1234-K931)))
  )

(DEFCLASS 1234-F400.ECT (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   TO ((1 . 1234-K400))
   FROM 1234-F400)
  )

(DEFCLASS 1234-F413.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F413
   TO ((1 . 1234-K413)))
  )

(DEFCLASS 1234-F504.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F504
   TO ((1 . 1234-K905)
       (1 . 1234-K901)
       (1 . 1234-K903)))
  )

(DEFCLASS 1234-F510.ECT1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F510
   TO ((2 . 1234-K905)
       (2 . 1234-K901)
       (2 . 1234-K903)))
  )
```

```
(DEFCLASS 1234-F545.ECT1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F545
   TO ((1 . 1234-K545)))
  )

(DEFCLASS 1234-F680.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F680
   TO ((1 . 1234-K930)))
  )

(DEFCLASS 1234-F690.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F690
   TO ((1 . 1234-K935)
       (1 . 1234-K936)
       (1 . 1234-K937)))
  )

(DEFCLASS 1234-F690.ECT.2 (CTI)
  ()
  (METACLASS EXPANSION.CT
   FROM 1234-F690
   TO ((1 . 1234-K935)
       (1 . 1234-K936)
       (1 . 1234-K939))
   BIN.VARS.REQUIRED (TRACER.NUMBER)
   WHEN (TRACER.NUMBER <= 1769))
  )

(DEFCLASS 1234-F691.ECT1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-F691
   TO ((2 . 1234-K935)
       (2 . 1234-K936)
       (2 . 1234-K937)))
  )

(DEFCLASS 1234-K400.ECT1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   TO ((1 . 1234-K400-V001))
   FROM 1234-K400)
  )
```

```
(DEFCLASS 1234-P413.ECT (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-P413
   TO ((1 . 1234-F413)))
  )

(DEFCLASS 1234-P723.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-P723
   TO ((1 . 1234-F290)))
  )

(DEFCLASS 1234-P724.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-P724
   TO ((1 . 1234-F295)))
  )

(DEFCLASS 1234-P742.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-P742
   TO ((1 . 1234-F360)
       (1 . 1234-F680)))
  )

(DEFCLASS 1234-P743.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-P743
   TO ((1 . 1234-F360)
       (1 . 1234-F690)))
  )

(DEFCLASS 1234-P954.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-P954
   TO ((1 . 1234-F504)))
  )

(DEFCLASS 1234-P955.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROM 1234-P955
   TO ((1 . 1234-F504)))
  )
```

```
(DEFCLASS M1234-II.ECT.1 (CTI)
  ()
  (METACLASS ALWAYS.EXPAND.CT
   FROMM1234-II
   TO ((1 . 1234-3003-0690)))
  )
```

APPENDIX III (A)

MINICOMPUTER ORDER CHECKING TYPESCRIPT

```
(MINI) COAST executive> RUN

   Evaluating: (INPUT.ORDER)


Processing new order...
Enter name of order file: T
Reading order information...
Order information>
Reading parts to check...
Order line> 1 MINI-1000
Order line> 4 RAM-1000
Order line> 3 ROM-1000
Order line> 4 TERMINAL-1000
Order line> 3 DISC-1000
Order line>
Reading parts to be shipped F/S...
F/S order line>
Enter order processing options: SHOW-FULL-TRACE
   Adding the following parts: 1 MINI-1000, 4 RAM-1000, 3
   ROM-1000, 4 TERMINAL-1000 and 3 DISC-1000

    Evaluating: (DETERMINE '(TODAYS.DATE ORDERED.PARTS
    ORDERED.PRODUCT.LINE))
    Today's date of original components is set to: 20-Jun-84
    The ordered set of parts of original components is set
    to: MINI-1000, RAM-1000, ROM-1000, TERMINAL-1000 and
    DISC-1000
    The ordered product line of original components is set
    to: minicomputer components

    Evaluating: (IF CHECKING.ORDER
                    THEN   **COMMENT**
                         (CONFIGURE (FETCH
                           'ORDERED.PRODUCT.LINE)
                                    (FIND 'ALL 'CCS))
                         (CHECK 'EMPTY.BIN))


   Configuring minicomputer components
     Initial parts are: 1 MINI-1000, 4 RAM-1000, 3 ROM-1000,
4 TERMINAL-1000 and 3 DISC-1000
```

```
Evaluating: [CONFIGURE 'MAIN-FRAME (FIND 'ALL 'CCS
                '(IMPLEMENTS MAIN-FRAME]

Configuring mainframe components
  Initial parts are: 4 RAM-1000, 3 ROM-1000, 4
    TERMINAL-1000 and 3 DISC-1000

   Evaluating: [CONFIGURE 'CPU (FIND 'ALL 'CCS
                   '(IMPLEMENTS CPU]


   Configuring cpu components
     No initial parts

      Evaluating: (CHECK 'REQUIREMENTS)
      Checking REQUIREMENTS constraints in cpu
         components
        The following constraints apply: AT-MOST-
            ONE.CT NO-PARTS.CT
      Checking constraint: AT-MOST-ONE.CT (a MODIFY-
            ON-SUCCESS.CT)

        The statement: ((COUNT CONTENTS)
                        > 1)
        failed with value NIL when applied to cpu
            components
      Checking constraint: NO-PARTS.CT (a MODIFY-ON-
            SUCCESS.CT)

        The statement: ((COUNT CONTENTS)
                        IS ZERO)
        succeeded with value T when applied to cpu
            components
Adding required component to CPU.1
Adding 1 CPU-1000 to cpu components ; OK? YES
       Adding the following parts: 1 CPU-1000

   Evaluating: [CONFIGURE 'MEMORY (FIND 'ALL 'CCS
                   '(IMPLEMENTS MEMORY]


   Configuring memory components
     Initial parts are: 4 RAM-1000 and 3 ROM-1000

      Evaluating: [CONFIGURE 'RAM (FIND 'ALL 'CCS
                      '(IMPLEMENTS RAM]


      Configuring random access memory components
        Initial parts are: 4 RAM-1000

         Evaluating: (CHECK 'REQUIREMENTS)
         Checking REQUIREMENTS constraints in random
            access memory components
           The following constraints apply: NO-
               PARTS.CT
```

```
      Checking constraint: NO-PARTS.CT (a MODIFY-
            ON-SUCCESS.CT)

        The statement: ((COUNT CONTENTS)
                        IS ZERO)
        failed with value NIL when applied to
          random access memory components.

    Evaluating: [CONFIGURE 'ROM (FIND 'ALL 'CCS
                    '(IMPLEMENTS ROM]


    Configuring read-only memory components
      Initial parts are: 3 ROM-1000

       Evaluating: (CHECK 'REQUIREMENTS)
       Checking REQUIREMENTS constraints in read-
           only memory components
         The following constraints apply: NO-
            PARTS.CT
       Checking constraint: NO-PARTS.CT (a MODIFY-
           ON-SUCCESS.CT)

         The statement: ((COUNT CONTENTS)
                         IS ZERO)
         failed with value NIL when applied to read-
           only memory components

    Evaluating: (CHECK 'REQUIREMENTS)
    Checking REQUIREMENTS constraints in memory
        components
      The following constraints apply: MEMORY.CT1
    Checking constraint: MEMORY.CT1 (a WARN-ON-
        SUCCESS.CT)
    The number of ram boards ordered of minicomputer
        components is set to: 4
    The number of rom boards ordered of minicomputer
        components is set to: 3

      The statement: (#RAM + #ROM > 7)
      failed with value NIL when applied to memory
        components

Evaluating: [CONFIGURE 'IO (FIND 'ALL 'CCS
                '(IMPLEMENTS IO]


Configuring I/O components
  Initial parts are: 4 TERMINAL-1000 and 3 DISC-
      1000

    Evaluating: [CONFIGURE 'TERMINAL (FIND 'ALL 'CCS
                    '(IMPLEMENTS TERMINAL]
```

```
Configuring terminal components
  Initial parts are: 4 TERMINAL-1000

   Evaluating: (CHECK 'REQUIREMENTS)
   Checking REQUIREMENTS constraints in terminal
      components
     The following constraints apply: NO-
        PARTS.CT MAX.TERMINALS.CT
   Checking constraint: NO-PARTS.CT (a MODIFY-
        ON-SUCCESS.CT)

     The statement: ((COUNT CONTENTS)
                     IS ZERO)
     failed with value NIL when applied to
       terminal components
   Checking constraint: MAX.TERMINALS.CT (a
       MODIFY-ON-SUCCESS.CT)

     The statement: ((COUNT CONTENTS)
                     > 4)
     failed with value NIL when applied to
       terminal components

Evaluating: [CONFIGURE 'DISC (FIND 'ALL 'CCS
              '(IMPLEMENTS DISC]


Configuring disk drive components
  Initial parts are: 3 DISC-1000

   Evaluating: (CHECK 'REQUIREMENTS)
   Checking REQUIREMENTS constraints in disk
      drive components
     No REQUIREMENTS constraints were found.

Evaluating: (DETERMINE '(IF-CARDS.NEEDED))
The number of discs ordered of I/O components is
  set to: 3
The number of interface cards required for this
  order of I/O components is set to: 4

Evaluating: [CONFIGURE 'INTERFACE (FIND 'ALL
              'CCS '(IMPLEMENTS INTERFACE]


Configuring interface components
  No initial parts

   Evaluating: (CHECK 'REQUIREMENTS)
   Checking REQUIREMENTS constraints in
      interface components
     The following constraints apply: IF.REQ.CT1
```

```
Checking constraint: IF.REQ.CT1 (a MODIFY-ON-
     SUCCESS.CT)

   The statement: (IF-CARDS.NEEDED =
       CONTENTS)
   succeeded with value T when applied to
     interface components
You have not ordered the minimum number of interface cards.
Replacing contents of  interface components  with  4 IF-1000
; OK? YES
    Removing the following parts:
    Adding the following parts: 4 IF-1000

Evaluating: [CONFIGURE 'CARD-CAGE (FIND 'ALL 'CCS
                '(IMPLEMENTS CARD-CAGE]


Configuring card cage components
  No initial parts

   Evaluating: (CHECK 'REQUIREMENTS)
   Checking REQUIREMENTS constraints in card cage
      components
     The following constraints apply: AT-MOST-
        ONE.CT NO-PARTS.CT
   Checking constraint: AT-MOST-ONE.CT (a MODIFY-
        ON-SUCCESS.CT)

     The statement: ((COUNT CONTENTS)
                    > 1)
     failed with value NIL when applied to card
       cage components
   Checking constraint: NO-PARTS.CT (a MODIFY-ON-
        SUCCESS.CT)

     The statement: ((COUNT CONTENTS)
                    IS ZERO)
     succeeded with value T when applied to card
       cage components
Adding required component to CARD-CAGE.1
Adding 1 CAGE-1000 to card cage components ; OK? YES
    Adding the following parts: 1 CAGE-1000

Evaluating: (CHECK 'REQUIREMENTS)
Checking REQUIREMENTS constraints in mainframe
    components
  The following constraints apply: MAINFRAME.CT.1
Checking constraint: MAINFRAME.CT.1 (a WARN-ON-
    SUCCESS.CT)
The number of interface cards of minicomputer
    components is set to: 3

The statement: (#ROM + #RAM + #IF-CARDS + 1 > 16)
failed with value NIL when applied to mainframe
   components
```

```
Evaluating: [CONFIGURE 'POWER-SUPPLY (FIND 'ALL 'CCS
                '(IMPLEMENTS POWER-SUPPLY]


Configuring power supply components
  No initial parts

    Evaluating: (CHECK 'CONTENTS)
    Checking CONTENTS constraints in power supply
        components
      The following constraints apply: PS.AMBIGUOUS.CT
    Checking constraint: PS.AMBIGUOUS.CT (a MODIFY-ON-
            SUCCESS.CT)

      The statement: ((COUNT CONTENTS)
                      > 1)
      failed with value NIL when applied to power
        supply components

    Evaluating: (CHECK 'REQUIREMENTS)
    Checking REQUIREMENTS constraints in power supply
        components
      The following constraints apply: PS.REQ.CT1
        PS.REQ.CT2 PS.REQ.CT3
    Checking constraint: PS.REQ.CT1 (a MODIFY-ON-
        SUCCESS.CT)
    The power required to operate this order of
        minicomputer components is set to: 8.3

      The statement: (POWER.NEEDED <= 5 AND (NOT
                        (ORDERED? 'SUPPLY-05 T)))
      failed with value NIL when applied to power
        supply components
    Checking constraint: PS.REQ.CT2 (a MODIFY-ON-
        SUCCESS.CT)

      The statement: [POWER.NEEDED > 5 AND POWER.NEEDED
                  <= 10 AND (NOT (ORDERED? 'SUPPLY-10]
      succeeded with value T when applied to power
        supply components
Adding 1 SUPPLY-10 to power supply components ; OK? YES
    Adding the following parts: 1 SUPPLY-10
    Checking constraint: PS.REQ.CT3 (a MODIFY-ON-
        SUCCESS.CT)

      The statement: [POWER.NEEDED > 10 AND
       POWER.NEEDED <= 15 AND (NOT
          (ORDERED? 'SUPPLY-15]
        failed with value NIL when applied to power
          supply components
      Checking EMPTY.BIN constraints in power supply
          components
        The following constraints apply:
          UNASSIGNED.EMPTY.CT
      Checking constraint: UNASSIGNED.EMPTY.CT (a WARN-
          ON-SUCCESS.CT)
```

```
          The statement: (BIN:CONTENTS IS NOT NULL)
          failed with value NIL when applied to original
             components

   Evaluating: (DETERMINE '(FINAL.PARTS))
   The final set of parts of original components is set to:
MINI-1000, CPU-1000, RAM-1000, ROM-1000, TERMINAL-1000,
DISC-1000, IF-1000, CAGE-1000 and SUPPLY-10

   Evaluating: (OUTPUT.ORDER)
What type of output form would you like: FINAL.PARTS

1      CPU-1000
4      RAM-1000
3      ROM-1000
4      TERMINAL-1000
3      DISC-1000
4      IF-1000
1      CAGE-1000
1      SUPPLY-10


What type of output form would you like:
Do you want to use this order to reconfigure something in
finished goods?
Would you like an explanation of this order?
```

APPENDIX III (B)

MINICOMPUTER DESIGN TYPESCRIPT

```
(MINI) COAST executive> RUN

   Evaluating: (INPUT.ORDER)

Processing new order...
Enter name of order file: T
Reading order information...
Order information>
Reading parts to check...
Order line> 1 MINI-1000
Order line>
Reading parts to be shipped F/S...
F/S order line>
Enter order processing options: SHOW-FULL-TRACE
   Adding the following parts: 1 MINI-1000

   Evaluating: (DETERMINE '(TODAYS.DATE ORDERED.PARTS
                            ORDERED.PRODUCT.LINE))
   Today's date of original components is set to: 20-Jun-84
   The ordered set of parts of original components is set
         to: MINI-1000
   The ordered product line of original components is set
         to: minicomputer components
```

```
Evaluating: (IF CHECKING.ORDER
                THEN   **COMMENT**
                     (CONFIGURE (FETCH
                        'ORDERED.PRODUCT.LINE)
                                (FIND 'ALL 'CCS))
                     (CHECK 'EMPTY.BIN))


  Configuring minicomputer components
    Initial parts are: 1 MINI-1000

     Evaluating: [CONFIGURE 'MAIN-FRAME (FIND 'ALL 'CCS
                    '(IMPLEMENTS MAIN-FRAME]


     Configuring mainframe components
       No initial parts

        Evaluating: [CONFIGURE 'CPU (FIND 'ALL 'CCS
                       '(IMPLEMENTS CPU]
        Configuring cpu components
          No initial parts

           Evaluating: (CHECK 'REQUIREMENTS)
           Checking REQUIREMENTS constraints in cpu
              components
             The following constraints apply: AT-MOST-
                  ONE.CT NO-PARTS.CT
           Checking constraint: AT-MOST-ONE.CT (a MODIFY-
                  ON-SUCCESS.CT)

             The statement: ((COUNT CONTENTS)
                             > 1)
             failed with value NIL when applied to cpu
                  components
           Checking constraint: NO-PARTS.CT (a MODIFY-ON-
                  SUCCESS.CT)

             The statement: ((COUNT CONTENTS)
                             IS ZERO)
             succeeded with value T when applied to cpu
                  components
Adding required component to CPU.1
Adding 1 CPU-1000 to cpu components ; OK? YES
            Adding the following parts: 1 CPU-1000

        Evaluating: [CONFIGURE 'MEMORY (FIND 'ALL 'CCS
                       '(IMPLEMENTS MEMORY]

                                  —

        Configuring memory components
          No initial parts

           Evaluating: [CONFIGURE 'RAM (FIND 'ALL 'CCS
                          '(IMPLEMENTS RAM]
```

```
          Configuring random access memory components
            No initial parts

               Evaluating: (CHECK 'REQUIREMENTS)
               Checking REQUIREMENTS constraints in random
                   access memory components
               The following constraints apply: NO-PARTS.CT
               Checking constraint: NO-PARTS.CT (a MODIFY-
                   ON-SUCCESS.CT)

                 The statement: ((COUNT CONTENTS)
                                 IS ZERO)
                 succeeded with value T when applied to
                   random access memory components

Adding required component to RAM.1
Adding 1 RAM-1000 to random access memory components ; OK?
YES
                  Adding the following parts: 1 RAM-1000

               Evaluating: [CONFIGURE 'ROM (FIND 'ALL 'CCS
                                '(IMPLEMENTS ROM]


               Configuring read-only memory components
                 No initial parts

                  Evaluating: (CHECK 'REQUIREMENTS)
                  Checking REQUIREMENTS constraints in read-
                      only memory components
                    The following constraints apply: NO-
                      PARTS.CT
                  Checking constraint: NO-PARTS.CT (a MODIFY-
                      ON-SUCCESS.CT)

                    The statement: ((COUNT CONTENTS)
                                    IS ZERO)
                    succeeded with value T when applied to
                      read-only memory components
Adding required component to ROM.1
Adding 1 ROM-1000 to read-only memory components ; OK? YES
                  Adding the following parts: 1 ROM-1000

               Evaluating: (CHECK 'REQUIREMENTS)
               Checking REQUIREMENTS constraints in memory
                     components
                 The following constraints apply: MEMORY.CT1
               Checking constraint: MEMORY.CT1 (a WARN-ON-
                     SUCCESS.CT)
               The number of ram boards ordered of minicomputer
                     components is set to: 1
               The number of rom boards ordered of minicomputer
                     components is set to: 1
```

```
The statement: (#RAM + #ROM > 7)
failed with value NIL when applied to memory
    components

Evaluating: [CONFIGURE 'IO (FIND 'ALL 'CCS
                '(IMPLEMENTS IO]


Configuring I/O components
  No initial parts

   Evaluating: [CONFIGURE 'TERMINAL (FIND 'ALL 'CCS
                   '(IMPLEMENTS TERMINAL]

   Configuring terminal components
     No initial parts

      Evaluating: (CHECK 'REQUIREMENTS)
      Checking REQUIREMENTS constraints in terminal
          components
        The following constraints apply: NO-
          PARTS.CT MAX.TERMINALS.CT
      Checking constraint: NO-PARTS.CT (a MODIFY-
          ON-SUCCESS.CT)

        The statement: ((COUNT CONTENTS)
                        IS ZERO)
        succeeded with value T when applied to
          terminal components
Adding required component to TERMINAL.1
Adding 1 TERMINAL-1000 to terminal components ; OK? YES
          Adding the following parts: 1 TERMINAL-1000
          Checking constraint: MAX.TERMINALS.CT (a
              MODIFY-ON-SUCCESS.CT)

            The statement: ((COUNT CONTENTS)
                           > 4)
            failed with value NIL when applied to
              terminal components

   Evaluating: [CONFIGURE 'DISC (FIND 'ALL 'CCS
                   '(IMPLEMENTS DISC]


   Configuring disk drive components
     No initial parts

      Evaluating: (CHECK 'REQUIREMENTS)
      Checking REQUIREMENTS constraints in disk
          drive components
        No REQUIREMENTS constraints were found.

   Evaluating: (DETERMINE '(IF-CARDS.NEEDED))
   The number of discs ordered of I/O components is
          set to: 0
```

```
The number of interface cards required for this
       order of I/O components is set to: 1

Evaluating: [CONFIGURE 'INTERFACE (FIND 'ALL
                'CCS '(IMPLEMENTS INTERFACE]


Configuring interface components
  No initial parts

   Evaluating: (CHECK 'REQUIREMENTS)
   Checking REQUIREMENTS constraints in
       interface components
     The following constraints apply: IF.REQ.CT1
   Checking constraint: IF.REQ.CT1 (a MODIFY-ON-
       SUCCESS.CT)

     The statement: (IF-CARDS.NEEDED =
       CONTENTS)
     succeeded with value T when applied to
       interface components
You have not ordered the minimum number of interface cards.
Replacing contents of  interface components  with  1 IF-1000
; OK? YES
      Removing the following parts:
      Adding the following parts: 1 IF-1000

Evaluating: [CONFIGURE 'CARD-CAGE (FIND 'ALL 'CCS
                '(IMPLEMENTS CARD-CAGE]

Configuring card cage components
  No initial parts

   Evaluating: (CHECK 'REQUIREMENTS)
   Checking REQUIREMENTS constraints in card cage
       components
     The following constraints apply: AT-MOST-
       ONE.CT NO-PARTS.CT
   Checking constraint: AT-MOST-ONE.CT (a MODIFY-
       ON-SUCCESS.CT)

     The statement: ((COUNT CONTENTS)
                     > 1)
     failed with value NIL when applied to card
       cage components
   Checking constraint: NO-PARTS.CT (a MODIFY-ON-
       SUCCESS.CT)

     The statement: ((COUNT CONTENTS)
                     IS ZERO)
     succeeded with value T when applied to card
       cage components
Adding required component to CARD-CAGE.1
Adding 1 CAGE-1000 to card cage components ; OK? YES
      Adding the following parts: 1 CAGE-1000
```

```
Evaluating: (CHECK 'REQUIREMENTS)
Checking REQUIREMENTS constraints in mainframe
          components
  The following constraints apply: MAINFRAME.CT.1
Checking constraint: MAINFRAME.CT.1 (a WARN-ON-
          SUCCESS.CT)
The number of interface cards of minicomputer
          components is set to: 1

  The statement: (#ROM + #RAM + #IF-CARDS + 1 > 16)
  failed with value NIL when applied to mainframe
          components

Evaluating: [CONFIGURE 'POWER-SUPPLY (FIND 'ALL 'CCS
              '(IMPLEMENTS POWER-SUPPLY]

Configuring power supply components
  No initial parts

  Evaluating: (CHECK 'CONTENTS)
  Checking CONTENTS constraints in power supply
          components
    The following constraints apply: PS.AMBIGUOUS.CT
  Checking constraint: PS.AMBIGUOUS.CT (a MODIFY-ON-
          SUCCESS.CT)

    The statement: ((COUNT CONTENTS)
                    > 1)
    failed with value NIL when applied to power
          supply components

  Evaluating: (CHECK 'REQUIREMENTS)
  Checking REQUIREMENTS constraints in power supply
          components
    The following constraints apply: PS.REQ.CT1
          PS.REQ.CT2 PS.REQ.CT3
  Checking constraint: PS.REQ.CT1 (a MODIFY-ON-
          SUCCESS.CT)
  The power required to operate this order of
          minicomputer components is set to: 3.7

    The statement: (POWER.NEEDED <= 5 AND (NOT
          (ORDERED? 'SUPPLY-05 T)))
    succeeded with value T when applied to power
          supply components
Adding 1 SUPPLY-05 to power supply components ; OK? YES
  Adding the following parts: 1 SUPPLY-05
  Checking constraint: PS.REQ.CT2 (a MODIFY-ON-
          SUCCESS.CT)

    The statement: [POWER.NEEDED > 5 AND POWER.NEEDED
          <= 10 AND (NOT (ORDERED? 'SUPPLY-10]
    failed with value NIL when applied to power
          supply components
  Checking constraint: PS.REQ.CT3 (a MODIFY-ON-
          SUCCESS.CT)
```

```
          The statement: [POWER.NEEDED > 10 AND
                  POWER.NEEDED <= 15 AND (NOT (ORDERED?
                    'SUPPLY-15]
          failed with value NIL when applied to power
                  supply components
        Checking EMPTY.BIN constraints in power supply
                  components
          The following constraints apply:
                  UNASSIGNED.EMPTY.CT
        Checking constraint: UNASSIGNED.EMPTY.CT (a WARN-
                  ON-SUCCESS.CT)

          The statement: (BIN:CONTENTS IS NOT NULL)
          failed with value NIL when applied to original
                  components

   Evaluating: (DETERMINE '(FINAL.PARTS))
   The final set of parts of original components is set to:
MINI-1000, CPU-1000, RAM-1000, ROM-1000, TERMINAL-1000, IF-
1000, CAGE-1000 and SUPPLY-05

   Evaluating: (OUTPUT.ORDER)
What type of output form would you like: FINAL.PARTS

1      CPU-1000
1      RAM-1000
1      ROM-1000
1      TERMINAL-1000
1      IF-1000
1      CAGE-1000
1      SUPPLY-05


What type of output form would you like:
Do you want to use this order to reconfigure something in
finished goods?
Would you like an explanation of this order?
```

APPENDIX IV (A)

KNOWLEDGE BASE FOR MINICOMPUTER
FUNCTIONAL HIERARCHY

```
(* MINI-FHS definition)
(DEFINSTANCE MINI-FHS FHS
  SET.OF MINIKB
  FILED.ON MINIFH
  ELEMENTS (CARD-CAGE CPU DISC INTERFACE IO MAIN-FRAME
MEMORY MINICOMPUTER POWER-SUPPLY RAM ROM TERMINAL))

(DEFCLASS CARD-CAGE (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (MAIN-FRAME)
   DESCR "card cage")
  )
```

```
(DEFCLASS CPU (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (MAIN-FRAME)
   DESCR "cpu")
  )

(DEFCLASS DISC (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (IO)
   DESCR "disk drive")
  )

(DEFCLASS INTERFACE (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (IO)
   DESCR "interface")
  )

(DEFCLASS IO (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (MAIN-FRAME)
   DESCR "I/O")
  )

(DEFCLASS MAIN-FRAME (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (MINICOMPUTER)
   DESCR "mainframe")
  )

(DEFCLASS MEMORY (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (MAIN-FRAME)
   DESCR "memory")
  )

(DEFCLASS MINICOMPUTER (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   PROD.GEN (PRODUCT.LINE)
   DESCR "minicomputer")
  )

(DEFCLASS POWER-SUPPLY (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (MINICOMPUTER)
   DESCR "power supply")
  )
```

```
(DEFCLASS RAM (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (MEMORY)
   DESCR "random access memory")
  )

(DEFCLASS ROM (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (MEMORY)
   DESCR "read-only memory")
  )

(DEFCLASS TERMINAL (BIN)
  ()
  (METACLASS FUNCTION.CLASS
   COMPOSES (IO)
   DESCR "terminal")
  )
```

APPENDIX IV (B)

KNOWLEDGE BASE FOR MINICOMPUTER
TASK BLOCKS

```
(* MINI-TBS definition)
(DEFINSTANCE MINI-TBS TBS
  SET.OF MINIKB
  FILED.ON MINIFH
  ELEMENTS (CARD-CAGE.TB CPU.TB DISC.TB INTERFACE.TB IO.TB
MAIN-FRAME.TB MEMORY.TB
               MINICOMPUTER.TB POWER-SUPPLY.TB RAM.TB ROM.TB
                   TERMINAL.TB))

(DEFTASKBLOCK CARD-CAGE.TB CARD-CAGE () (CHECK
               'REQUIREMENTS)
  )

(DEFTASKBLOCK CPU.TB CPU () (CHECK 'REQUIREMENTS)
  )

(DEFTASKBLOCK DISC.TB DISC () (CHECK 'REQUIREMENTS)
  )

(DEFTASKBLOCK INTERFACE.TB INTERFACE () (CHECK
               'REQUIREMENTS)
  )

(DEFTASKBLOCK IO.TB IO ()

          (* The order of the CONFIGURE statements is
              significant below.
      In particular, INTERFACE should be configured last!)
```

```
  (CONFIGURE 'TERMINAL (FIND 'ALL 'CCS '(IMPLEMENTS
                TERMINAL)))
  (CONFIGURE 'DISC (FIND 'ALL 'CCS '(IMPLEMENTS DISC)))
  (DETERMINE '(IF-CARDS.NEEDED))
  (CONFIGURE 'INTERFACE (FIND 'ALL 'CCS '(IMPLEMENTS
                INTERFACE))))

(DEFTASKBLOCK MAIN-FRAME.TB MAIN-FRAME () (CONFIGURE 'CPU
(FIND 'ALL 'CCS '(IMPLEMENTS CPU)))
  (CONFIGURE 'MEMORY (FIND 'ALL 'CCS '(IMPLEMENTS MEMORY)))
  (CONFIGURE 'IO (FIND 'ALL 'CCS '(IMPLEMENTS IO)))
  (CONFIGURE 'CARD-CAGE (FIND 'ALL 'CCS '(IMPLEMENTS CARD-
                CAGE)))
  (CHECK 'REQUIREMENTS))

(DEFTASKBLOCK MEMORY.TB MEMORY () (CONFIGURE 'RAM (FIND 'ALL
                'CCS '(IMPLEMENTS RAM)))
  (CONFIGURE 'ROM (FIND 'ALL 'CCS '(IMPLEMENTS ROM)))
  (CHECK 'REQUIREMENTS))

(DEFTASKBLOCK MINICOMPUTER.TB MINICOMPUTER () (CONFIGURE
                'MAIN-FRAME
    (FIND 'ALL 'CCS '(IMPLEMENTS

    MAIN-FRAME)))
  (CONFIGURE 'POWER-SUPPLY (FIND 'ALL 'CCS '(IMPLEMENTS
                POWER-SUPPLY))))

(DEFTASKBLOCK POWER-SUPPLY.TB POWER-SUPPLY () (CHECK
                'CONTENTS)
  (CHECK 'REQUIREMENTS))

(DEFTASKBLOCK RAM.TB RAM () (CHECK 'REQUIREMENTS)
  )

(DEFTASKBLOCK ROM.TB ROM () (CHECK 'REQUIREMENTS)
  )

(DEFTASKBLOCK TERMINAL.TB TERMINAL () (CHECK 'REQUIREMENTS)
  )
```

APPENDIX IV (C)

KNOWLEDGE BASE FOR MINICOMPUTER
PARTS CATALOG

```
(* MINI-PRODS definition)
(DEFINSTANCE MINI-PRODS PRODS
  SET.OF MINIKB
  FILED.ON MINIPROD
  ELEMENTS (CAGE-1000 CPU-1000 DISC-1000 IF-1000 MINI-1000
RAM-1000 ROM-1000 SUPPLY-05 SUPPLY-10 SUPPLY-15 TERMINAL-
1000))
```

```
(DEFCLASS CAGE-1000 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN CARD-CAGE)
  )

(DEFCLASS CPU-1000 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN CPU)
  )

(DEFCLASS DISC-1000 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN DISC)
  )

(DEFCLASS IF-1000 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN INTERFACE)
  )

(DEFCLASS MINI-1000 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN MINICOMPUTER)
  )

(DEFCLASS RAM-1000 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN RAM)
  )

(DEFCLASS ROM-1000 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN ROM)
  )

(DEFCLASS SUPPLY-05 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN POWER-SUPPLY)
  )

(DEFCLASS SUPPLY-10 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN POWER-SUPPLY)
  )
```

```
(DEFCLASS SUPPLY-15 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN POWER-SUPPLY)
  )

(DEFCLASS TERMINAL-1000 (PRODUCT.ID)
  ()
  (METACLASS PID
   IMPLEMENTS.FN TERMINAL)
  )

(* MINI-ECTS definition)
(DEFINSTANCE MINI-ECTS ECTS
  SET.OF MINIKB
  FILED.ON MINIPROD
  ELEMENTS NIL)
```

APPENDIX IV (D)

KNOWLEDGE BASE FOR MINICOMPUTER
CONSTRAINTS & BIN VARIABLES
KNOWLEDGE BASE FUNCTIONS

```
(* MINI-CTS definition)
(DEFINSTANCE MINI-CTS CTS
  SET.OF MINIKB
  FILED.ON MINICT
  ELEMENTS (AT-MOST-ONE.CT IF.REQ.CT1 MAINFRAME.CT.1
MAX.TERMINALS.CT MEMORY.CT1 NO-PARTS.CT PS.AMBIGUOUS.CT
PS.REQ.CT1 PS.REQ.CT2 PS.REQ.CT3))

(DEFCLASS AT-MOST-ONE.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   SET-PARTS (LIST (A PRODUCT.QTY WITH QTY = 1 NAME =
(GETCLASS (CAR (CURRENT.BIN:CONTENTS)))))
   STATEMENT ((COUNT CONTENTS)
          > 1)
   BIN.TYPES (CPU CARD-CAGE)
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS IF.REQ.CT1 (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   SET-PARTS (LIST (A PRODUCT.QTY WITH QTY = (FETCH 'IF-
              CARDS.NEEDED)
               NAME = 'IF-1000))
   MESSAGE ("You have not ordered the minimum number of
interface cards.")
```

```
   BIN.VARS.REQUIRED (IF-CARDS.NEEDED)
   STATEMENT (IF-CARDS.NEEDED = CONTENTS)
   BIN.TYPES (INTERFACE)
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS MAINFRAME.CT.1 (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   MESSAGE ("You have ordered more than 16 cards for this
             mainframe.")
   BIN.VARS.REQUIRED (#ROM #RAM #IF-CARDS)
   STATEMENT (#ROM + #RAM + #IF-CARDS + 1 > 16)
   BIN.TYPES (MAIN-FRAME)
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS MAX.TERMINALS.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   SET-PARTS (LIST (A PRODUCT.QTY WITH QTY = 4 NAME =
'TERMINAL-1000))
   MESSAGE ("You have ordered too many terminals; the
             maximum is 4.")
   STATEMENT ((COUNT CONTENTS)
         > 4)
   BIN.TYPES (TERMINAL)
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS MEMORY.CT1 (CTI)
  ()
  (METACLASS WARN-ON-SUCCESS.CT
   MESSAGE ("You have ordered too many memory (RAM and ROM)
             cards.")
   BIN.VARS.REQUIRED (#RAM #ROM)
   STATEMENT (#RAM + #ROM > 7)
   BIN.TYPES (MEMORY)
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS NO-PARTS.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   ADD-PARTS (LIST (A PRODUCT.QTY WITH QTY = 1 NAME =
            (SELECTQ (GETCLASS CURRENT.BIN)
            (RAM 'RAM-1000)
            (ROM 'ROM-1000)
            (TERMINAL 'TERMINAL-1000)
            (INTERFACE 'IF-1000)
            (CARD-CAGE 'CAGE-1000)
            (CPU 'CPU-1000)
            (SHOULDNT))))
   MESSAGE ("Adding required component to " CURRENT.BIN)
```

```
   STATEMENT ((COUNT CONTENTS)
          IS ZERO)
   BIN.TYPES (RAM ROM TERMINAL CPU CARD-CAGE)
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS PS.AMBIGUOUS.CT (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   SET-PARTS (MAKE.PRODUCT.QTYS CURRENT.BIN:CONTENTS)
   STATEMENT ((COUNT CONTENTS)
          > 1)
   BIN.TYPES (POWER-SUPPLY)
   CHECK.KEYWORDS (CONTENTS))
  )

(DEFCLASS PS.REQ.CT1 (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   ADD-PARTS (LIST (A PRODUCT.QTY WITH QTY = 1 NAME =
              'SUPPLY-05))
   BIN.VARS.REQUIRED (POWER.NEEDED)
   STATEMENT (POWER.NEEDED <= 5 AND (NOT (ORDERED? 'SUPPLY-
              05 T)))
   BIN.TYPES (POWER-SUPPLY)
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS PS.REQ.CT2 (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   ADD-PARTS (LIST (A PRODUCT.QTY WITH QTY = 1 NAME =
              'SUPPLY-10))
   BIN.VARS.REQUIRED (POWER.NEEDED)
   STATEMENT (POWER.NEEDED > 5 AND POWER.NEEDED <= 10 AND
              (NOT (ORDERED? 'SUPPLY-10)))
   BIN.TYPES (POWER-SUPPLY)
   CHECK.KEYWORDS (REQUIREMENTS))
  )

(DEFCLASS PS.REQ.CT3 (CTI)
  ()
  (METACLASS MODIFY-ON-SUCCESS.CT
   ADD-PARTS (LIST (A PRODUCT.QTY WITH QTY = 1 NAME =
              'SUPPLY-15))
   BIN.VARS.REQUIRED (POWER.NEEDED)
   STATEMENT (POWER.NEEDED > 10 AND POWER.NEEDED <= 15 AND
              (NOT (ORDERED? 'SUPPLY-15)))
   BIN.TYPES (POWER-SUPPLY)
   CHECK.KEYWORDS (REQUIREMENTS))
  )
```

```
(* MINI-BVS definition)
(DEFINSTANCE MINI-BVS BVS
  SET.OF MINIKB
  FILED.ON MINICT
  ELEMENTS (#DISCS #IF-CARDS #RAM #ROM #TERMINALS IF-
              CARDS.NEEDED POWER.NEEDED))

(DEFCLASS #DISCS (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (IO)
   HOW.TO.DETERMINE (COMPUTE.#DISC ASK.QUESTION)
   LEGAL.VALUES INTEGER
   DESCR "the number of discs ordered")
  )


(DEFCLASS #IF-CARDS (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (MINICOMPUTER)
   HOW.TO.DETERMINE (COMPUTE.#IF-CARDS ASK.QUESTION)
   LEGAL.VALUES INTEGER
   DESCR "the number of interface cards")
  )


(DEFCLASS #RAM (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (MINICOMPUTER)
   HOW.TO.DETERMINE (COMPUTE.#RAM ASK.QUESTION)
   LEGAL.VALUES INTEGER
   DESCR "the number of RAM boards ordered")
  )


(DEFCLASS #ROM (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (MINICOMPUTER)
   HOW.TO.DETERMINE (COMPUTE.#ROM ASK.QUESTION)
   LEGAL.VALUES INTEGER
   DESCR "the number of ROM boards ordered")
  )


(DEFCLASS #TERMINALS (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (IO)
   HOW.TO.DETERMINE (COMPUTE.#TERMINALS ASK.QUESTION)
   LEGAL.VALUES INTEGER
   DESCR "the number of terminals ordered")
  )
```

```
(DEFCLASS IF-CARDS.NEEDED (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (IO)
   HOW.TO.DETERMINE (COMPUTE.IF-CARDS.NEEDED ASK.QUESTION)
   LEGAL.VALUES INTEGER
   DESCR "the number of interface cards required for this
          order")
  )

(DEFCLASS POWER.NEEDED (BVI)
  ()
  (METACLASS BIN.VAR
   BIN.TYPES (MINICOMPUTER)
   HOW.TO.DETERMINE (COMPUTE.POWER.NEEDED ASK.QUESTION)
   LEGAL.VALUES NUMBER
   DESCR "the power required to operate this order")
  )
(* MINI-BVMS definition)
(DEFINSTANCE MINI-BVMS BVMS
  SET.OF MINIKB
  FILED.ON MINICT
  ELEMENTS (COMPUTE.#DISC COMPUTE.#IF-CARDS COMPUTE.#RAM
COMPUTE.#ROM COMPUTE.#TERMINALS
COMPUTE.IF-CARDS.NEEDED COMPUTE.POWER.NEEDED))

(DEFKBFUN COMPUTE.#DISC ()(COUNT.PARTS 'DISC-1000 T)
  )

(DEFKBFUN COMPUTE.#IF-CARDS ()(COUNT.PARTS 'IF-1000 T)
  )

(DEFKBFUN COMPUTE.#RAM ()(COUNT.PARTS 'RAM-1000 T)
  )

(DEFKBFUN COMPUTE.#ROM ()(COUNT.PARTS 'ROM-1000 T)
  )

(DEFKBFUN COMPUTE.#TERMINALS ()(COUNT.PARTS 'TERMINAL-1000
                  T)
  )

(DEFKBFUN COMPUTE.IF-CARDS.NEEDED ()

          (* Compute the minimum number of interface cards
                  required by this order.
       This calculation assumes that the number of terminals
              and discs are correct and acceptable so
               configure those first.)

  (PROG (#TERMINALS:INTEGER #DISCS:INTEGER)
        (#TERMINALS :=(SAND CURRENT.BIN 'FETCH '#TERMINALS))
        (#DISCS :=(SAND CURRENT.BIN 'FETCH '#DISCS))
        (RETURN (SELECTQ #DISCS
```

```
    (0 1)
    (1 (IF #TERMINALS = 1
  THEN 1
    ELSE 2))
    (2 (IF #TERMINALS = 4
  THEN 3
        ELSE 2))
    (3 (IF #TERMINALS = 4
  THEN 4
        ELSE 3))
    #DISCS))))

(DEFKBFUN COMPUTE.POWER.NEEDED ()(PROG (#RAM:INTEGER
          #ROM:INTEGER #IF-CARDS:INTEGER)
          (#RAM :=(SAND CURRENT.BIN 'FETCH '#RAM))
          (#ROM :=(SAND CURRENT.BIN 'FETCH '#ROM))
          (#IF-CARDS :=(SAND CURRENT.BIN 'FETCH '#I
             F-CARDS))
          (RETURN (2 + 1*#RAM + .5*#ROM + .2*#IF-CA
             RDS)))
  )
```

APPENDIX V

GLOSSARY

ASSEMBLY
A specific set of product instances or parts.

ASSEMBLY DESCRIPTION
A Boolean function or predicate defining a set of parts in terms of a set of part names, names of assemblies or part types. A collection of parts satisfies an assembly description when the parts fit that description.

BIN
A set of product instances created during execution of a CONFIGURE operation in a task block. The product instances are obtained from the order lines, by an EXPANSION operation upon parts initially placed in the bin, or by application of a modification constraint.

BIN TREE
A hierarchical structure of bins corresponding to the hierarchy of the functional classes of the bins.

CHECK
An operation which applies a specified set of constraints to the product instances or parts in the current bin in order to check whether there are any configuration problems with the parts in the bin.

CONDITION
A Boolean function of the elements in a defined set.

CONFIGURE
An operation which creates a new bin of a specified functional class and moves a respective subset of parts from the current bin to the new bin, and then invokes execution of the task block associated with the type of the new bin.

CONFIGURATION
The grouping of a set of elements into subsets.

CONSTRAINTS
A statement of a condition that certain parts must satisfy and a respective action to be conditionally executed in response to whether the condition is satisfied when the constraint is applied.

EXPAND
An operation which selects a specified subset of parts from the current bin and adds any new parts that correspond to the "product expansions" or subcomponents of the specified subset of parts.

FIELD SERVICE PARTS
A list of parts, such as replacement and spare parts, that are included in an order but which are not checked for compatibility with themselves or with the rest of the order.

FIND
An operation which matches the parts in the current bin to an assembly description and copies the matching parts to an assembly list.

FUNCTION
(1) The purpose or performance of an assembly.
(2) A numerical or logical value responsive to the domain of the function. A function of the parts in an assembly, for example, may indicate whether a specified number of specified parts are included in the assembly or whether the parts have specified conditions or attributes. In a computer program, functions are applied or evaluated at only specified instants of time.

FUNCTIONAL HIERARCHY
The decomposition of a configuration or product into its major functional components of sub-assemblies.

INITIAL BIN
The root of the bin tree

MODIFICATION CONSTRAINT
A constraint which changes the set of product instances or parts in the current bin, and may also issue a warning message to the user.

OPERATION
A specific action in a control procedure or computer program which, when executed, modifies the sequence of execution of the control procedure or modifies the data or configuration elements processed by the control procedure.

ORDER
The input to the order processing system including order information, order lines, and field serivce parts.
ORDER INFORMATION
Information which primarily identifies an order, such as customer name and order number, and does not affect the configuration process.
ORDER LINES
A list of parts ordered and the quantity of each part that will be checked.
PRODUCT EXPANSION
For a specified part or product instance, the set of corresponding sub-parts or product instances making up the specified part or product instance.
PRODUCT INSTANCE
An explicit representation of an individual part considered by the order processing system.
TASK BLOCK
A set of imperative language statements specifying operations to perform with respect to associated bins.
UNASSIGNED
A type of bin denoting that the configuration system has not yet determined what product the parts of the bin will implement and therefore which knowledge base should be used to configure the order.
WARNING CONSTRAINT
A constraint which issues a message, informing the user of a problem with the configuration.

What is claimed is:

1. A knowledge system comprising a computer having a memory storing a knowledge base, said knowledge base including

predefined descriptions of assemblies of predefined components for indicating whether a specified component is permitted in a specified assembly, and

for at least some of said descriptions of assemblies, sets of predefined conditions of different properties of the components in the assembly for indicating whether certain predefined properties of a specified set of components are compatible within the assembly,

said memory comprising a control procedure for

first obtaining a predetermined initial set of predefined components,

thereafter matching the initial set of components to the predefined descriptions of assemblies to determine respective sets of matching components including the components in the initial set which are also permitted in the respective assemblies,

thereafter applying the sets of conditions to the respective sets of matching components for the respective assemblies, and

indicating whether the respective conditions are satisfied,

said computer including means for executing said control procedure to thereby first obtain said predetermined initial set of predefined components, thereafter match said initial set of components to said predefined descriptions of assemblies, thereafter apply said sets of conditions to the respective sets of matching components for the respective assemblies and generate an indication of whether the conditions are satisfied.

2. The knowledge system as claimed in claim 1, wherein said descriptions of assemblies include the names of at least some of said components in the initial set of components.

3. The knowledge system as claimed in claim 1, wherein said knowledge base further includes a catalog of components including component names and respective names of assemblies comprising said components, and wherein said descriptions of assemblies include the names of some of said assemblies in said catalog of components.

4. The knowledge system as claimed in claim 3, wherein said descriptions of assemblies comprise Boolean functions of names of assemblies.

5. The knowledge system as claimed in claim 1, wherein said control procedure includes a set of control steps for obtaining attributes of the components in said initial set of components, and wherein said descriptions of assemblies include respective Boolean functions of said attributes.

6. The knowledge system as claimed in claim 1, wherein said control procedure includes a set of control steps for obtaining a predetermined initial set of predefined components by inputting a predetermined list of components.

7. The knowledge system as claimed in claim 1, wherein the knowledge base further includes respective actions for said conditions, and wherein said control procedure includes a set of control steps for carrying out said actions for indicating whether the respective conditions are satisfied.

8. The knowledge system as claimed in claim 7, wherein said actions comprise adding and deleting specified components from said initial set of components.

9. The knowledge system as claimed in claim 1, wherein the descriptions of assemblies of predefined components include at least one definition of a relationship between at least two of said descriptions of assemblies, and wherein the control procedures includes a set of control steps for matching the initial set of components to at least one of said two descriptions of assemblies in response to the descriptions of the two assemblies and in response to the defined relationship between the two assemblies.

10. The knowledge system as claimed in claim 9, wherein said relationship is a structural relationship between said two assemblies.

11. The knowledge system as claimed in claim 9, wherein said relationship is a functional relationship between said two assemblies.

12. The knowledge system as claimed in claim 1, wherein some of said assemblies are sub-assemblies of other of said assemblies, and wherein said knowledge base includes a definition of a hierarchy of said descriptions of assemblies defining offspring-parent relationships between said descriptions of sub-assemblies and the respective descriptions of assemblies which comprise said sub-assemblies, and wherein said control procedure for matching the initial set of components to the predefined descriptions of assemblies includes means for first matching the list of components to the parent descriptions of assemblies, and then matching the list of the respecting matching components for the respective parent assemblies to their respective offspring descriptions of sub-assemblies.

13. The knowledge system as claimed in claim 12, wherein some of said parent descriptions of assemblies represent structures comprising their respective sub-assemblies physically connected together.

14. The knowledge system as claimed in claim 12, wherein some of said parent descriptions of assemblies represent functions collectively performed by the respective offspring sub-assemblies.

15. The knowledge system as claimed in claim 1, wherein said knowledge base further comprises a set of expansion rules defining expandable components including some of said components in the initial set of components in terms of respective sub-components of the expandable components and wherein said control procedure further includes a set of control steps for selecting expandable components from the initial set of components and adding the respective sub-components corresponding to the selected expandable components to the initial set of components.

16. The knowledge system as claimed in claim 15, wherein said set of control steps for selecting expandable components includes control steps for searching the initial set of components for specified components and when any of the specified expandable components are found, expanding those expandable components by adding their respective sub-components to the initial set of components.

17. The knowledge system as claimed in claim 1, further comprising a working configuration portion of said memory, and wherein said set of control steps for matching the initial set of components to the defined assemblies includes control steps for recording the respective matching components in respective bin portions of said working configuration portion of said memory.

18. The knowledge system as claimed in claim 17, wherein said knowledge base further includes respective modification actions for making specified changes to the recorded set of matching components in the respective bin portions of said working configuration portion of said memory, and wherein the said control procedure for applying the sets of conditions includes a set of control steps for conditionally executing said changes in response to whether the respective conditions are found to be satisfied for the respective assemblies when the respective conditions are applied.

19. The knowledge system as claimed in claim 18, wherein said conditions include descriptions of assemblies including specific components.

20. The knowledge system as claimed in claim 18, wherein said descriptions of assemblies include Boolean functions of specified attributes of components.

21. The knowledge system as claimed in claim 1, wherein said control procedure includes

a built-in control procedure independent of the descriptions of assemblies and said predefined conditions, and

task blocks defining control steps for said matching of said components to specified descriptions of assemblies and applying said conditions to the respective matching components, said task blocks being executed when specified steps in said built-in control procedure are reached.

22. The knowledge system as claimed in claim 21, wherein said task blocks include task blocks associated with respective assemblies and executed when said components are matched with the respective descriptions of assemblies of the task blocks.

23. The knowledge system as claimed in claim 22, wherein said descriptions of assemblies of predefined components include Boolean functions of specified attributes of components.

24. The knowledge system as claimed in claim 22, wherein said control steps defined by said task blocks include control steps for applying specified subsets of said conditions corresponding to the specified descriptions of assemblies.

25. The knowledge system as claimed in claim 21, wherein said task blocks include a specified task block to be executed before the other task blocks are executed.

26. The knowledge system as claimed in claim 21, wherein said conditions include Boolean functions of specified attributes of specified components, and wherein said knowledge base includes a knowledge base function associated with a respective one of said specified attributes, said knowledge base function being implicitly invoked when said condition is applied including said Boolean function of said specified attribute, so that the knowledge base function determines a value for its respective specified attribute.

27. The knowledge system as claimed in claim 21, wherein said built-in control procedure includes a set of control steps for generating a trace of the operations performed by said control procedure including the sequence of control steps performed when said task blocks are executed, and wherein said control procedure also includes a set of control steps for generating a user-understandable explanation from said trace and transmitting said explanation to a user.

28. The knowledge system as claimed in claim 1, wherein said control procedure includes a set of control steps for generating a trace of the results of said matching of said components and said application of said conditions and also includes a set of control steps for generating an explanation understandable to a human user from said trace and transmitting said explanation to a human user.

29. A knowledge system comprising a computer having a memory storing a knowledge base, said knowledge base including

predefined descriptions of assemblies of predefined components for indicating whether a specified component is permitted in a specified assembly, and

corresponding sets of assembly constraints including predefined conditions of different properties of the components in their respective assemblies for indicating whether certain predefined properties of a specified set of components are compatible within the assembly, and also including predefined actions to perform when the constraints are applied and the conditions are satisfied including outputting a warning to a user indicating specified conditions are satisfied,

said memory further comprising a control procedure for

first obtaining a predetermined list of components,

thereafter matching the components in said list to the predefined descriptions of assemblies to determine respective sets of matching components containing the components in said list which are permitted in the respective assemblies,

thereafter applying the sets of assembly constraints to the respective sets of matching components from the list for the respective assemblies, and

conditionally performing the respective actions in response to whether their respective conditions are satisfied when the assembly constraints are applied, said computer including means for executing said control procedure to thereby first obtain said predetermined list of components, thereafter match the components in said list to the predefined descriptions of assemblies, thereafter apply said sets of constraints to the respective sets of matching components from said list of components, and con-

ditionally perform said respective actions in response to whether their respective conditions are satisfied when the assembly constraints are applied.

30. The knowledge system as claimed in claim 29 wherein said descriptions of assemblies define a hierarchy of descriptions of assemblies and sub-assemblies, and said control procedure for matching includes a procedure for matching said components in the respective assemblies to the descriptions of specified sub-assemblies to determine the components for which said constraints are applied.

31. The knowledge system as claimed in claim 29, wherein said constraints include conditions responsive to whether specified parts have been matched to the respective descriptions of assemblies.

32. The knowledge system as claimed in claim 29, wherein said constraints include conditions responsive to whether specified parts have specified attributes.

33. The knowledge system as claimed in claim 32, wherein the knowledge base includes definitions of variables for specifying the attributes of components included in the descriptions of respective assemblies.

34. The knowledge system as claimed in claim 33, wherein said knowledge base includes knowledge base functions for at least some of said variables specifying steps for determining values for the respective attributes of said components, and said control procedure includes a set of control steps for implicitly invoking and executing the respective knowledge base functions when applying constraints including conditions referencing the respective variables having the knowledge base functions.

35. The knowledge system as claimed in claim 29, further comprising a working configuration portion of said memory, and wherein said control procedure for matching the components in said list to the descriptions of assemblies includes a set of control steps for recording the respective matching components in respective bin portions of said working configuration portion of said memory.

36. The knowledge system as claimed in claim 35, wherein said control procedure for applying the sets of assembly constraints includes a set of control steps for applying a specified set of constraints to the matching components recorded in a specified bin portion of said working configuration portion of said memory.

37. The knowledge system as claimed in claim 35, wherein said knowledge base includes expansion rules defining sub-components for respective components, and wherein said control procedure includes a set of control steps for searching a specified bin for at least some of said components having sub-components defined by said expansion rules, and recording the respective sub-components in the specified bin.

38. The knowledge system as claimed in claim 35, wherein the actions of the assembly constraints include actions specifying modifications to the set of components recorded in a specified bin.

39. The knowledge system as claimed in claim 35, wherein said working configuration portion of said memory includes a current bin memory location for storing a value specifying a particular bin.

40. The knowledge system as claimed in claim 39, wherein the control procedure includes task blocks including control steps, at least some of the bins having an associated task block for specifying operations to perform in connection with their respective bins, and the control procedure includes a set of steps for executing the task block associated with a specified bin.

41. The knowledge system as claimed in claim 40, wherein the set of steps for executing the task block associated with a specified bin include the steps of pushing the value of the current bin memory location onto a stack, setting the value of the current bin memory location to specify the specified bin, executing the control steps in the task block for the specified bin, popping the stack and assigning the popped value to the current bin memory location.

42. The knowledge system as claimed in claim 40, wherein the task blocks include imperative language statements specifying distinct steps in said control procedure.

43. The knowledge system as claimed in claim 42, wherein the knowledge base includes expansion rules specifying that at least some of said components are expandable and are comprised of sub-components and wherein the imperative language statements include a separate statement for specifying that for each expandable component in a specified bin, its respective sub-components are to be recorded in the specified bin.

44. The knowledge system as claimed in claim 42, wherein the imperative language statements include a separate statement for specifying that specified assembly constraints are to be applied to the components in a specified bin.

45. The knowledge system as claimed in claim 42, wherein the imperative language statements include a separate statement for executing a specified task block.

46. The knowledge system as claimed in claim 42, wherein the control procedure includes a set of control steps for generating a sequential record of the imperative language statements that are executed, the components added to bins, the constraints applied, and the actions that are executed when the conditions of the applied constraints are satisfied.

47. The knowledge system as claimed in claim 40, wherein said knowledge base includes a plurality of separate portions for processing predetermined lists to configure different respective products, and wherein the control procedure includes control steps for obtaining said predetermined list by an input operation and for selecting the knowledge base portion corresponding to a product number in said predetermined list.

48. The knowledge system as claimed in claim 35, wherein said control procedure for matching the components in said list includes a set of control steps for finding the components in a specified bin which satisfy a specified assembly description.

49. The knowledge system as claimed in claim 48, wherein said control steps for finding the components in a specified bin are responsive to specified assembly descriptions including Boolean functions of part names, names of assemblies and part types.

50. A knowledge system comprising a computer having memory storing

a knowledge base including knowledge about a set of related elements, and

an initial list of certain ones of said elements,

said knowledge about said set of related elements including

a predefined declaration of hierarchical decomposition of said set of related elements into separately defined subsets of said elements, some of said separately defined subsets being composed of other of

said separately defined subsets having fewer elements, and

for said separately defined subsets of said elements, respective predefined functions of the respective elements in the subsets, the domain of each function thereby being its respective subset, such function having a predefined result obtained when the function is applied responsive to the specified elements within its domain,

said memory including a working configuration portion for storing lists of said elements,

said memory storing a predefined matching procedure for matching a specified list of certain ones of said elements to a specified one of said subsets to thereby obtain a list of matching elements which are the elements in said specified list which are also elements of said specified subset, said list of matching elements being stored in said working configuration portion of said memory, and said memory further storing a predefined control procedure for

successively applying said matching procedure to match said initial list to said subsets of elements composing said set of related elements, and for each of said subsets of elements composed of said subsets of fewer elements applying said matching procedure to match the previously obtained respective list of matching elements to each of said subsets of fewer elements composing said subset of elements to thereby obtain lists of matching elements representing the configuration of said initial list according to said hierarchical decomposition, and

applying said functions to the respective lists of matching elements within the domains of the respective functions and obtained by matching to the respective subsets, and

said computer including means for executing said control procedure to thereby obtain said matching lists of elements representing a configuration of said initial list according to said hierarchical decomposition, and for applying said functions to the respective matching lists within the domains of the respective functions.

51. The knowledge system as claimed in claim **50**, wherein said functions include functions responsive to whether specified elements of the initial list are included within said respective lists of matching elements within the domains of the respective functions.

52. The knowledge system as claimed in claim **50**, wherein the knowledge base includes predetermined changes to the composition of said lists of matching elements stored in said working configuration portion of said memory, and wherein said functions include respective Boolean condition functions for indicating said changes, and wherein the control procedure includes control steps for executing the indicated changes to thereby conditionally change said lists of matching elements.

53. The knowledge system as claimed in claim **52**, further comprising an explanation facility for generating an ordered list of the conditions for which changes were indicated and the changes which were executed, and transmitting the ordered list to a user.

54. The knowledge system as claimed in claim **50** wherein said subsets correspond to hierarchical functions collectively performed by the combinations of their respective elements, and the definitions of said subsets explicitly stage their respective functions.

55. The knowledge system as claimed in claim **54**, wherein said respective functions of the respective elements in the subsets include at least one function defining a variable, and at least one of said task blocks includes a statement specifying when the function is applied to determine a value for its respective variable.

56. The knowledge system as claimed in claim **50**, wherein said declaration of hierarchical decomposition defines classes of subsets of said elements, and said control procedure includes configuring control steps for creating specified instances of said classes of subsets and adding corresponding elements of said initial list to the instances of said subsets.

57. The knowledge system as claimed in claim **56**, wherein said control procedure includes a set of control steps for recursively applying said configuring control steps.

58. The knowledge system as claimed in claim **57**, wherein said set of control steps for recursively applying starts recursive application at the uppermost level of the hierarchical decomposition.

59. The knowledge system as claimed in claim **50**, wherein said knowledge base includes a plurality of task blocks specifying in part the operation of said control procedure, and said subsets are associated with particular ones of said task blocks, and said task blocks specify operations to perform with respect to their associated subsets.

60. The knowledge system as claimed in claim **59**, wherein said task blocks include imperative language statements including separate statements for creating and storing in said working configuration portion of memory the list of matching elements obtained by matching to a specified one of said subsets, and for applying a specified set of functions to the respective lists of matching elements within the domains of said functions.

61. A knowledge system for checking a production request for a flexibly-assembled product, said request including a list of part names and respective quantities, said knowledge system comprising a computer having a memory storing a knowledge base, said knowledge base including

a declaration of a hierarchy of funtional assemblies including parent-offspring relationships between respective assemblies and sub-assemblies,

a parts catalog including a predefined set of parts and identification of the functional assemblies which the respective parts compose, and

a set of conditions applicable to respective assemblies for indicating whether parts configured into the respective assemblies are compatible,

said memory also including

a working configuration portion of said memory subdivided into respective bins for receiving the names of parts configured into the respective assemblies, and

a control procedure for

obtaining said request,

configuring the part names in the request into at least one parent assembly by referencing the parts to the parts catalog to determine whether the parts compose the parent assembly and when they compose the parent assembly adding the respective part names to the parent bin,

configuring the part names in the request to respective offspring assemblies by referencing the parts in the respective parent bins to the parts catalog

to determine whether the parts comprise the offspring assemblies and when they comprise the respective offspring assemblies adding the respective part names to the respective offspring bins, and

applying said conditions to the part names in the respective assemblies, and said computer including means for executing said control procedure to obtain said request, configure the part names in said request to respective offspring assemblies, and apply said conditions to the part names in the respective assemblies to thereby check whether the request includes compatible parts for all of the assemblies in the product.

62. The knowledge system as claimed in claim 61 wherein the knowledge base further comprises task blocks for respective ones of said assemblies, said task blocks including imperative language statements for specifying control procedure steps responsive to the parts in the bins for the respective assemblies.

63. The knowledge system as claimed in claim 62 wherein the knowledge base further comprises a separate set of expansion rules specifying the composition of expandable parts in terms of sub-parts, and wherein the task blocks include imperative language statements for adding the sub-parts of expandable parts in the bins to the respective bins.

64. The knowledge system as claimed in claim 61, wherein said knowledge base further comprises a set of actions specifying changes to the set of parts configured into at least some of said assemblies in response to the respective conditions for said assemblies, and wherein said control procedure includes a set of control steps for changing the sets of parts configured into the respective assemblies in the fashion indicated by said actions in response to whether the respective conditions are satisfied when the respective conditions are applied.

65. The knowledge system as claimed in claim 64 wherein the control procedure includes a set of control steps for generating and transmitting to a user a list of said changes made to the set of parts configured into the respective assemblies in the fashion indicated by said actions and an explanation of reasons why the changes were made.

66. The knowledge system as claimed in claim 64 wherein some of said actions specify that certain additional information is to be requested from a user and that certain changes are to be conditionally made in response to information received from the user.

67. A knowledge system for designing a product including assemblies of predetermined parts, said knowledge system comprising a computer having a portion of memory storing a knowledge base and a portion of memory subdivided into respective bins for receiving selected names of said parts for comprising the respective assemblies,

said knowledge base including a control procedure for

obtaining a set of assembly constraints including conditions applicable to respective assemblies for indicating whether parts in the respective bins have desired properties and respective actions indicating predetermined changes to the sets of parts in the respective bins for obtaining said desired properties in response to whether the respective conditions are satisfied, and

applying said conditions of said assembly constraints to the respective assemblies and conditionally executing the respective changes in response to whether the conditions are satisfied,

wherein said knowledge base further includes a hierarchy defining at least some of said assemblies as offspring sub-assemblies of respective parent assemblies, and wherein said assembly constraints associated with said parent assemblies have conditions referencing the conditions of parts in the respective offspring bins, and

said computer including means for executing said control procedure to thereby obtain said set of assembly constraints, apply said assembly constraints, and conditionally execute said respective changes to thereby obtain a design for said product, said design being indicated by the list of names of the parts in the bins after said conditions are applied and said changes are conditionally executed.

68. The knowledge system as claimed in claim 67 wherein the constraints of said offspring assemblies reference the conditions of parts in the respective parent bins via respective bin variables having respective scopes including the respective parent assembly conditions and the respective offspring assembly conditions.

69. The knowledge system as claimed in claim 67 wherein the knowledge base further comprises task blocks for respective ones of said assemblies, said task blocks including imperative language statements for specifying control procedure steps responsive to the parts in the bins for the respective assemblies.

70. The knowledge system as claimed in claim 67 wherein the control procedure includes a set of control steps for generating and transmitting to a user an ordered list of said changes in the sequence that the changes are executed, and an explanation of why said changes were made.

* * * * *

United States Patent [19]

Carlson, Jr. et al.

[11] **Patent Number: 4,646,238**

[45] **Date of Patent: Feb. 24, 1987**

[54] **MATERIAL REQUIREMENTS PLANNING SYSTEM AND PROCEDURES FOR USE IN PROCESS INDUSTRIES**

[75] Inventors: **William H. Carlson, Jr.**, Milton, Mass.; **Paul H. Shafer**, San Jose, Calif.

[73] Assignee: **Analog Devices, Inc.**, Norwood, Mass.

[21] Appl. No.: **665,036**

[22] Filed: **Oct. 26, 1984**

[51] **Int. Cl.**[4] **G06F 15/00**

[52] **U.S. Cl.** **364/403**; 364/468

[58] **Field of Search** 364/403, 468

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,459,663 7/1984 Dye 364/200

Primary Examiner—Jerry Smith
Assistant Examiner—Allen MacDonald
Attorney, Agent, or Firm—Parmelee, Bollinger & Bramblett

[57] **ABSTRACT**

A system for controlling the flow of semiconductor products and their components through a production facility including assembly and final testing of a large number of different products with multiple product grades. The system stores information on the demand and inventory of all product grades together with grade distribution data giving the yield of all co-products of a product family from testing the common component of that family. The system includes means for calculating the number of common components to be tested to cover the requirements for all co-products of a family. The distribution data also gives the yield of all by-products of a test for particular products, and the system operates to adjust the inventory status for other products corresponding to such by-products, to avoid excess production of such other products in meeting projected demand. For cases where the data indicate for any product family a surplus of a higher grade and a shortage of a lower grade, the system includes means to decide whether to downgrade a portion of the higher grade to the lower grade; the projected inventory status of each grade is adjusted accordingly.

10 Claims, 7 Drawing Figures

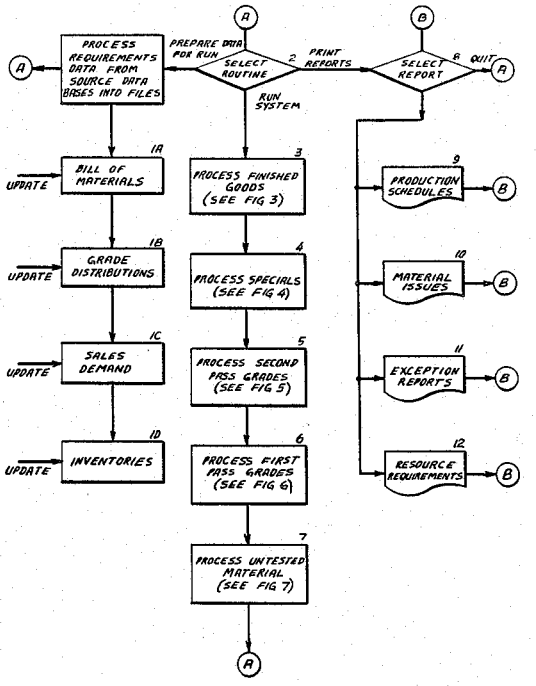


Fig. 1.

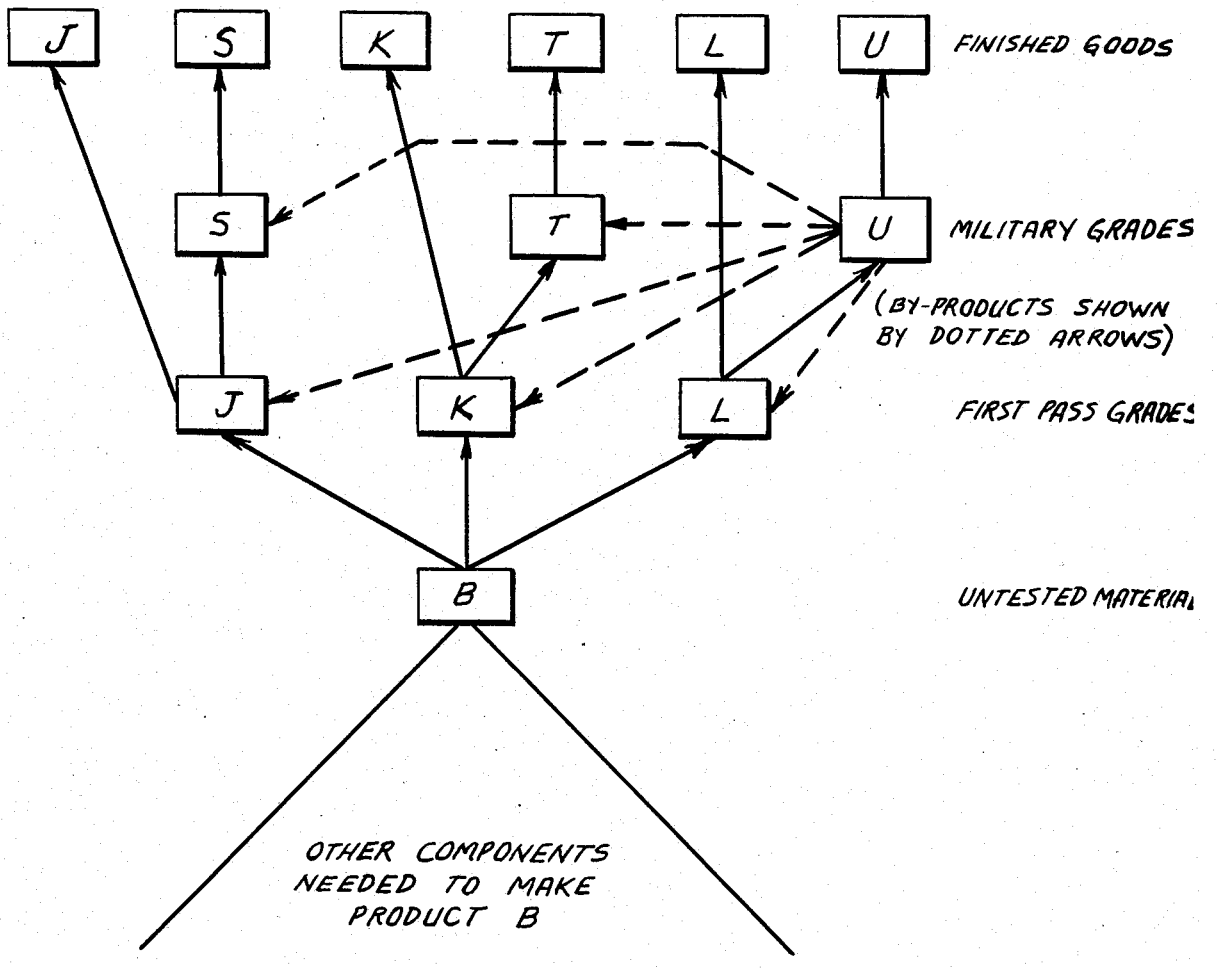
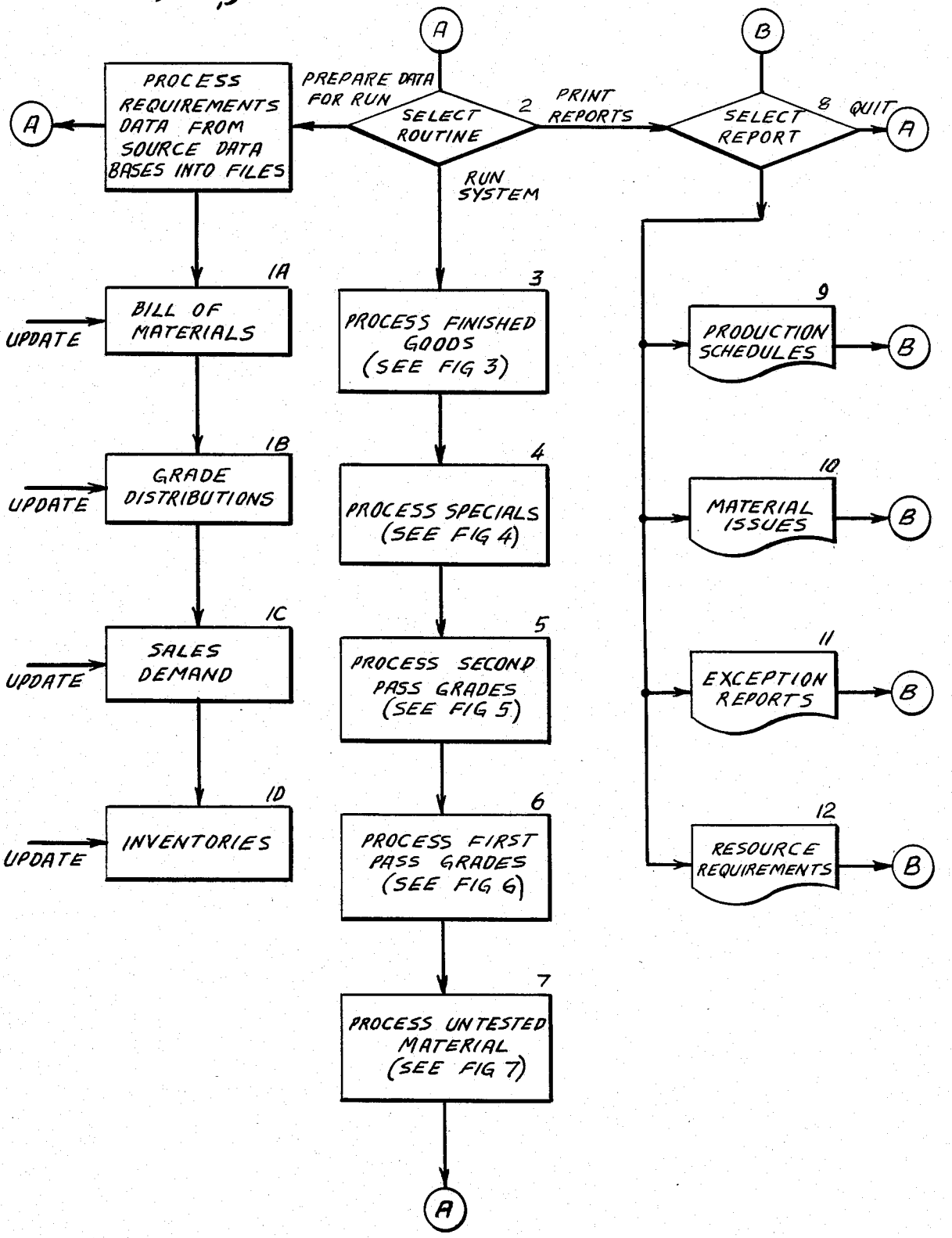

J
S
K
T
L
U
FINISHED GOODS
S
T
U
MILITARY GRADES
(BY-PRODUCTS SHOWN BY DOTTED ARROWS)
J
K
L
FIRST PASS GRADES
B
UNTESTED MATERIAL
OTHER COMPONENTS NEEDED TO MAKE PRODUCT B

Fig. 2.


A
B
PROCESS REQUIREMENTS DATA FROM SOURCE DATA BASES INTO FILES
PREPARE DATA FOR RUN
SELECT ROUTINE
2
PRINT REPORTS
SELECT REPORT
8
QUIT
RUN SYSTEM
1A
BILL OF MATERIALS
UPDATE
1B
GRADE DISTRIBUTIONS
UPDATE
1C
SALES DEMAND
UPDATE
1D
INVENTORIES
UPDATE
3
PROCESS FINISHED GOODS (SEE FIG 3)
4
PROCESS SPECIALS (SEE FIG 4)
5
PROCESS SECOND PASS GRADES (SEE FIG 5)
6
PROCESS FIRST PASS GRADES (SEE FIG 6)
7
PROCESS UNTESTED MATERIAL (SEE FIG 7)
9
PRODUCTION SCHEDULES
10
MATERIAL ISSUES
11
EXCEPTION REPORTS
12
RESOURCE REQUIREMENTS

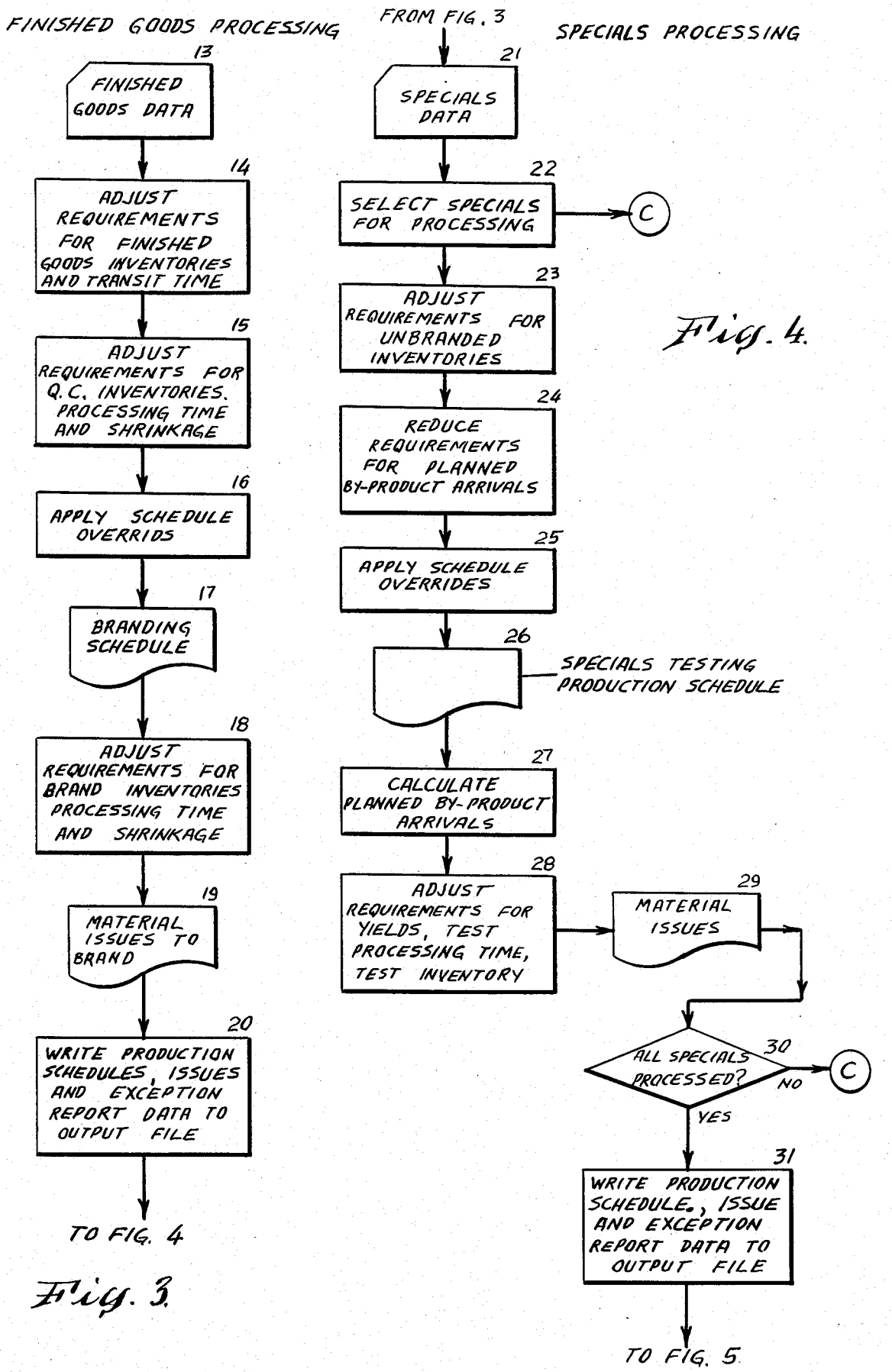
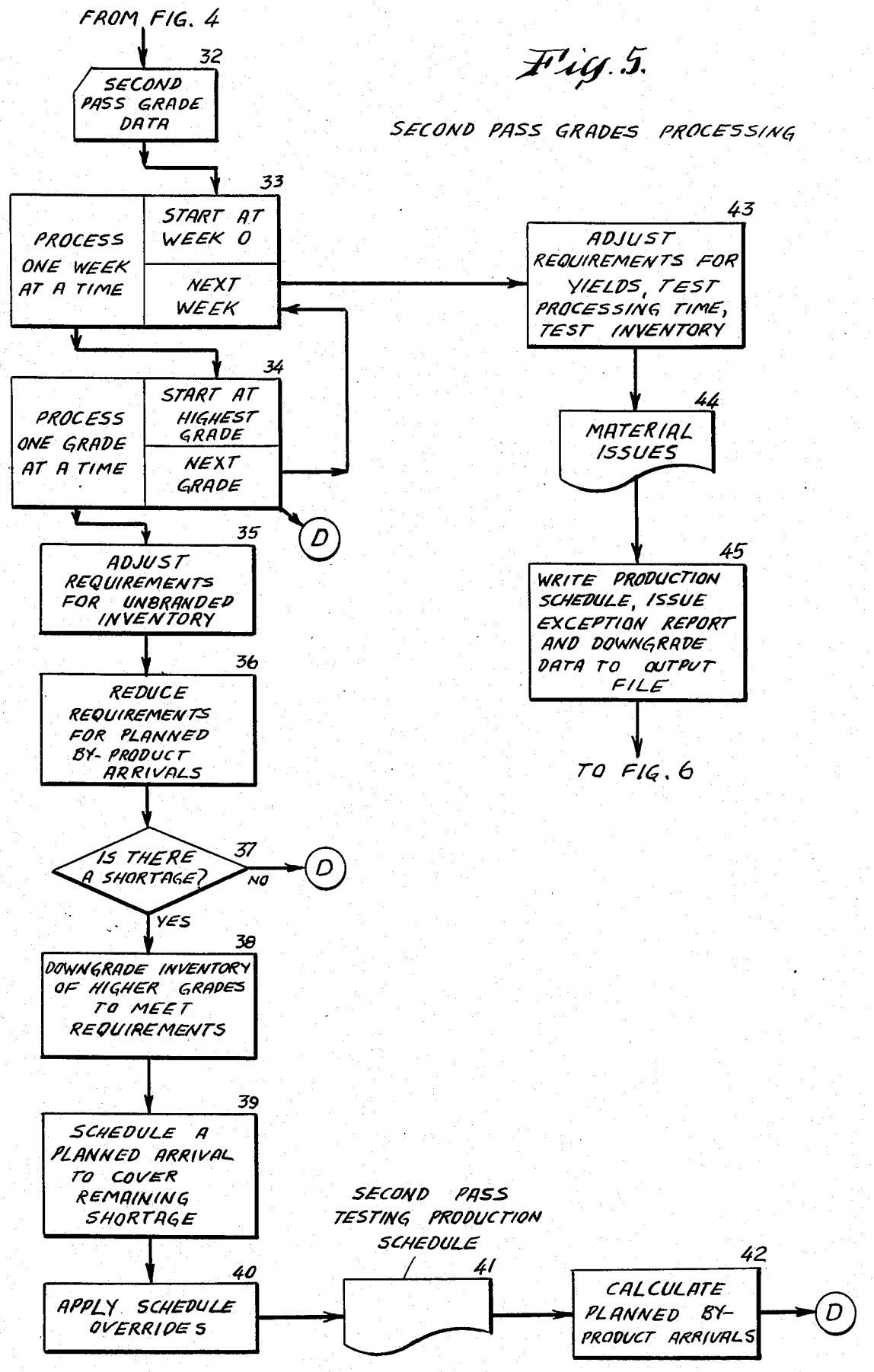

FINISHED GOODS PROCESSING
13
FINISHED GOODS DATA
14
ADJUST REQUIREMENTS FOR FINISHED GOODS INVENTORIES AND TRANSIT TIME
15
ADJUST REQUIREMENTS FOR Q.C. INVENTORIES, PROCESSING TIME AND SHRINKAGE
16
APPLY SCHEDULE OVERRIDS
17
BRANDING SCHEDULE
18
ADJUST REQUIREMENTS FOR BRAND INVENTORIES PROCESSING TIME AND SHRINKAGE
19
MATERIAL ISSUES TO BRAND
20
WRITE PRODUCTION SCHEDULES, ISSUES AND EXCEPTION REPORT DATA TO OUTPUT FILE
TO FIG. 4
FROM FIG. 3
SPECIALS PROCESSING
21
SPECIALS DATA
22
SELECT SPECIALS FOR PROCESSING
C
23
ADJUST REQUIREMENTS FOR UNBRANDED INVENTORIES
24
REDUCE REQUIREMENTS FOR PLANNED BY-PRODUCT ARRIVALS
25
APPLY SCHEDULE OVERRIDES
26
SPECIALS TESTING PRODUCTION SCHEDULE
27
CALCULATE PLANNED BY-PRODUCT ARRIVALS
28
ADJUST REQUIREMENTS FOR YIELDS, TEST PROCESSING TIME, TEST INVENTORY
29
MATERIAL ISSUES
30
ALL SPECIALS PROCESSED?
NO
C
YES
31
WRITE PRODUCTION SCHEDULE., ISSUE AND EXCEPTION REPORT DATA TO OUTPUT FILE
TO FIG. 5

Fig. 3.

Fig. 4.

Fig. 5.

SECOND PASS GRADES PROCESSING



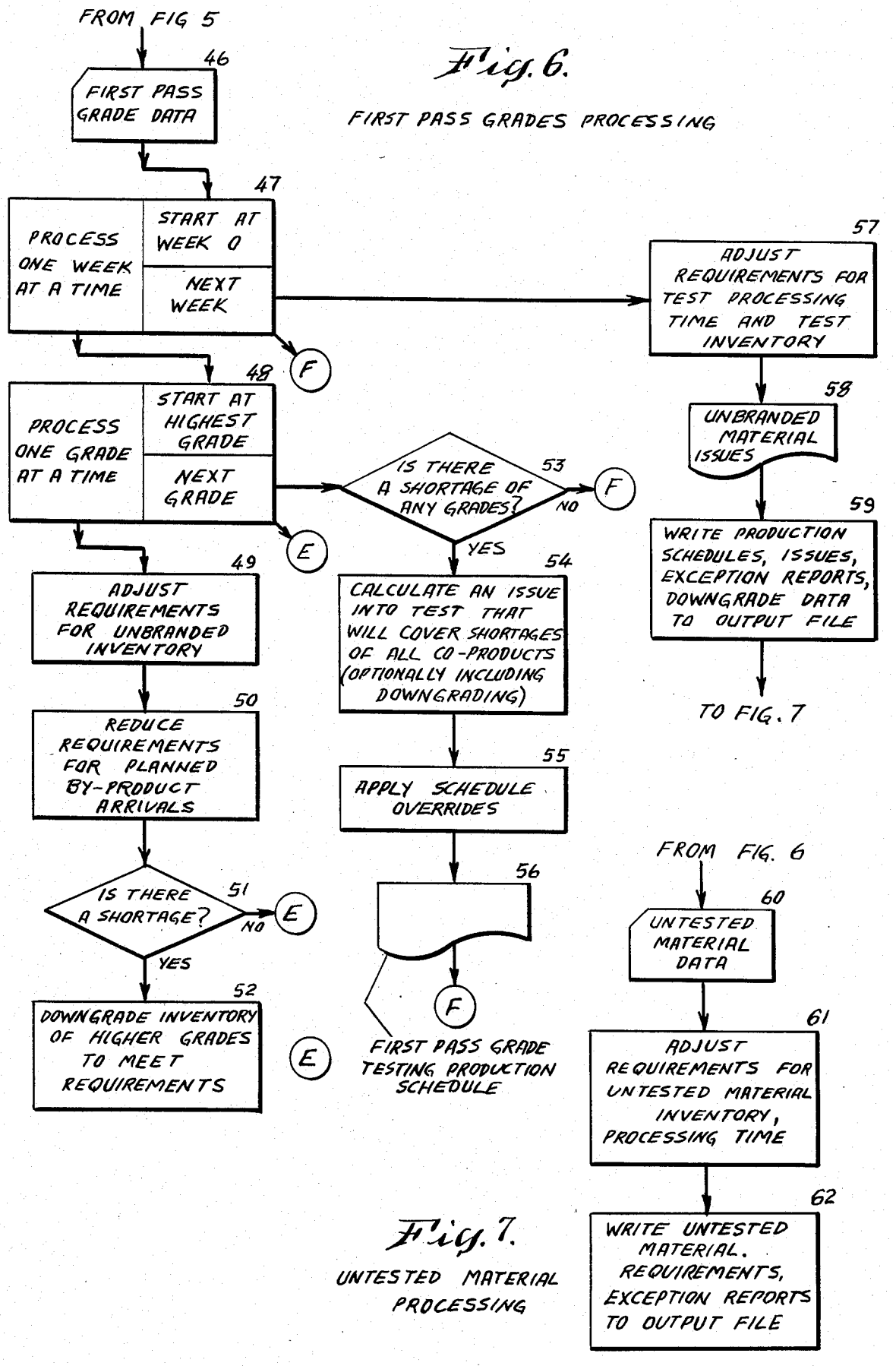


Fig. 6.

FIRST PASS GRADES PROCESSING

Fig. 7.

UNTESTED MATERIAL PROCESSING

MATERIAL REQUIREMENTS PLANNING SYSTEM AND PROCEDURES FOR USE IN PROCESS INDUSTRIES

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to a computerized system for planning the testing of products as part of a manufacturing process. The invention is illustrated for use in planning the testing of semiconductors in the form of integrated circuit (IC) chips.

Semiconductor chips require testing and grading against specifications before they can be sold. The testing of any lot of product will also sort the functional units into separate grades depending on their electrical characteristics. More rigorous or selective testing can then be performed on these grades to produce products which are guaranteed to perform in the more strenuous environment demanded by military or aerospace applications. In addition, certain customers may require special tests to be performed on the product for their specific use.

Production of semiconductors begins in wafer fabrication during which the electronic circuits are grown on thin wafers of silicon in a series of alternating photographic and deposition processes which create hundreds or thousands of identical circuits on each wafer. This process can take from two to upwards to ten weeks depending on the complexity of the process and the number of "layers" involved.

The effectiveness of the wafer fabrication process is dependent on many environmental factors over which the manufacturer has varying degrees of control but which have a definite effect on the functionality and electrical characteristics of the resulting product. In order to eliminate further expensive processing of non-functional circuits, the wafers are probed at this point—i.e. each circuit is powered up, tested and the failures are marked with ink to be later discarded.

After probing, the individual dice are separated, inspected for visual defects and mounted in the individual packages which allow them to communicate with the real world. These packages come in many shapes and sizes from two-lead metal cans to upwards of sixty-four lead dual in-line pin ceramic or plastic packages. Wires finer than a human hair connect the terminals on the dice to the leads on the packages which are then sealed to protect the chip from environmental and mechanical damage.

After this assembly process is complete, the units are tested and graded according to their electrical performance against the range of specifications. The simplest grades are measured at room temperature and are known as commercial. The more difficult grades are those for military and aerospace applications. These must pass strict government standards after being subjected to extreme environmental stress, including continual operation for a week under high temperatures. In addition, the critical characteristics of the higher grade products must not drift out of specification over the whole operating temperature range.

The testing process can be considered as a series of hurdles which the product must pass. Product which fails can be saved to retest against a different set of hurdles, sold as the highest grade for which it has qualified or conceivably sold as a lower grade if no other demand for the product exists.

The process is made especially complex by the large number of different types and grades of product. For example, in one commercial plant, 80 wafer types are assembled into 160 different package configurations which yield over 2500 different end products after testing. The simplest product has 2 grades while the most complex had 88 different grades of standard catalog items and customer specials.

The task for production control is to identify and schedule the correct sequence of tests and an adequate supply of untested material coming out of assembly to support the demand for all of the items. The goal is to accomplish this with minimum inventories and optimal utilization of scarce and expensive computerized test equipment whil maintaining timely and accurate deliveries.

2. Description of the Prior Art

The standard prior art technique for planning the type of production used for semiconductor chips—that is, the technique for specifically determining the requirements for production of the source materials needed in a higher level product—is called Material Requirements Planning (MRP). Such planning is carried out by a computer operating under software control. The computer computations typically are based on a Bill of Materials which defines the source components of any product, the sub-components of such components, and so forth down to the lowest level materials—usually raw materials.

Briefly, Material Requirements Planning performs its planning function by considering the demand for an item and subtracting out any inventories to determine additional production required. This net requirement is then "exploded", that is, multiplied by the appropriate quantities of each component and advanced by the time needed to produce the item. The result is the demand for each component. The process is repeated for each component in turn until the lowest level is reached.

SUMMARY OF THE INVENTION

In the testing process described herein, all untested products are given a first pass test to eliminate non-functional units and produce a preliminary grading. The output from the test becomes input to the military or second pass test. Optionally, the output from either the first pass or second pass test may become input to a test for so-called customer specials. Any product which passes any of the tests can be branded with its identifying grade and sold as a finished product.

The procedure is controlled by a software program operating with a data base defining the proper components for finished goods, customer specials (if used), second pass grades and first pass grades. The procedure resolves three issues in the testing environment which standard MRP does not address. These are: by-product planning, co-product planning, and downgrading, which may be described briefly as follows:

BY-PRODUCT PLANNING

When testing for military grades, some of the product will pass the test. The product which fails is still good product, however, and is called a by-product. Normally this by-product will be another item already in the Bill of Materials and most often the component item itself. Standard MRP does not account for the production of these by-products nor does it consider them as available

inventory to be netted out of additional production requirements.

CO-PRODUCT PLANNTING

A special case occurs for first pass grades where two or more grades are simultaneously produced from the same test, and are referred to as co-products. In effect these grades are by-products of each other. Standard MRP which calculates items one at a time sequentially cannot simultaneously plan the component quantity which must be tested to meet the demand for all of these grades together.

DOWNGRADING

Because grading is a natural phenomenon independent of demand, its distribution does not necessarily match the distribution of demand. This may result in surplus inventories of higher grades and shortages of lower grades. There are circumstances where it makes economic sense to downgrade these higher grades to cover the shortages rather than to produce more of the lower grades. Standard MRP always covers shortages by additional production. In the system described herein, additional processing is employed to determine which products, when and in what quantities should be downgraded.

A general purpose computer system operating under software control as described herein resolves the above three issues through logic procedures inserted into standard MRP software.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a simplified diagrammatic representation of hierarchical levels to which items can be assigned for testing;

FIG. 2 is a flow diagram giving an overview of the entire MRP system as modified to incorporate the present invention;

FIG. 3 is a flow diagram showing the logic of finished goods scheduling represented by functional block 3 of FIG. 2. (Note: the logic is standard prior art MRP and is shown herein only for completeness);

FIG. 4 is a flow diagram showing the logic of customer specials represented by the functional block 4 of FIG. 2;

FIG. 5 is a flow diagram showing the logic of second pass test planning represented by functional block 5 of FIG. 2;

FIG. 6 is a flow diagram showing the logic of first pass test planning represented by functional block 6 of FIG. 2; and

FIG. 7 is a flow diagram showing the logic for planning untested material represented by the functional block 7 of FIG. 2.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention is based on the concept that in some manufacturing industries there are relationships between the products being processed through testing which make it important and advantageous to plan the testing of certain product families together. To that end, the system to be described categorizes products into different levels so that each level can be processed using the distinctive procedures which apply to that level. A multi-level product structure used in the system to be described is as follows:

Level 0—Finished Goods
Level 1—Customer Specials
Level 2—Second Pass Tested Standard Items
Level 3—First Pass Tested Standard Items
Level 4—Untested Product

Defining these in reverse order (i.e. in the direction in which product actually moves), Level 4 contains untested material as it comes from assembly, and Level 3 contains the grade distribution resulting from the first ambient test of newly assembled product. Typically in the semiconductor industry there are three grades—J, K and L, with J the lowest grade and L the highest. These grades may be branded and shipped as their level 0 branded standard commercial counterparts or issued for higher grade testing.

Level 2 contains the military versions of the commercial grades. The product is stressed, burned in and retested for its drift over a temperature range. An L so tested which retains its electrical characteristics might be called a U. A K might become T and a J and S. The L which fails as a U might still be an L or perhaps have degraded to a T or S, K or J. The exact distribution would. be spelled out on a "lot traveller" accompanying the group of items to be tested and specifying characteristics derived from the Bill of Materials stored in the data base of the computer which controls the flow of product testing.

Level 1 contains customer specials which are produced by additional selective testing of Level 2 or 3 standard products for some customer specific parameters. This level may also contain specials which are made from other specials and thus can be divided into several sublevels. Level 1 can be considered optional, and the processing in Level 1 will not be discussed in detail herein.

Level 0 contains finished goods which are the final branded versions of the products manufactured at one of the lower levels. No additional testing is required to move a product to this level. Due to branding variations, however, several finished products can be made from the same unbranded component.

While this concept of levelling serves to group together products which interact in the planning process and which are planned using the same logic, it also retains the original concept of low level standard MRP code, namely that products are always at a higher level than their components. Level 0 items are always made from Level 1 2 or 3 items. Level 1 items are always made from other Level 1 items or from Level 2 or 3 items. Level 2 is always made from Level 3 which is always made from Level 4.

FIG. 1 presents a simplified exemplary illustration of the type of product-level hierarchy described above (omitting any showing of the "specials" level). It will be seen that the initial group of untested items referred to as B are assembled from other components (not identified in the diagram but listed in the Bill of Materials). A planned first pass test will be expected to produce products graded as J, K, L (identifying prescribed electrical characteristics), in pre-established ratios in accordance with experience data stored in the data base. Rejects will be discarded.

If it is desired to produce a number of items of military grade U, a group of grade L products will be tested. That is, military grade U requires the superior electrical characteristics of commercial grade L, but must be capable of such performance under military environmental conditions. Upon such testing, some of grade L will pass, to be designated grade U. Others will

pass the military, requirements but with lower electrical characteristics, and be designated grades S or T (as indicated by the dotted lines). Still others will fail the military requirements, but still pass the electrical tests to be accepted back as grade L or as grades J or K. The final graded products can be branded and transferred to "finished goods".

In standardizing the product flow through test in such a level-oriented fashion, it is desirable to close out all test lots to a stock room after each level of testing is completed. This improves inventory control and lot traceability, and makes it possible to capture the actual grade distribution of all intermediate tests as inventory transactions which later become the basis for up-dating the grade distribution data base.

The grade distribution data base (which is not used in standard MRP planning) contains the planned yield distribution to all grades resulting from testing a component. If it is desired to plan a test for item SC from component JC, the yield must be known. For example, if 100 units of SC are needed and the yield is 50%, the demand for component JC will be 200. In addition, however, the present system takes into account the yield and identity of the by-products of this test. In this case there might be a 45% yield to JC (components which failed the SC test) while the other 5% are scrap, destroyed in the testing process.

A typical grade distribution data base entry might include the following:

	PRODUCT	COMPONENT	BY-PRODUCTS		
Item	TC	KC	SC	KC	JC
Yield	.40		.10	.30	.10

The grade distribution data base defines for all items in Levels 1, 2 and 3 the yield to that item from its component. Also defined is the identity of all by-products (which may be on the same or lower levels), and the yield to each. The system updates these grade distributions periodically through batch analysis of the historical stockroom grading inventory transactions.

The present system also incorporates a function referred to as downgrading, which will be described in detail hereinbelow. In order to plan downgrading, it is necessary to know the hierarchy of grades, that is, which standard grades within Levels 2 and 3 can be substituted for lower grades with no additional testing. This information also is stored in the grade distribution data base.

Referring now to FIG. 2, it will be seen that the overall system flow diagram breaks into three main routines. The diamond 2 represents the select routine decision, to prepare the data for run, to run the system, or to select a report. The letters in circles represent return points for the system loops.

Block 1 indicates the portion of the system which prepares the raw data for processing including the Bill of Materials (Block 1A), grade distribution data (Block 1B), sales demand (Block 1C) and inventories (Block 1D). This information is stored in data base files and used by the operating part of the system when it is being run to prepare the plan for assembly and testing.

In a standard Bill of Materials used in prior art MRP systems, products are assigned to levels based on their relationship to other products. Finished goods are assigned to the first level (usually called Level 0); their components are assigned to Level 1 etc., down to the lowest level. The general rule is that an item's level number is equal to the highest level number of any item it is used in plus one. Standard MRP processes items by level starting at 0 thus assuring that for any item all demand for it as a component of other items will be present at the time that item is processed.

In the Bill of Materials used in the present system, items are not assigned to relative levels but rather to absolute levels based on their testing classification, namely: Finished Goods, Customer Specials, Second Pass, First Pass and Untested Product. This classification does not violate the standard rule that an item comes after any item of which it is a component, but it does assign to the same level items which are to be planned together because of by-product or co-product relationships. This assignment is also advantageous because it defines exactly the proper sequence for planning the products and makes it possible to match each level or set of products to the appropriate modified logic. Block 1A represents this Bill of Material data base information. A simplified example of the basic type of information contained in the Bill of Materials is represented by the diagram of FIG. 1.

Block 1B represents a series of tables stored in the data base giving the percentage yield to by-product or co-product items which result from testing a component. (A by-product is defined as: a usable unbranded grade that has failed the test for a desired item. A co-product is defined as: one of several related usable unbranded grades classified by the same test.)

Block 1C represents projected sales demand such as that based on marketing sales estimates. Block 1D represents the inventory data. The information of Blocks 1C and 1D would be found in one form or other in any MRP system.

Blocks 9 through 12 represent output reports generated by the system. Such reports would be generated in any standard MRP system.

Blocks 3 through 7 represent the logical processing procedures which are at the heart of the present system. The processing steps for each of these blocks is described in expanded detail in FIGS. 3–7, and will be explained in more detail hereinafter. It may be noted, however, that each block 3–7 represents a single level as defined above. The processing at each of these blocks applies the appropriate logic to the items at its level and all items in a level are finished before the next level is begun. The end result is a projected assembly and test plan for a relatively long time frame typically subdivided into weeks, and providing projections for all products indicating the number of each to be assembled and to be input into the various test levels.

FIG. 3 shows the data processing logic for the finished goods level (Level 0) represented by Block 3 of FIG. 2. This logic is standard MRP logic and is included only for completeness. NO testing, yielding or grading is involved in the brand/final-QC process. Independent demand is introduced at Block 13 as a combination of backlog, forecast and safety stock for all standard and special finished goods. This demand, net of finished goods inventories, becomes dependent demand for those items' unbranded components and all items are processed before proceeding to Level 1.

The resulting requirements are not part of the production schedule for test, but rather a transfer plan which drives the test schedule, serving as the master schedule. This transfer plan is exploded through the test planning logic sequences to produce a testing schedule

for Levels 1, 2 and 3 and a Level 4 requirement for assembly. Test capacity constraints for the test schedule levels and material constraints in assembly can then be identified if necessary.

Blocks **14**, **15** and **18** execute normal inventory netting, lead time adjustments and shrinkage calculations. Block **16** allows the schedules to modify the system generated schedules using a standard industry technique known as a firm planned order. Blocks **17** and **19** create reports which can be printed as normal MRP outputs and Block **20** saves the exploded demand for components which will be part of the input at the lower levels.

FIG. **4** shows the data processing logic for the customer specials level. Block **21** assembles the demand, yield, inventory, and Bill of Materials data for all of the items which have been assigned to this level. Although the processing for this level is optional, depending upon the nature of the industry, it does include process steps which are used at lower levels, and therefore certain parts of the customer specials procedure will be described herein.

Block **22** selects the special products to be processed first. This is necessary because some specials may have other specials as their component or as their by-product. In such a case it is necessary to process the higher level item first and then the component or by-product item. Block **22** first selects all of the specials which are neither components nor by-products of other items for processing at this time. After these items have been completed and eliminated from the list, the next execution of Block **22** will select the next level of components for processing. This process is repeated until all of the specials have been processed.

For those items selected in Block **22** the processing continues in Block **23**. Here the inventory is subtracted from the demand as in normal MRP.

Block **24** introduces part of the new logical processing which deals with by-product planning, specifically that of netting out by-products as an inventory supply. This is necessary at the specials level because the by-products of some specials are other specials. Block **23** subtracted inventory on hand from the first time period's demand. Block **24** subtracts out by-product supplies from the demand in the period when they will be available. (Note: The by-product supply data comes from other portions of the system, as will be described later.) For example, if there is demand of 500 units each for two periods and an on-hand inventory of 50 units and by-product supplies of 100 units each period, the net demand for each period would be calculated as follows:

	Period 1	Period 2
Demand	500	500
Inventory	50	
By-product	100	100
Net demand	350	400

This computation has the effect of covering demand with an alternate supply of material (i.e. the by-product from prior planning calculations) before calling for more production. Standard MRP does not do this.

Blocks **25** and **26** concern firm planned orders and standard MRP reports, which are conventional, and therefore will not be described.

Block **27** introduces the remaining part of the new by-product planning logic—the calculation of the supply of by-products resulting from the tests which are being planned. (This result may be used in Block **24** in a later processing sequence for another item at this level.)

This by-product computation requires two steps. Step one computes the quantity of the component which must be tested to cover the demand. This is done by dividing the net demand by the yield. For example, 100 units of product A with a 50% yield requires the testing of 200 units of component B.

Step two computes the by-products. For example, if the grade distribution data base for product A above shows two by-products B (yield 20%) and C (yield 25%), then block **27** generates a supply of 40 units of B and 50 units of C for the same period as the projected demand of 100 units of A. Note that B can be both a component and a by-product of the same test, but because the component demand is advanced to an earlier period by the length of the testing process the demand for 200 units of B as a component will precede the supply of B as a by-product.

Blocks **28** and **29** continue the normal MRP logic. Block **30** checks to see whether or not there are additional levels of specials to be processed. If so, the program branches back to Block **22** as indicated by the letter C in a circle. If not, it proceeds to Block **31** which saves the exploded demand and any by-product supplies for the second or first pass grades, then passes control to the second level on FIG. **4**. If no specials processing is included in the system, the finished goods processing of FIG. **2** will be followed immediately by that of FIG. **4**.

FIG. **4** specifies the processing of second pass grades. Here, unlike the specials level, all of the second pass grades of the same product family are processed together, one week at a time. This is necessary in order to correctly account for by-product supplies to a lower grade from a higher grade and to allow for downgrading. This simultaneous processing of multiple products is a significant departure from standard MRP and highly important in the functioning of the new system.

Block **32** functions in the usual way to assemble the data for the products at this level. This data comes from the data files of Block **1** (FIG. **2**) and comprises inventories, projected demand, grade distribution and the Bill of Materials. Blocks **33** and **34** control the sequence of processing one week at a time and, within each week, one grade at a time, proceeding from the highest to the lowest. Block **35** nets out the inventory using standard MRP logic. Block **36** represents the new logical processing used to net out by-product arrivals which may either come from the specials level or from higher second-pass grades. The calculations are exactly the same as described above with respect to Block **24** and therefore will not be repeated here. Block **37** branches to the next lower grade if there is no requirement for the current one.

Block **38** introduces the logical processing for the downgrading procedure discussed previously. Because grading is a natural phenomenon independent of demand its distribution does not necesarily match the distribution of demand. This may result in surplus inventories of higher grades and shortages of lower grades. There are circumstances where it makes economic sense to downgrade these higher grades to cover the shortages rather than produce more of the lower grades. The processing procedures in Block **38** decide whether or not to downgrade any surplus inventory of higher grades in the same family to cover the require-

ment of the current grade. This downgrading is a very significant departure from standard MRP.

Any downgrading decision involves three questions, asked from the point of view of the receiving item, i.e. the lower grade:

1. Is there a shortage—if not, no downgrading is necessary.

2. Is there a higher grade with a surplus—if not, no downgrading is possible.

3. Should the higher grade be downgraded—this decision involves a comparison of the relative demands and yields of the two grades.

For example, in the case of three grades L, K and J where L is the highest grade and J the lowest, the procedure first determines whether there is a shortage of K and if so, whether or not to downgrade L. Then a shortage of J is determined and, if so, a decision is made whether or not to downgrade first K and then L.

1. A shortage exists if the demand for a grade in the current period is larger than the total of the beginning inventory and the by-product supply.

2. A surplus exists if the demand for the grade in the current period is smaller than the total of the beginning inventory and the by-product supply.

3. The formula for determining whether to downgrade has three parts:

(a) For the higher (donor) grade compute:

$$\sum_{I=W}^{N} (D_I - B_I) \times (N - I + 1)/Y$$

where

I is a counter beginning with the current week W

N is the total number of periods (weeks) in the plan

D_I is the demand for the grade in the period

B_I is the by-product supply in the period

Y is the yield to the grade (N−I+1) is a factor which gives greater weight to near term demand than future demand.

It can be seen that the outcome from this formula will be increased by large demand, near term demand, small by-product supply, and low yield. The outcome will be decreased by small demand, far future demand, large by-product supply and high yield.

(b) Compute the same formula for the lower (receiver) grade.

(c) Compare the results. If the result for the donor is lower than for the receiver, the higher grade should be downgraded.

For example: Period (I)		3	4	5	6	7	8 (N)
L	Beginning Inventory	200					
Y = .3	Demand	50	150	100	300	500	100
	By-Product Supply	100	100	100	100	100	100
K	Beginning Inventory	0					
Y = .4	Demand	600	500	800	300	900	700
	By-Product Supply	50	100	50	40	200	300

-continued

For example: Period (I)	3	4	5	6	7	8 (N)
(N − I + 1)	6	5	4	3	2	1

1. K has a shortage of 550 in week 3 (600 − 50)
2. L has a surplus of 250 in week 3 (200 + 100 − 50)
3. (a) Donor formula = (6 × −50) + (5 × 50) + (4 × 0) + (3 × 200) + (2 × 400) + (1 × 0)/.3 4500
(b) Receiver formula = (6 × −550) + (5 × 400) + (4 × 750) + (3 × 260) + (2 × 700) + (1 × 300)/.4 27200
(c) Since (b) is larger than (a) the surplus of 250 L should be downgraded to K leaving 300 more K to make from new testing.

Once the decision to downgrade has been made, then the quantity to downgrade is the surplus of the higher grade or the net requirement of the lower grade, whichever is less. Downgrading should only take place when the demand for the higher grade is low or far enough in the future (relative to the demand for the lower grade), and the yield (the ability to produce more of the higher grade later) is reasonably high.

The net effect of downgrading is to minimize the cost of carrying the excess inventory and the cost of current testing.

Block **39** calculates the planned production to cover the balance of the current grade (i.e. after the downgrading transfer) using normal MRP logic. Block **40** applies firm planned orders and Block **41** prepares the schedule report, as discussed previously.

Block **42** calculates by-product supplies to lower second-pass grades or to first-pass grades, using the same calculations as described above with respect to Block **27**. Blocks **43**, **44** and **45** perform the same standard MRP functions as Blocks **28**, **29** and **31**, referred to above. All of the components are on the first-pass level.

After all grades have been completed for one week, the processing proceeds to the next week. After all families on the second-pass level have been completed, the processing proceeds to the first pass level of FIG. **6**.

The first-pass processing is very similar to the second-pass processing. The sequence proceeds from week to week, and from high grade to low grade, governed by Blocks **47** and **48**. This sequence provides the basis for the co-product planning at the first-pass level. Blocks **49–52** represent logical processing steps identical to those of Blocks **35–39** at the second-pass level.

Once the net requirements for all grades in the family have been calculated, including the projection of inventory downgrades as described above, co-product processing is carried out to determine the proper quantity of the component common to all grades which will yield enough of each grade to cover all of their requirements. That is, it is necessary to compute the issue quantity of the common component which will cover the yield to the most limiting of the grades in the set. Most simply this means dividing the net demand for each grade by its yield and issuing the largest of the three resulting quantities.

Set forth below is an example of such co-product planning using three grades (L, K and J) where L is the highest grade.

Item	Demand	Yield	Test Qty.	Output
L	100	.20	500	200 L
K	300	.30	1000*	300 K
J	450	.50	900	500 J

The demand for the three grades are 100, 300 and 450 respectively, and their yields are 20, 30 and 50%. The processing divides the net demand for each grade by its yield and issues an order for the largest of the three resulting quantities. In this case, it would calculate an issue of 1000 which will just cover the requirement for K, and result in an extra 100 L's and an extra 50 J's.

It may be advantageous also to employ downgrading in co-product planning. By using surplus yield of higher grades to cover lower grades, all the demand can be satisfiled while testing fewer parts, using less capacity and generating less inventory.

Set forth below is a table giving an example where downgrading is applied fully to the same three products with the same initial demand and yield as listed above.

Item	Demand	Yield	Test Qty.	Output	Downgrading
L	100	.20	500	170 L	−70
K	400	.50	800	255 K	+45
J	850	1.00	850*	425 J	+25

In this case, however, the computation takes into account cumulative demand and cumulative yield for each successive grade. For example, the demand for K is cumulative of L and K (100+300=400) with yield of L and K combined (0.20+0.30=0.50) before dividing to determine the limiting test quantity. While 500 will cover the L alone and 800 will cover L and K combined, 850 are needed to cover all three grades. This is the amount to be tested—less than the 1000 needed in the no-downgrade option. This yields an extra 70 L's, and 45 of them will be used to cover the K deficit (300 were needed) and 25 to cover the J deficit (450 were needed).

Alternatively, the downgrading algorithm (see above) can be used to decide which grades should be downgraded to which. These downgrade combinations can then be planned together by adding their demands and yields and computing the issue quantity which covers the combination. After this downgrading adjustment has been made, the planned issue quantity is the largest needed to support any of the combinations or uncombined grades in the set. It will be found that use of the downgrading algorithm for co-product planning will produce a result intermediate the no-downgrading case (1000 units) and the full-downgrading case (850) units, which may be a more satisfactory decision from a practical point of view.

The resulting yield to each grade and the appropriate downgrade transactions then are computed, followed by the ending balances which will roll into the following period.

Block **55** applies firm planned orders as described above. Blocks **56–59** are the same as Blocks **41, 43 44** and **45** at the second-pass level.

After all first-pass families have been processed for all weeks, there results a test schedule for Level 3 items and dependent demand for Level 4, i.e. the components of untested material. This net requirement is passed on to the standard MRP logic at the lower levels of the product structure, as indicated in FIG. 7.

To summarize, processing procedures of the disclosed system are carried out at five levels of the product structure as set forth below:

0. FINISHED BRANDED DEVICES

This module first calculates a schedule for receipts into finished goods by netting out existing on-hand inventories against demand (backlog and forecast) and inventory targets. Against this requirement it applies inventories in factory bonded stock and final QC and offsets by lead time to compute the schedule for branding and transfer to final QC. This schedule is saved and printed and can be modified with "firm planned orders" which affect the schedule at lower levels and can be saved to apply to subsequent runs of the system.

These requirements net of in process inventories at brand are captured as issues from unbranded stock of the item's component and will be used as demand for that component at the appropriate level (1, 2 or 3).

Exception reports highlighting current and projected inventory imbalances are generated at this and all subsequent levels.

1. PRODUCTS SPECIALLY SELECTED FROM OTHER PRODUCTS

In this level, items and their components can both exist on the same level of the product structure. Items are processed in successive batches so that the requirements for an item's component are generated before that component is processed.

The processing consists of the normal calculations, i.e. demand (for the Level 0 or other Level 1 item) net of existing unbranded inventory becomes the scheduled test completion for the item. That requirement, divided by the yield, offset by a lead time, and net of existing work in process inventory is the demand for the item's component which is another Level 1 or a Level 2 or 3 item.

This level (a) considers by-product arrivals of the item from other Levell tests before computing additional tests for the item and, (b) calculates by-products to other products (on level 1, 2 or 3) which will result from the test for this item.

2. SECOND PASS TESTED STANDARD GRADES

This level employs the by-product computations to consider arrivals from Level 1 and generates by-product calculations for other Level 2 and 3 items.

This level also employs downgrading to apply excess inventories of higher grades to demand for lower grades of the same product family before scheduling new testing of the lower grade. This level therefore is processed one week at a time and within each week from the highest to the lowest grades.

3. FIRST PASS STANDARD GRADES

This level includes consideration of by-product arrivals from Levels 1 and 2 tests and inventory downgrading logic introduced in Level 2.

In addition, it calculates the correct quantity of the untested component which will yield the proper quantities of each of the several first pass grades to meet their demands. In so doing it may determine when and in what amounts higher grades yielded from the tests should be downgraded to meet the demand for lower grades thus reducing the total requirement for testing.

4. UNTESTED MATERIAL

This level uses standard MRP logic to determine the required receipts of untested material to meet the Level 3 requirements.

Although a preferred embodiment of the invention has been disclosed herein in detail, it is to be understood that this is for the purpose of illustrating the invention, and should not be construed as necessarily limiting the invention since those of skill in this art can readily make various changes and modifications thereto without departing from the scope of the invention as reflected in the claims hereof.

What is claimed is:

1. In combination in a system for controlling the flow of a plurality of products and components thereof through the assembly and testing procedures of a manufacturing operation comprising plural sequential process steps, including the development of an assembly and test schedule for the successive periods of a forward planning time frame, and wherein the products are arranged in families comprising different grades meeting respective performance specifications corresponding to different quality ratings, said system including data file means for storing information representing the inventory status of each product grade and the expected demand for each product grade during each of said periods, said system being of the type which is arranged to make calculations with respect to such stored information so as to determine the net requirements for each product grade during each of said successive time periods;

that improvement to such system wherein said data file means stores grade distribution information giving the co-product yield for each grade of each product family from testing the common component of the product family; and

said system further comprising means to calculate, based on said grade distribution information, the number of such common components to be tested for each of said periods in order to meet the net requirements projected for the co-product grades for those periods.

2. A system as in claim 1, wherein said data file means further includes stored by-product information giving for at least certain of the products the identities and percentages of items which will be produced as by-products when testing for any particular one of said certain products selected;

said system further including means to calculate and to store for a projected test of such particular product the number of each such by-products to be produced; and

means responsive to such stored by-product number data for updating the effective inventory of a corresponding item when the net requirements for that item are being developed subsequently.

3. A system as in claim 1, including means operable in making said calculations with respect to said stored information, for determining the net requirements for at least two grades of the same product family, to downgrade a surplus portion of the higher of said grades to the lower grade so as to minimize the need for additional assembly or testing to produce the lower grade.

4. A system as in claim 3, wherein said downgrading decision is made by taking into account the relative demands and yields of the two grades.

5. A system as in claim 1, including means operable, in calculating said number of common components, to decide whether to downgrade a portion of a higher grade to cover the requirements for a lower grade of the product family.

6. A system as in claim 5, wherein the downgrading desicion is based on the relative demands and yields of the two grades.

7. In combination in a system for controlling the flow of a plurality of products and components thereof through testing procedures of a manufacturing operation comprising plural sequential process steps, said system including means for developing an assembly and test schedule setting forth the planning goals for each of the products for the successive periods of a forward planning time frame, each of the products being required to meet respective performance specifications to be designated as such product, said system including data file means for storing information representing the inventory status of each product and the expected demand for each product during each of said periods, said system being of the type which is arranged to make calculations with respect to such stored information so as to determine the net requirements for each product;

that improvement to such system wherein said data file means stores information giving for at least certain of the products the identities and yields of items which are expected to be produced as by-products when testing for any particular one of said certain products, said items corresponding to other products being manufactured in said manufacturing operation;

means to calculate, based on said by-product information, the number of such by-product items which will result from testing for such particular product; and

means for adjusting the effective inventory data for said other products in accordance with said by-product number data.

8. In combination in a system for controlling the flow of a plurality of products and components thereof through the assembly and testing procedures of a manufacturing operation comprising plural sequential process steps, including the development of an assembly and test schedule for the successive periods of a forward planning time frame, and wherein the products are organized in families comprising different grades meeting respective performance specifications corresponding to different quality ratings, said system including data file means for storing information representing the inventory status of each product grade and the expected demand for each product grade during each of said periods, said system being of the type which is arranged to make calculations with respect to such stored information so as to determine the net requirements for each product grade during each of said successive time periods;

that improvement to such system comprising means operable in making said calculations with respect to said stored information, for determining the net future requirements for at least two grades of the same product family, to downgrade a surplus portion of the higher grade to the lower grade and to adjust the effective inventory of both grades.

9. A system as in claim 8, including means responsive to the relative demands and yields of the two grades for making the decision as to whether to downgrade.

10. A system as in claim 9, wherein said responsive means comprising means to calculate, for the two products, the following:

$$\sum_{I=W}^{N} (D_I - B_I) \times (N - I + 1)/Y$$

where

I is a counter beginning with a current week W

N is a total number of periods (weeks) in a plan

D_I is a demand for the grade in the period

B_I is a by-product supply in the period

Y is a yield to the grade

(N−I+1) is a factor which gives greater weight to near term demand than future demand.

* * * * *

United States Patent [19]

Ockman

[11] **Patent Number: 4,700,318**

[45] **Date of Patent: Oct. 13, 1987**

[54] **PROJECT CONSTRUCTION WITH DEPICTION MEANS AND METHODS**

[76] Inventor: **Stuart Ockman,** 210 Copples La., Wallingford, Pa. 19086

[21] Appl. No.: **641,160**

[22] Filed: **Aug. 16, 1984**

Related U.S. Application Data

[63] Continuation-in-part of Ser. No. 559,706, Dec. 9, 1983, abandoned.

[51] **Int. Cl.**[4] **G06F 15/40;** G06F 15/60

[52] **U.S. Cl.** **364/518;** 364/512; 364/468; 434/108

[58] **Field of Search** ..... 364/518, 521, 512, 505, 364/400, 401, 419, 468; 334/108; 239/89 R, 69

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,006,345	2/1977	Bengtson	434/108
4,017,831	4/1977	Tieden et al.	364/518 X
4,019,027	4/1977	Kelley	434/108 X
4,108,356	8/1978	Bengtson	434/108 X
4,181,954	1/1980	Rosenthal et al.	364/512 X
4,275,449	6/1981	Aish	364/512
4,332,012	5/1982	Sekine et al.	364/468
4,352,165	9/1982	Hevenor	364/900
4,468,732	8/1984	Raver	364/200
4,498,139	2/1985	Malinovsky	364/518

OTHER PUBLICATIONS

Computer Decisions (Oct. 1980), "Capturing the Third Dimension", Katz et al, pp. 50–53.

IEEE Computer Graphics & Applications (Oct. 1981), "Computer Graphics and the Practice of Architecture", Fullenwider et al, pp. 18–26.

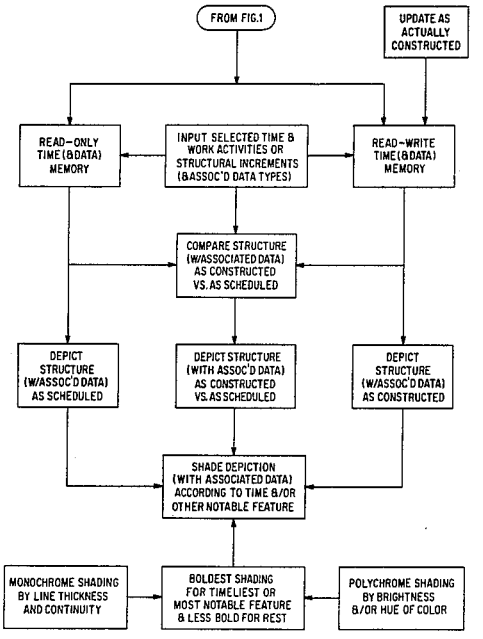
It Can't Fasten, Cut, Hammer, Drill, Dig, Bend, Level or Lift, IBM Sales Information Advertisement.

Primary Examiner—Errol A. Krass
Assistant Examiner—Kevin J. Teska
Attorney, Agent, or Firm—Charles A. McClure

[57] **ABSTRACT**

Construction of projects such as buildings, bridges, dams, industrial plants, means of transport, or the like with the aid of means and methods for incremental depiction of such buildings, etc. Complex structural features of the construction project are divided into discrete increments and characterized by ordered work activities essential to their construction, then are depicted as in construction drawings to assist in supervision of the project itself. Such characterization and depiction enable urgencies to be emphasized, as by shading structural features due to be started or finished by some given time in the construction schedule differently from those not due until later—or to have been completed earlier. Likewise, during actual construction, deviations from the project schedule can be distinguished readily, as by depicting structural features due (or overdue) in boldest shading and by diminishing the boldness of shading the more time is available within which to complete subsequent features. The boldness of monochromatic shading may be by line thickness and/or continuity, whereas for polychromatic representation it may be by hue and/or brightness of color. Storage and retrieval of structural information, work activities, and other data are conveniently machine-assisted , as by an appropriately programmed general-purpose computer or by an otherwise similar computer especially dedicated to such purpose.

34 Claims, 10 Drawing Figures



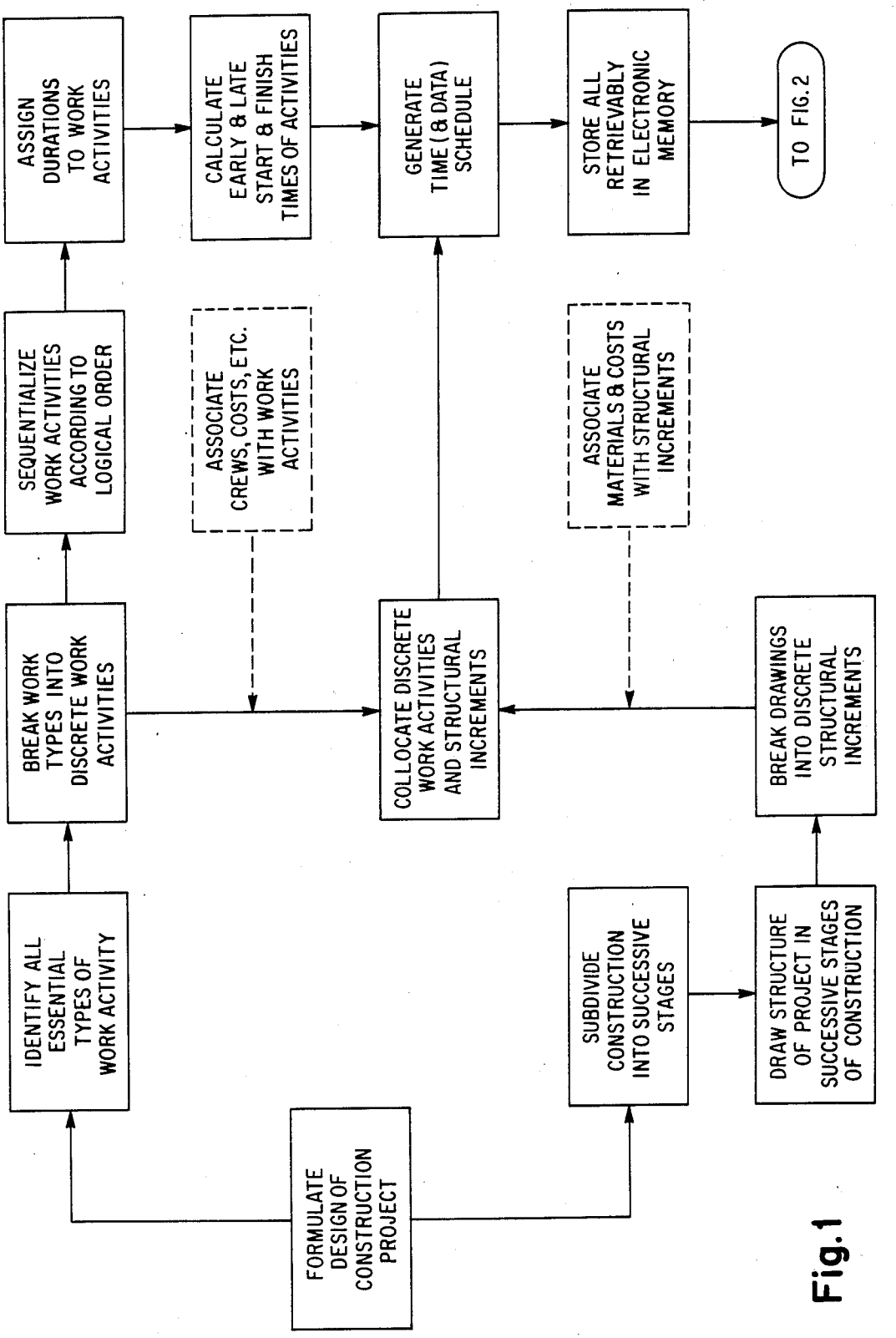
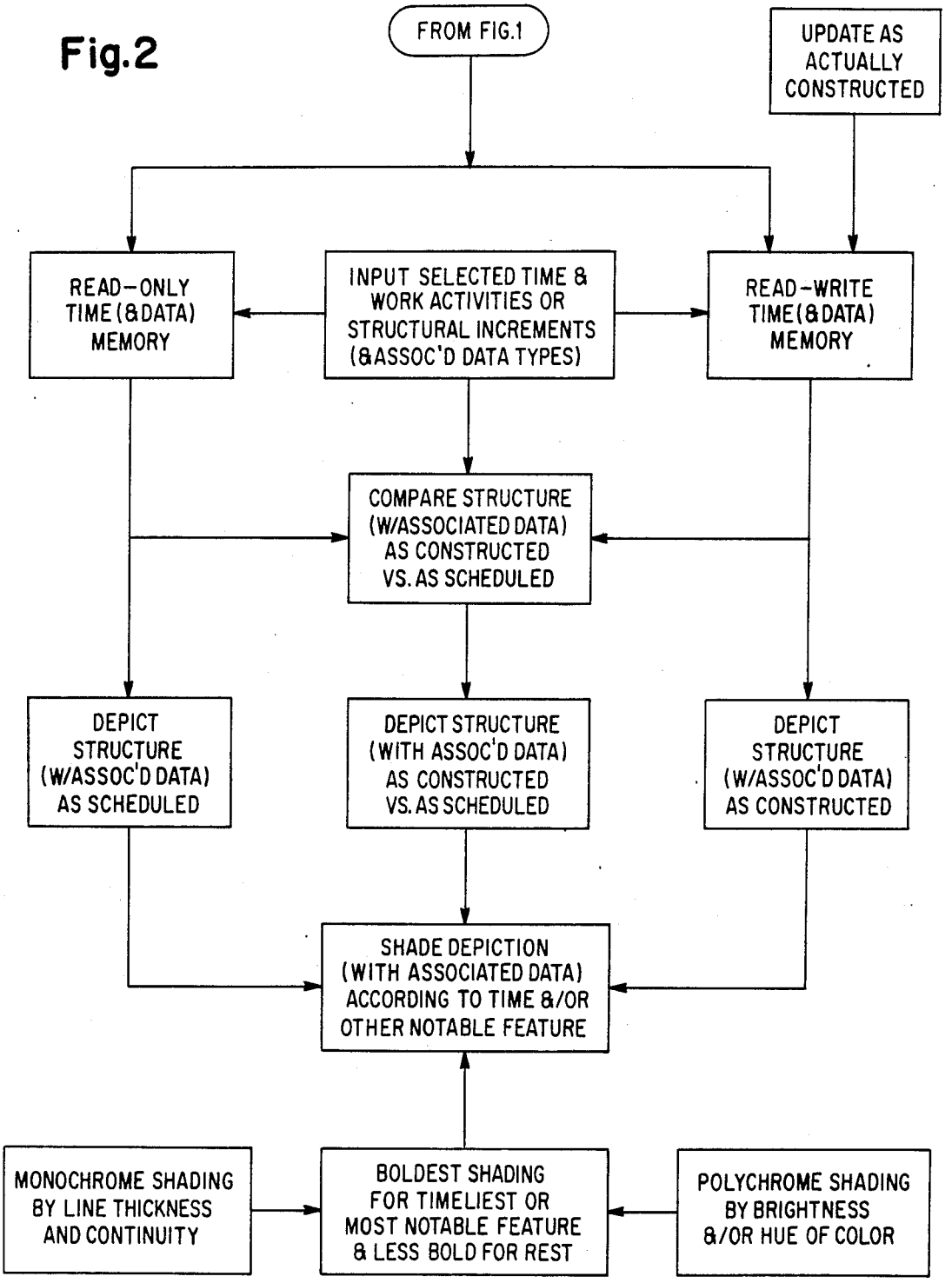

FORMULATE DESIGN OF CONSTRUCTION PROJECT
IDENTIFY ALL ESSENTIAL TYPES OF WORK ACTIVITY
BREAK WORK TYPES INTO DISCRETE WORK ACTIVITIES
SEQUENTIALIZE WORK ACTIVITIES ACCORDING TO LOGICAL ORDER
ASSIGN DURATIONS TO WORK ACTIVITIES
ASSOCIATE CREWS, COSTS, ETC. WITH WORK ACTIVITIES
CALCULATE EARLY & LATE START & FINISH TIMES OF ACTIVITIES
COLLOCATE DISCRETE WORK ACTIVITIES AND STRUCTURAL INCREMENTS
GENERATE TIME (& DATA) SCHEDULE
ASSOCIATE MATERIALS & COSTS WITH STRUCTURAL INCREMENTS
STORE ALL RETRIEVABLY IN ELECTRONIC MEMORY
TO FIG. 2
SUBDIVIDE CONSTRUCTION INTO SUCCESSIVE STAGES
DRAW STRUCTURE OF PROJECT IN SUCCESSIVE STAGES OF CONSTRUCTION
BREAK DRAWINGS INTO DISCRETE STRUCTURAL INCREMENTS

Fig. 1

Fig.2


FROM FIG.1
UPDATE AS ACTUALLY CONSTRUCTED
READ-ONLY TIME (&DATA) MEMORY
INPUT SELECTED TIME & WORK ACTIVITIES OR STRUCTURAL INCREMENTS (&ASSOC'D DATA TYPES)
READ-WRITE TIME (&DATA) MEMORY
COMPARE STRUCTURE (W/ASSOCIATED DATA) AS CONSTRUCTED VS. AS SCHEDULED
DEPICT STRUCTURE (W/ASSOC'D DATA) AS SCHEDULED
DEPICT STRUCTURE (WITH ASSOC'D DATA) AS CONSTRUCTED VS. AS SCHEDULED
DEPICT STRUCTURE (W/ASSOC'D DATA) AS CONSTRUCTED
SHADE DEPICTION (WITH ASSOCIATED DATA) ACCORDING TO TIME &/OR OTHER NOTABLE FEATURE
MONOCHROME SHADING BY LINE THICKNESS AND CONTINUITY
BOLDEST SHADING FOR TIMELIEST OR MOST NOTABLE FEATURE & LESS BOLD FOR REST
POLYCHROME SHADING BY BRIGHTNESS &/OR HUE OF COLOR

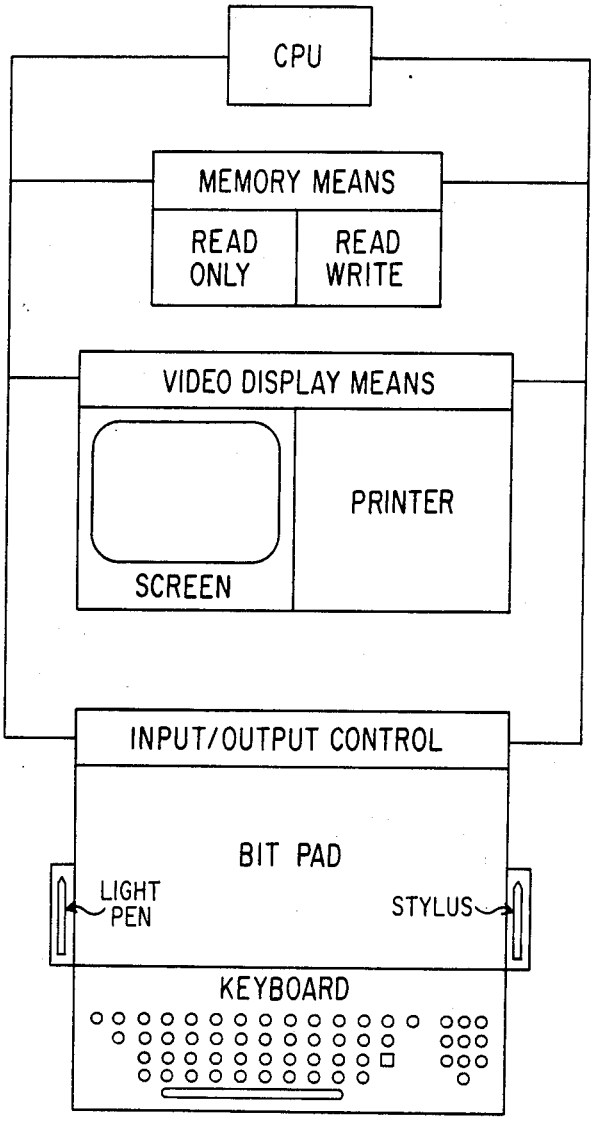

CPU
MEMORY MEANS
READ ONLY
READ WRITE
VIDEO DISPLAY MEANS
PRINTER
SCREEN
INPUT/OUTPUT CONTROL
BIT PAD
LIGHT PEN
STYLUS
KEYBOARD

Fig. 3

FIG. 3A

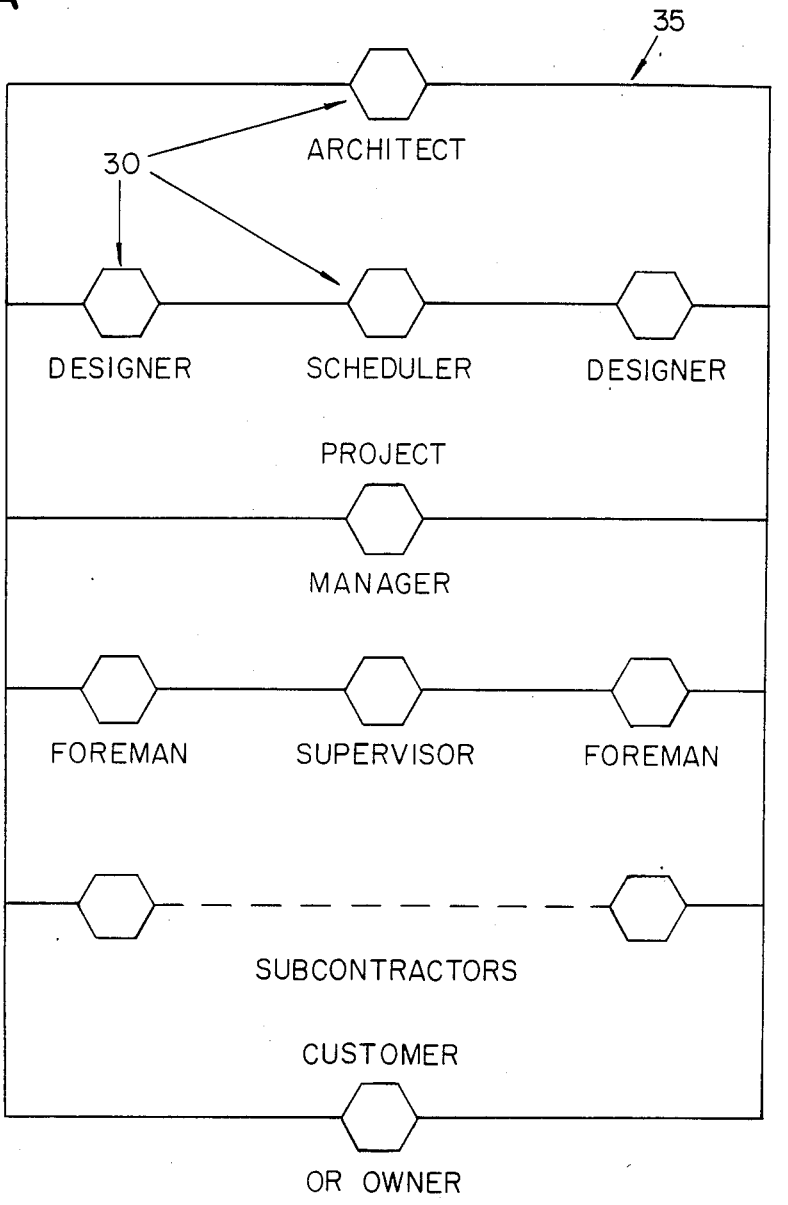
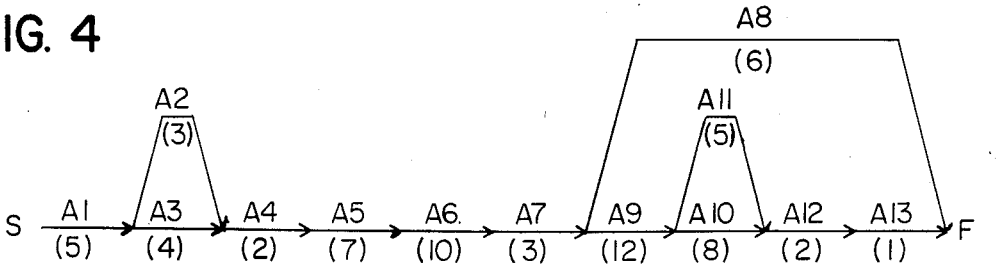

35
30
ARCHITECT
DESIGNER
SCHEDULER
DESIGNER
PROJECT
MANAGER
FOREMAN
SUPERVISOR
FOREMAN
SUBCONTRACTORS
CUSTOMER
OR OWNER

FIG. 4


A8
(6)
A2
(3)
A11
(5)
S
A1
(5)
A3
(4)
A4
(2)
A5
(7)
A6.
(10)
A7
(3)
A9
(12)
A10
(8)
A12
(2)
A13
(1)
F

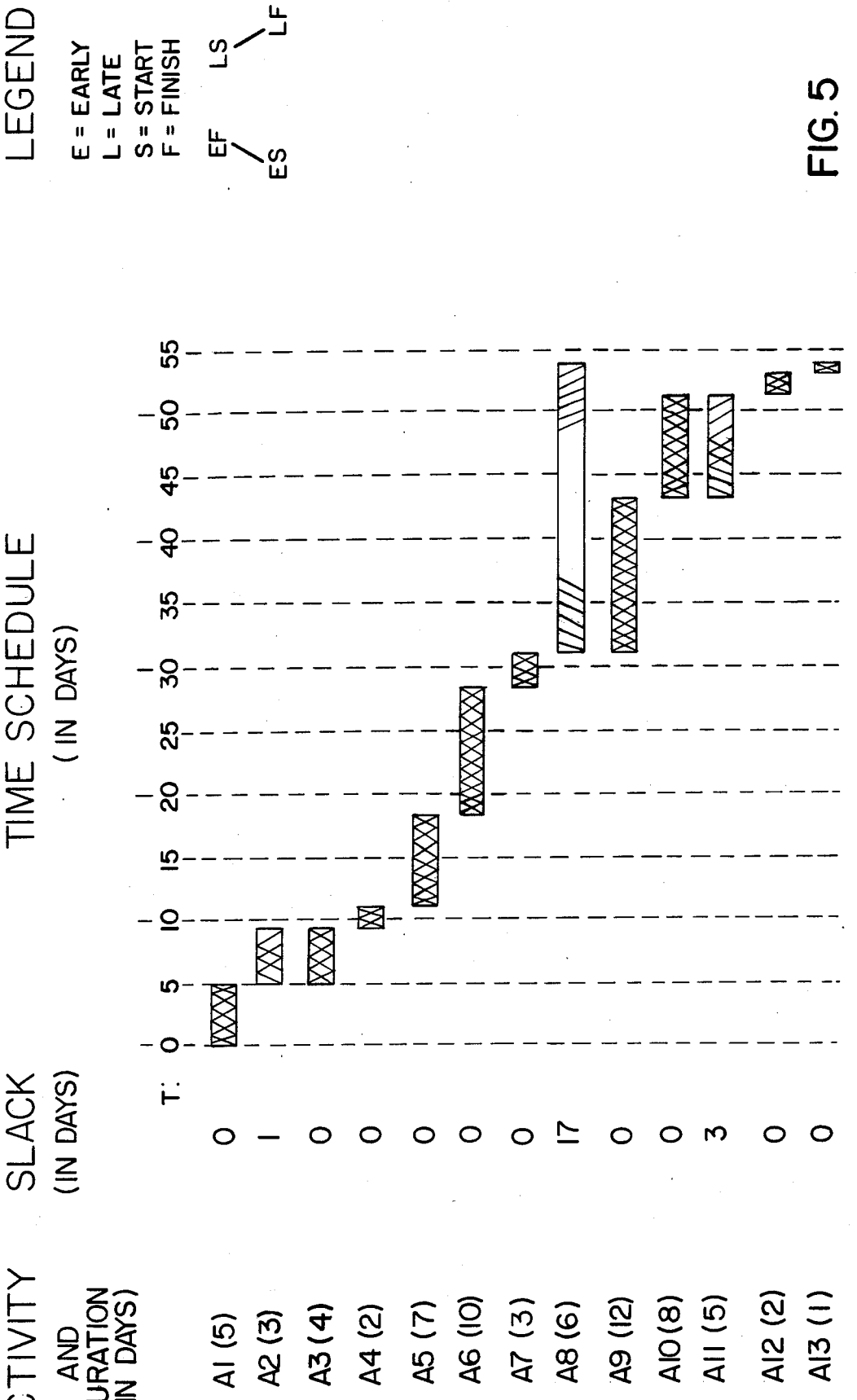

LEGEND
E = EARLY
L = LATE
S = START
F = FINISH
EF
ES
LS
LF
TIME SCHEDULE
(IN DAYS)
SLACK
(IN DAYS)
ACTIVITY
AND
(DURATION
IN DAYS)
T:
0
5
10
15
20
25
30
35
40
45
50
55
A1 (5) 0
A2 (3) 1
A3 (4) 0
A4 (2) 0
A5 (7) 0
A6 (10) 0
A7 (3) 0
A8 (6) 17
A9 (12) 0
A10 (8) 0
A11 (5) 3
A12 (2) 0
A13 (1) 0

FIG. 5

FIG. 6

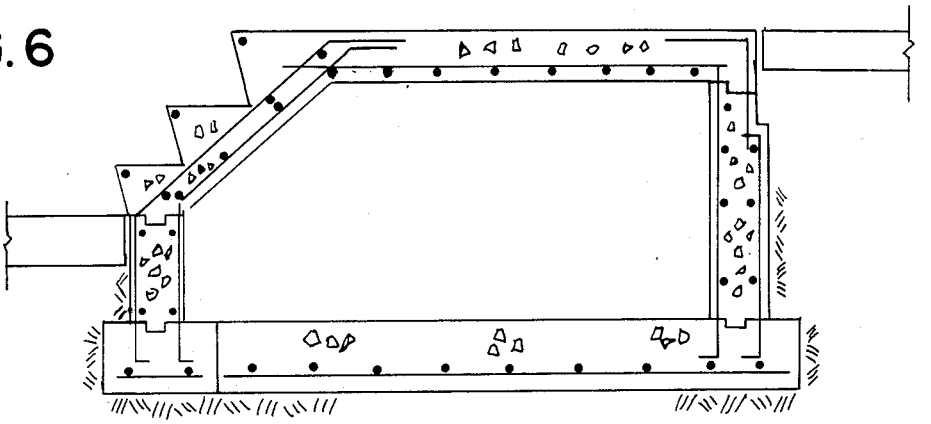


FIG. 7

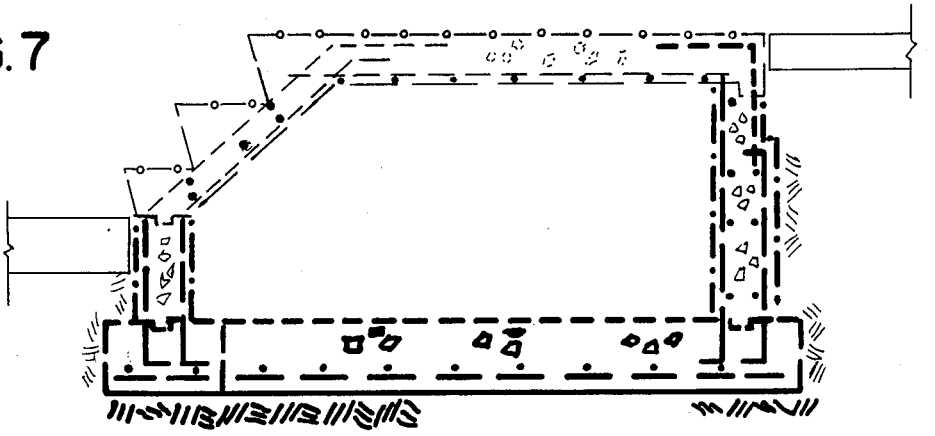
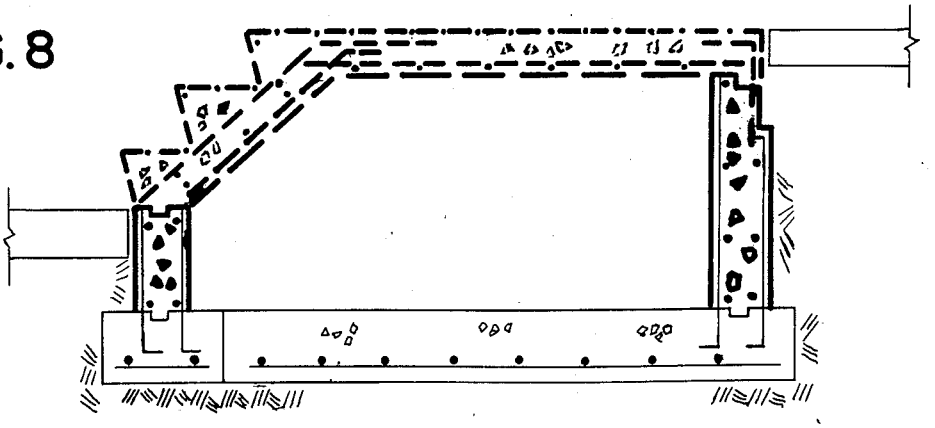


FIG. 8



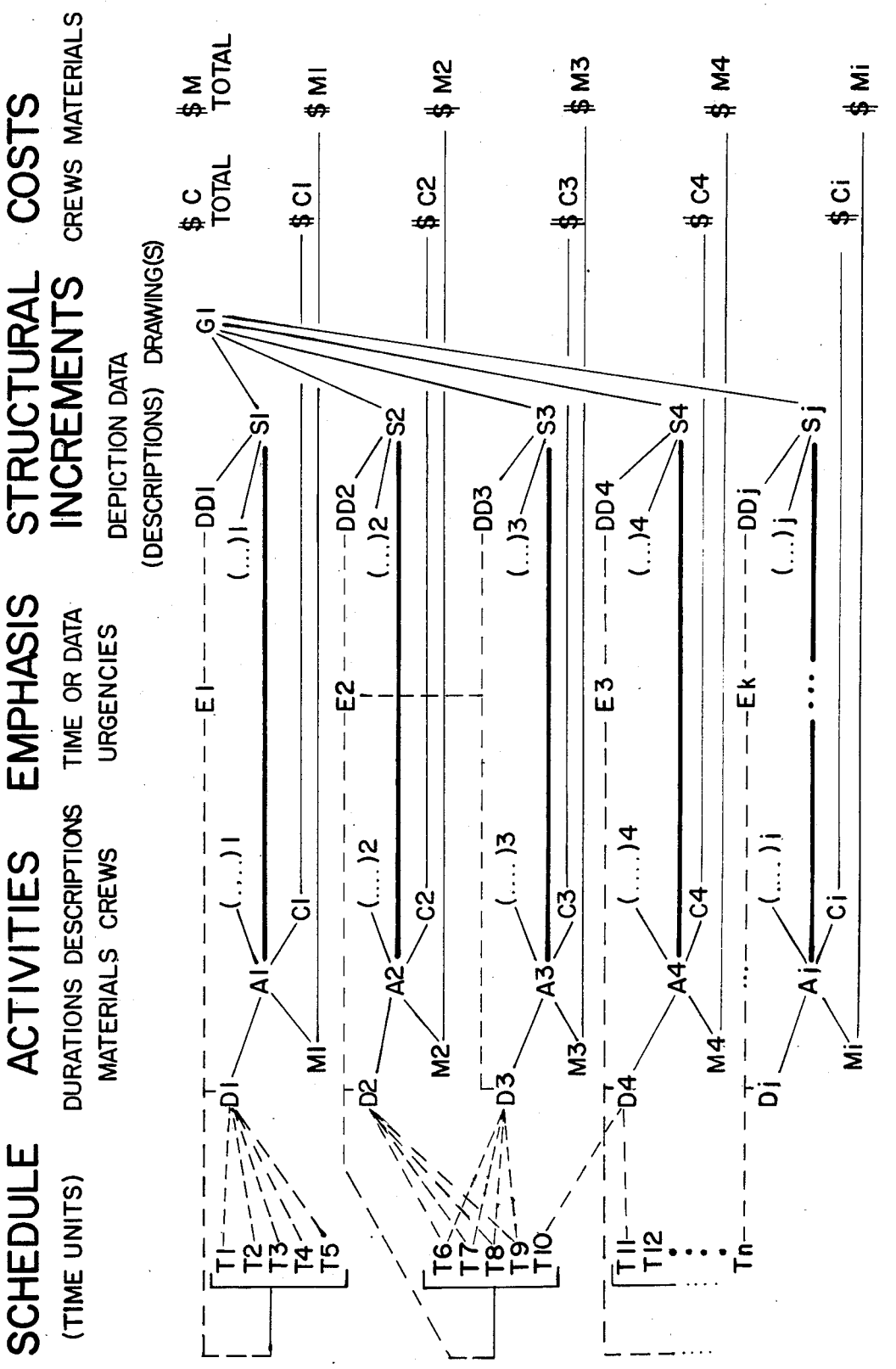

SCHEDULE ACTIVITIES EMPHASIS STRUCTURAL COSTS
(TIME UNITS) DURATIONS DESCRIPTIONS TIME OR DATA INCREMENTS CREWS MATERIALS
MATERIALS CREWS URGENCIES
DEPICTION DATA
(DESCRIPTIONS) DRAWING(S)
$ C TOTAL
$ M TOTAL
G1
E1
E2
E3
Ek
DD1
DD2
DD3
DD4
DDj
S1
S2
S3
S4
Sj
(....)1
(....)2
(....)3
(....)4
(....)j
$ C1
$ C2
$ C3
$ C4
$ Ci
$ M1
$ M2
$ M3
$ M4
$ Mi
A1
A2
A3
A4
Aj
C1
C2
C3
C4
Ci
M1
M2
M3
M4
Mi
D1
D2
D3
D4
Dj
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
Tn

FIG. 9

PROJECT CONSTRUCTION WITH DEPICTION MEANS AND METHODS

This is a continuation-in-part of my similarly entitled copending application, Ser. No. 559,706 filed Dec. 9, 1983 and subsequently abandoned.

This invention relates to novel visual depiction of structural and optional indication of related features of construction projects in their planned and actual construction, and is applicable to construction of buildings, bridges, dams, industrial plants, means of transport, and the like.

Complex construction projects comprise too many components and involve too many structural and other interrelationships to be kept in mind without visual aids; such aids include drawings of the intended results, graphical and tabular schedules of the order in which components have to become available and be put into place relative to one another, and budgets, for example. However, it is difficult to assemble the pertinent information in an intelligible manner, especially for such diverse audience types as artisans, architects, contractors, engineers, financiers, and others. Also, in international construction projects, diversity of language is often a barrier to common understanding of what has to be done. Ordinarily, whenever a project requires a large set of drawings, it (and they) are divided into manageable subsets, often with the aid of hierarchical levels of detail in the various drawings, but such subdivided drawings require additional notations to render their interrelationships fully intellgible to individual viewers.

A primary object of the present invention is to provide novel visual depiction in representation of structural and related information about construction projects, especially complex ones.

Another object of this invention is to render intelligible at a glance the scheduled status of selected increments of structure at any given stage in a construction project.

A further object of the invention is to focus attention, in visual pictorial representations of projects undergoing construction, upon actual status of structural increments versus scheduled status.

Yet another object of this invention is to depict structural features of a construction project, whether scheduled only or actually being constructed, in such manner as to convey information about related non-structural features.

A still further object of the invention is to provide in the management of a construction project—whether scheduled only or in construction—a data base comprising indicia of discrete increments of structure, collocated indicia of sequentialized work activities, a superimposed time schedule for the project, and related data.

Other objects of the present invention, together with methods and means for attaining the various objects, will be apparent from the following description and the accompanying diagrams, which are presented by way of example rather than limitation.

FIG. **1** is a block diagram of steps taken in practicing the present invention;

FIG. **2** is a block diagram of further steps taken in practicing this invention; and

FIG. **3** is a largely schematic diagram of apparatus useful in practicing the invention; and

FIG. **3A** is a schematic diagram of a network of workstations, each comprising apparatus such as is shown in the preceding view.

FIG. **4** is a project-scheduling network diagram of activities for a portion of a building construction project; and

FIG. **5** is a construction project schedule corresponding to the scheduled activities network diagram of the preceding view.

FIG. **6** is a substantially conventional construction drawing of the same scheduled portion of a building construction project;

FIG. **7** is a similar drawing of the same building portion, but shaded in accordance with the scheduled start of construction; and

FIG. **8** is a similar drawing of the same building portion, but shaded according to the project status well into the schedule, with part of the scheduled construction overdue for completion.

FIG. **9** is a schematic diagram of interrelationships between and among data elements useful according to the present invention.

In general, the objects of this invention are accomplished, in building and similar construction projects, by depicting selected structural features characterized by times of the work activities essential to their construction, plus (optionally) additional related data. Such characterization highlights urgencies, as by depicting structural features due to be started or to be completed by any given time in the construction schedule, more boldly than structural features not then due. At any given time during actual construction, structural features due (or overdue) for completion can be emphasized in similar manner, with little or no resort to verbal representation, in a mnemonically helpful way that is advantageous in reducing time, effort and cost or taking other remedial measures.

In summary detail, one is enabled to attain those objects by a sequence of steps including identifying, in advance, discrete increments of structure from start to finish of a given project, depicting the structural increments as in construction drawings, identifying discrete work activities essential to constructing the respective structural increments, assigning durations to the given activities, sequentializing the work activities among themselves, calculating start and finish times for the respective activities so sequentialized, preparing a time (and data) schedule from start to finish of the project, and storing the foregoing for retrieval.

Thus, the invention provides a data base management for and uses construction projects, with a time schedule as a key by which to relate indicia of sequentialized work activities and discrete increments of resulting structure, as well as other similarly related data. The resulting relational data base enables data associated with either the work activities or the structural increments to be related to the other by way of the time schedule—whether expressed in days, weeks, or other units of time—and enables the structural increments to be depicted with differential emphasis dependent upon urgencies of time or other important data item, such as cost.

FIG. **1** illustrates these and related steps in more detail. The usual preliminary task is to formulate the design of a given construction project, as in terms of specific design criteria. The intended construction is subdivided into successive stages, and corresponding structural drawings are prepared, subdivided into dis-

crete increments of structure. Likewise, kinds of work activity essential to the construction are identified and are broken down into discrete work activities. Optionally, as suggested within broken lines, related data may be associated with work activities (e.g., budgeted labor costs, crew sizes, and availability) or with structural increments (e.g., budgeted materials costs, quantities, and availability). Discrete structural increments and discrete work activities are collocated; that is, cross-referred to each other, in a one-to-one relationship.

Too, as noted in FIG. 1, durations are assigned to repsective work activities, whch are sequentialized among themselves in their logical order (i.e., which of them must precede which, and both early and late start and finish times are calculated for those activities. The resulting time data and the collocated discrete work activities and structural increments, together with other related data, are then used to generarte a time (and data) schedule for the entire construction project from start to finish, whereupon the schedule and related information are stored for subsequent retrieval, as in electronic memory means.

It will be understood that, in generality, the foregoing steps can be carried out by hand, with pencil and paper, but that large construction projects are so complex that it is convenient to have the assistance of inanimate apparatus, such as a digital computer, capable of speeding up the necessary logical and mathematical calculations and, optionally, the actual drafting process, so as to complete them within a reasonable time. Storage in electronic memory means follows naturally upon such machine-assisted steps.

FIG. 2, in similar manner to FIG. 1, shows subsequent steps that, although also conceivably performable by hand, actually are reasonably feasible only with similar machine assistance. Indeed, FIG. 2 begins by showing the storing of information from FIG. 1 twice: once in a read-only memory, which represents the scheduled construction, and additionally in a read-write memory that can be updated from time to time in accordance with actual construction during the project. One or more intermediate types of memory (not shown), that may be overwritten only after due precautions are met may be interposed to represent target schedules bearing authorized changes from the original schedule, as wil be readily apparent. However, in the interest of clarity no such embodiment is described further here or illustrated in the drawings, nor are means shown for storing various operating programs, or for enabling essential calculation to be performed, as is conventional.

As shown in FIG. 2, the read-write memory is updated during the project with whatever departures from the schedule actually occur in the process of construction. The operator may retrieve upon request from either memory a structural depiction, as of any selected time within the schedule, characterized by time urgency or other data so stored. Also, the two memories can be compared in any such regards to disclose any discrepancies between the two, as where actual construction is ahead of schedule (or under budget) or—usually more important—is behind schedule (or over budget), for example. Furthermore, either of the memories can be compared with one or more intermediate memories to identify any differences between original or current plans and interim target schedules—and to depict such differences in terms of the corresponding structure.

As indicated further in FIG. 2, according to this invention the structural increments of interest (or related to other items of interest) are depicted so as to emphasize their status, such as urgency, timeliness, or expense, as well as (if desired) annotated with related numerical data or verbal comments. Such emphasis is accomplished by what is called "shading" and more particularly as quasi-mnemonic "boldness" of shading to signify comparable (but unlike) characteristics in monochromatic and polychromatic video representations.

Because of black's optical dullness it is especially suitable for both monochromatic and polychromatic depiction of structural features already completed or otherwise not requiring emphasis. As shown subsequently, boldness and monochrome shading is conveniently represented by line thickness and continuity; thus, continuous or solid lines are bolder than broken ones, and thick lines are bolder than normally thin ones. In polychrome, shading is represented by hue and/or brightness of color. Structure overdue for completion takes the boldest representation, such as a double-thick line in monochrome, or the boldest color (e.g., red) in polychrome. Those structures with later due dates drop off incrementally to less bold colors, as in spectral order (e.g., orange, yellow green, forest green, sky blue, navy blue, violet—or dark brown); for which suitable monochrome analogs are long-dash, short-dash, and dash-dot double-thick lines, followed by long-dash, short-dash, and dash-dot normal or single-thickness lines, respectively. For convenience, these relationships are summarized in the table below, although it will be recognized that others may be substituted with like effect.

TABLE 1

	PRIORITIES AND SHADINGS	
PRIORITIES	POLYCHROME	MONOCHROME
Greatest	Red	Continuous double-thick
Next greatest	Orange	Long-dash double-thick
Not so great	Yellow Green	Short-dash double-thick
Intermediate	Forest Green	Dash-dot double-thick
Less	Sky Blue	Long-dash single thickness
Even less	Navy Blue	Short-dash single thickness
Least	Dark Brown	Dash-dot single thickness

Such graduated boldness of shading of the structural increments themselves conveys at a glance language-independent information that otherwise could be conveyed'if at all—only by relatively less effective methods. Of course, more or fewer gradations could be used, as could other shading methods in color or in monochrome. Users differ in preference for monochrome or polychrome emphasis of lines, but where area shading is desired the usual choice is color.

A useful supplementary representational aid is an intensity difference sufficient to be readily distinguishable from existing shadings. Whether continuous or intermittent, such differences are useful for emphasize areas, lines, or points of particular interest. For example, features to be removed may be illustrated as above, but at reduced (or increased) intensity, all or part of the time.

A blinking or flashing of the video display, such as between normal intensities, is useful for noting the presence of slack or float time in an activity for constructing a given structural increment. Blinking or flashing is also analogously useful to show temporary features (e.g., forms) that are both placed and removed during the course of the project. The showing of such temporary

features preferably will alternate between the respective shadings characterizing their priorities of placement and removal, with the latter preferably diminished in intensity. Another use of blinking or flashing of the display, such as between normal and higher intensities, is to draw attention to structural features characterized by some data element (e.g., availability of work crews or materials, budgeted or actual costs or variances between them, or progress payments) not usually found in drawings. Verbal or graphical addenda may be included to similar effect, optionally with the aid or arrows, flags, or other pointers.

FIG. **3** shows computer apparatus **30** comprising components to enable this invention to be practiced with a high degree of machine assistance. Despite the largely schematic diagram here, persons ordinarily skilled in the art of digital computers will readily understand what is meant, so as to become enabled to practice this invention with relative ease. Shown centrally is the video display means, which conveniently comprises a video screen and optionally a printer capable of displaying both alphanumerics and graphics. A plotter or photographic device may replace or supplement the printer as a static or non-transient display device, whereas video screens or equivalent devices can provice a dynamic or transient display, such as blinking or alternating over time between more than one representation, which is sometimes desirable. Connected to the video means is electronic memory means including both read-only and read-write memories, as previously described. Also connected thereto are a central processing unit (CPU) and input/output control means, as is conventional.

The input/output means conveniently includes not only the customary keyboard but also an electronic drafting surface or "bit pad" to transmit into the memory drawings made thereon, as with the indicated stylus. Alternatively, a light pen may be used similarly in conjunction with the screen. Not shown is conventional scanning means that may be used with pen-and-ink construction drawings to convert them into readily stored digital signals. The foregoing apparatus may be of general-purpose type or may be dedicated especially to practicing this invention.

FIG. **3A** shows shematically a number of computer workstations **30** connected together into communications network **35**, which itself is useful according to this invention. One or more workstations **30** can readily be made available to major functional groups involved in any construction project (e.g., proposal, design, scheduling, procurement, construction, and startup); or trades (e.g., concrete, carpentry, electrical, plastering, and plumbing); and at levels of management from foremen through intermediate supervisory personnel to the project manager; and optionally even to higher line managers and possibly to pertinent staff personnel (e.g. financial, legal, or writers). This diagram fragmentarily illustrates representative availability of such apparatus in actual construction operations, specifically to the architect, designers, scheduler, contractor's managers, subcontractors, and customer. Of course, authority to revise the design or the schedule and update work performed and structure built should be limited to authorized persons. Great care should be taken for security of the data base.

It will be understood, of course, that such networking is well within the current state of the art and yet may take advantage of future improvements, inasmuch as the proprietary nature of this invention lies more in what is done than in the specific means by which it is accomplished. The same is true of the computer means. Individual stations **30** preferably have self-contained computer units or "nodes" with their own memory banks and so interconnected that what is stored in the memory of each is available to every other unit, and that unavailability of any unit does not interrrupt the network but only renders the unit and its memory unavailable until again on line. Also desirable are high resolution of display so as to show the structural increments in considerable detail, and rapid response because of the quantity of detail to be processed. A computer that meets these needs is readily available from Apollo Computer Inc. (of Chelmsford, Mass.) under the name DOMAIN.

FIG. shows a project scheduling network diagram resulting from sequentialization of work activities for constructing a minor portion of a building as shown in subseqeunt diagrams. About a dozen activities (designated by arrows) suffice to go from early to late stages, such as excavation, forming, placing, and stripping of foundation, stairs, and landing. The respective activities are designated by An above the arrow, where n is a numerical indicator of a given work activity; and by (d) below the arrow, where d is the number of days from start to finish of such activity. Thus, the first activity arrow is marked **A1/(5)**; the 1 indicating that it is the first, and the 5 indicating that it is to require five days.

Thus, FIG. **4** shows the critical path as a horizontal line from left to right, made up of arrows to which are juxtaposed activity indicia **A1**, **A3**, **A4**, **A5**, **A6**, **A7**, **A9**, **A10**, **A12**, and **A13**; whereas activities **A2**, **A8**, and **A11** all require less time to complete than do one or more parallel activities and so have slack time available to them, as appears more definitely in the next view. Activities may be identified in more complex manner (e.g., with indicia of trade type or other information), but these simple indicia will suffice here.

Tabulated below are the work activities and their indicia.

TABLE 2

EXAMPLE OF WORK ACTIVITIES	
INDICIA	DESCRIPTIONS
A1	Excavate for foundation slab
A2	Place reinforcing for foundation slab
A3	Place forms for foundation slab
A4	Place foundation slab concrete
A5	Place reinforcing for walls
A6	Place forms for walls
A7	Place wall concrete
A8	Strip forms already placed, backfill
A9	Place forms for stairs and landing
A10	Place reinforcing for stairs
A11	Place reinforcing for landing
A12	Place stairs and landing concrete
A13	Strip remaining forms

FIG. **5** shows a resulting construction schedule, with a time scale (in days) at the top, and at the left a list of the various activities by indicia (An, where n is the number assigned to the given activity) plus duration (in days). Activity early start and late finish dates (days), respectively, are indicated by the left and right ends of the horizontal bars extending to the right of the respective activity indicia. Inside the bars are both forward and backward slant (or slash) characters, whose leftmost ends indicate the respective early and late start times and whose right ends indicate the respective early and late finish times, as stated in the Legend included in

the upper right corner of the drawing. A bar completely cross-hatched by the oppositely slanted characters indicates the duration of an activity without slack, whereas a bar that is not completely cross-hatched indicates presence of slack. Thus, activity A2 has one day of float or slack, activity A8 has seventeen such days, and A11 has three of them. Such a bar chart is an accepted scheduling aid, but its graphical character is only indirectly (verbally) related to the intended resulting structure.

FIG. 6 shows in sectional elevation the corresponding portion of the scheduled construction project, comprising a plurality of structural increments uncharacterized as to the actual or scheduled order in which they were (or were to have been) completed, together with forms, reinforcing bars, etc. to be used in the construction. Included are a foundation, walls, stairs, and a landing—all rather fragmentarily shown—but adequate for an illustrative example. In more complex drawings, features to be shown might be grouped by type of work activity, location on a drawing, or by subcontractor, for example. As will be apparent, ordering or ranking of pictorial structural increments may be by starting times or completion times (or both can be shown alternately, or on two separate display devices simultaneously). For convenience, the examples here favor completion times as a matter of choice. Shown in normal continuous lines at the middle left and upper right are adjacent structural features not part of the present schedule.

FIG. 7 depicts the same structural portion (of FIG. 6) shaded in monochrome according to the scheduled activity completion times (of FIG. 5). Scheduled first as most urgent is the excavation for the foundation, shown accordingly in continuous double-thick lines (which in polychrome would be red, but of merely single thickness). Scheduled next are reinforcing bars—horizontal for the foundation and vertical for the walls—which appear in long-dash thick lines (orange if in color). Next, in short-dash double-thick lines (cf. yellow green) is the concrete foundation itself. Scheduled next are wall forms, shown in thick dot-dash lines (cf. forest green), which—because they are to be removed subsequently—should appear intermittently in diminished intensity, preferably coded to their removal deadline if in an electronic video display (but not here). Structural increments resulting from the next few activities are shown in long-dash single-thickness lines (cf. sky blue): the foundation walls (formed and placed, stripped and backfilled). However, because of the slack in the stripping activity, it may well appear alternately as an item of least urgency (dot-dash, cf. dark brown). Forms for both the stairs and the landing, and the reinforcement for the stairs, scheduled next, are shown in normally thick (thin) short-dash lines (cf. navy blue). The concrete stairs and landing, as well as final stripping, being of lesser priority, are also dot-dashed (cf. dark brown) normal single-thickness lines.

FIG. 8 shows in similar fashion the same structural portion, in its actual stage of construction on a given day (T29, the start of the second half), but shaded to emphasize construction lagging the schedule. Reference to FIG. 5 indicates that the first six of the activities were to have been completed by T28. A1–A5 are done, so their resultant structural increments are shown in continuous black single-width lines (similar if in color), confirming that the project is on schedule to that extent. However, FIG. 8 shows by is use of solid double-thick lines (cf. red) that A6 (placing forms for the walls) is behind schedule and, thus, very likely to delay subsequent activities, especially those having no scheduled slack (A7, placing wall concrete; A9, forms for stairs and landing; and A10, stairs and landing concrete). Such visual emphasis encourages prompt remedial measures, such as shortening the duration of one or more of the activities, as by enlarging work crews or paying them for overtime—doubtless at increased cost, which can be similarly highlighted in a separate depiction as an aid to decision-making.

In FIG. 8 slack-rich form-stripping step A8 is shaded like A13 (the one-day final form-stripping step) because of their identical completion dates, shown in FIG. 5. Activity A11 (placing landing reinforcing), which also has some slack, is shaded like the similar step of placing reinforcing for the landing because they also have identical required completion dates. According to this invention, as already noted, such slack could be shown readily in a video Display—not here—by flashing between the respective shadings for the activity's scheduled early start and permissible late finish. Two-stage flashing to indicate slack preferably differs from that for showing placing and removal of a structural feature by being of equal intensity in both stages of its showing (compared with a diminished intensity in one of the stages for a removal feature).

FIG. 9 shows schematically main data elements under headings: SCHEDULE, ACTIVITIES, EMPHASIS, STRUCTURAL INCREMENTS, and COSTS. The elements are shown in the form of alphabetical indica, which in general form are Tn for schedule times, Ai for work activities, Ek for shading emphasis, Sj for structural increments, $Ci for crew costs, and $Mi for materials costs. In specific occurrences numbers are substituted for the appended lower-case letters, and in the drawing such numbers are more or less sequential. Mentioning of the data elements frequently by name (rather than by indicia) in further discussion of this diagram will be understood as compatible (rather than in conflict) with the principle that the indicia stand for their respective elements in this data base and are often meant when the context fails to indicate otherwise.

Interrelationships among the indicated data elements in FIG. 9 are shown by lines (of various degrees of boldness) joining them. Primary linkage between a given discrete work activity, on the one hand, and the increment(s) of structure produced by that activity, on the other hand, are represented in bold horizontal lines. Each activity has linked to it, as shown by ordinary lines, its duration Di, its crew Ci and materials Mi requirements, and its description (. . .)i—all expressed here in general terms. Each structural increment has similarly linked to it: its graphical depiction data DDj and usually also verbal description (. . .)j. Correspondingly numbered descriptions of activities and structural increments may be merged into composite descriptions, if desired. A many-to-one relationship of structural increments to their source drawings is common, and here they all are linked to only one drawing (G1).

In FIG. 9 the duration of each work activity is linked by dashed lines at the left to the time schedule (shown fragmentarily as a succession of time units T1 to T12 . . . Tn). One such linking line appears for each time unit required by the activity duration. Early and late times—where not identical—would unduly complicate this diagram and, thus, are omitted here for clarity). Successive time units are bracketed into five-unit intervals for convenience of illustration and as an example of

a common range (a 5-day week). Other user-defined ranges may be substituted as the user may prefer for review of the project or schedule, and the degrees of emphasis, e.g., shorter ranges in the near term and longer ones further away. Here the listed activities (A1 through A4) and their durations are those of the same example treated in FIG. 5 and corresponding text.

The middle heading in FIG. 9 is EMPHASIS, a concept most often based upon time urgency but sometimes on some other connected data element, such as cost. Here emphasis indicia (for degrees of time urgency or data variance) E1, E2, E3, . . . Ek scale downward, linked to both the duration indicia and the time schedule by broken lines, to indicate that the linkage is subject to changes as time passes. Successive time units (e.g., days) may be deemed to define a fixed reference system, whereas emphasis can be expected to slide fairly regularly past time and relative to the various activities as well, though perhaps not so regularly in the instance of budgeted costs.

Also commonly in many-to-one relations are types (and numbers) of workers and materials, but it is convenient to cross-reference each work activity, in one-to-one manner, to its own crew and materials requirements (shown in FIG. 9 as Ci and Mi). It is apparent that the time schedule is linked to crews, materials, and cost data via the activity indicia (and durations). Crews and materials are linked directly to their costs—shown generally as $Ci and $Mi (at the lower right)—below their individual values and (at the upper right) their totals $C and $M. Of course, averages, dispersions, variances, other statistical and accounting measures, and the like can be calculated for various data categories.

In such a data base representation (as FIG. 9) any given time intersects, somewhat indirectly, at least one activity (usually many more) and structural increment(s), whereupon the array of time intersections with activities and structures constitutes a calendar of the project and may be reconstituted more directly as such—as suggested in dashed lines on the drawing. Moreover, as previously noted, the schedulded and actual construction times for the various structural features provide calendars for them in like manner, and comparison of the structures as scheduled and as actually built readily highlights departures of construction from the original schedule—which may have to be revised from time to time.

Placement of suitable relational links (shown in part in FIG. 9) in digital electronic computer memories (as via memory addresses) is well known in the art and is within the skill of software designers. Also within programming skill are the steps of converting drawn elements of structure to depiction in another medium such as a video display, shading such display with reference to a schedule or related time characteristics, comparing scheduled and actual items and determining differences between them, and annotating a display of structural features graphically with arrows or other symbols and optionally with textural information. The same is true of devising ways of varying such displays by diminished or enhanced intensity, blinking or flashing between normal and varied intensity (including on and off) and other visual modifications that will come to mind.

The drawings to be stored electronically need not be prepared differently from what draftsmen normally provide for construction projects; indeed, some customary annotation of relationships can be eliminated because they will be apparent in the video display. This is not to say that improvements in draftsmanship will not come in handy in preparing drawings for use according to this invention.

Examples of improvements include performing the drafting with implements adapted to store a drawing as soon as it is made (or while it is being made), and utilizing prepared graphical or pictorial symbols for components that can be called up from storage rather than being drawn from scratch every time they are required, also keying the views so that they can be combined into more comprehensive illustrations—all as are becoming well known in the art of computer-aided drafting. These and other measures reduce an already large memory requirement and permit advantageous reduction in paper storage, use, and disposal.

It should be understood that "structural" is used generally in this specification to denote whatever makes up a constructed work (building, bridge, etc.) rather than being interpreted in a more limited structural engineering sense. Thus, electrical, plumbing, and finishing features of construction, for example, may well be treated as structural increments in the practice of the invention, as may collateral activities essential to a construction project, such as hiring personnel, buying or leasing equipment, expediting delivery, and receiving materials on site.

As is well known in the scheduling art, the "early start" date of any given work activity is the earliest date by which all essential prior activities can be (or have been) completed, and the "early finish" date is obtained by adding the minimum duration of that activity to its early start date. The "late finish" date of a given work activity is the latest date by which that activity can be completed without delaying completion of the project, and the "late start" date is obtained by subtracting the minimum duration of that activity from its late finish date. Time units other than days may be more suitable for some projects. The assigned duration for any work activity, if not a fixed number of time units, may be expressed in a range with a stated minimum and a stated maximum, with or without a stated intermediate, such as their mean duration, a defined "most likely" duration, etc. Software for handling these variants will come readily to mind for persons skilled in the art of computer programming.

Accordingly, programs as such do not constitute part of the invention claimed here, whereas marking for display and displaying timely structural features (with or without related data) are subject methods or steps of this invention. It is also apparent that a report generator (program) may be devised so as to enable a human operator to obtain one or more of a wide variety of temporary displays or permanent printouts at the touch of a key or (with voice-recognition equipment) just by an oral request. Indeed, computer programs or software devised or adapted especially for use in effecting the present objectives may be particularly useful—and may be patentable (or not).

Advantages and benefits of this invention have been mentioned above; other advantages may become self-evident to readers of this specification; and ultimately the greatest advantages and benefits can be expected to accrue to those who undertake to practice it. Prominent among the advantages and benefits are increased facility in project scheduling and control, decrease in required personnel and communications, and consequent reduction in costs all around. Some modifications in this invention have been noted above. Others may be made, as

by adding, deleting, combining, or subdividing parts or steps, while retaining advantages and benefits of the invention, which itself is defined in the following claims.

I claim:

1. In means to accomplish a building construction project provided with a schedule of discrete work activities to construct increments of structure of the project and provided with drawings of such structural increments,

means depicting such structural increments as of a given time in the schedule, and

means shading the depicted structural increments with degrees of boldness determined by time characteristics of the depicted structural increments.

2. Construction project means according to claim 1, including display means for depicting selected structural increments with differential boldness.

3. Construction project means according to claim 2, wherein the display means is monochromatic, and boldness of shading comprises lateral thickness in depicted lines.

4. Construction project means system according to claim 2, wherein the display means is polychromatic, and boldness of shading comprises hue of color in depicted lines or areas.

5. Construction project means according to claim 1, wherein the shading means is effective to shade most boldly the increments of structure due for completion by the given time in the schedule, to shade less boldly the increments of structure due for completion within a selected increment of time after the given time, and to shade even less boldly the increments of structure not due for completion until thereafter.

6. Construction project means according to claim 1, wherein the given time is a time during the actual project construction, wherein the shading means is effective to shade most boldly the increments of structure due for completion by the given time, to shade less boldly the increments of structure due for completion within a selected increment of time after the given time, and to shade with an unlike degree of boldness the increments of structure already completed.

7. Construction project means according to claim 1, wherein the given time is a time during the actual project construction, including means for determining the time relationship between the originally scheduled and currently projected completion times for the respective structural increments, and wherein the shading means is effective to shade more boldly the increments of structure whose currently projected completion times are later than as scheduled, and to shade less boldly the increments of structure whose currently projected completion times are as scheduled.

8. Construction project means according to claim 1, wherein the given time is a time during the actual project construction, and wherein the shading means is effective to shade more boldly the increments of structure whose currently projected costs are over the budget, and to shade less boldly the increments of structure whose currently projected costs are within the budget.

9. Construction project means according to claim 1, wherein the given time is a time after the actual construction of the project, and the shading means is effective to shade more boldly the increments of structure whose actual costs exceeded the budget, and to shade less boldly the increments of structure whose actual costs were within the budget.

10. Structural graphics representation system for construction projects, comprising operatively interconnected means for depicting discrete increments of structure for a given project,

memory means for retrievably storing data about activities essential to construction of the discrete structural increments and data representing durations of the respective work activities and their sequencing,

processing means to calculate therefrom early and late start and finish times for the respective activities and to calculate a resulting time schedule for the period from start to finish of the construction project,

input/output means for use in inputting data and instructions to the memory means and to the processing means and in selectively retrieving data therefrom for display including structural increment data to depict selected structural increments thereon,

and means for shading structural increments differently in accordance with their scheduled times of completion.

11. Structural graphics system according to claim 10, wherein instructions and data are retrievably stored in the memory means, for effecting diverse shading of these structural increments scheduled to have been completed before a selected time, those increments scheduled for completion then or within a selected time thereafter, and those increments not scheduled for completion until later.

12. Structural graphics system according to claim 11, wherein the stored instructions for time-dependent diverse shading prescribe the boldest diverse such shading for depiction of those structural increments due for completion at—or within the selected time after—the given time, and less bold such shading for increments not due until later.

13. Structural graphics system according to claim 12, wherein stored instructions for monochromatic shading prescribed boldness in terms of longitudinal continuity and lateral thickness of line.

14. Structural graphics system according to claim 12, wherein stored instructions for polychromatic shading prescribe boldness in terms of colors ordered in selected hues and brightnesses.

15. Structural graphics system according to claim 11, wherein stored instructions prescribe diminished intensity of depiction for a structural feature to be removed during the project.

16. Structural graphics system according to claim 11, wherein stored instructions prescribe intermittent switching of depiction, for a structural feature to be placed during the project and then removed later during the project, between one such diverse shading for the feature's placement time and another such diverse shading for its removal time.

17. In means for construction of a building, bridge, dam, industrial plant, or means of transport,

whereby a schedule of discrete work activities is prepared for constructing corresponding increments of structure of the project,

and whereby depictions of such resulting structural increments in construction drawings are also prepared,

the improvement comprising

means for depicting the corresponding structure as in such construction drawings, as of a given time within the schedule,

including means for shading the respective structural increments with degrees of boldness corresponding to respective time relationships relative to the given time.

18. Construction project means according to claim **17**, including means for displaying such depiction as a non-transient image.

19. Construction project means system according to claim **9**, wherein a structural increment is scheduled for removal, and including means in the shading means for showing it shaded in diminished intensity.

20. Construction project means according to claim **18**, including as display means a printer.

21. Construction project means according to claim **17**, including means for displaying such depiction as a dynamic transient image.

22. Construction project means system according to claim **12**, wherein a given structural increment is scheduled to be placed before being removed, and including means in the shading means for showing it alternately shaded according to its placement time and its removal time.

23. Construction project means according to claim **21**, including as display means a video screen.

24. Means for use in construction of a building, bridge, dam, industrial plant, or means of transport, wherein work activities are performed to build increments of structure according to drawings thereof and a time schedule therefore, such means comprising

means for storing the schedule and the structural drawings for retrieval,

means for retrieving such structural drawings corresponding to the status of the project as of any given time in the schedule, and

means for shading the respective structural increments as of such given time with degrees of boldness corresponding to selected construction characteristics.

25. Construction project means according to claim **24**, including in the storage means a read-write memory containing data defining the actual construction.

26. Construction project means according to claim **25**, including means for comparing the read-write memory of the actual construction with the read-only memory of the scheduled construction in terms of the selected construction characteristics.

27. Construction project means according to claim **24**, including in the storage means a read-only memory containing data defining the scheduled construction.

28. In means for construction of a building, bridge, dam, industrial plant, or means of transport, wherein certain work activities are performed to build discrete increments of structure according to a given time schedule, the improvement comprising

comparing scheduled and actual structure of such increments at a given time in the schedule during actual construction,

depicting most boldly incomplete structural increments scheduled for completion within a selected incremental period of time from the given time, and

altering the construction rate to bring the completion of incomplete structural increments into closer agreement with their scheduled construction times.

29. Project construction process according to claim **28**, including enlarging work crews to increase the actual rate of construction.

30. In means for construction of a building, bridge, dam, industrial plant, or means of transport, wherein certain costs are budgeted and work activities are performed to build discrete increments of structure according to a given time schedule, the improvement comprising

comparing scheduled and actual structure of such increments at a given time in the schedule during actual construction,

depicting most boldly incomplete structural increments scheduled for completion within a selected incremental period of time from the given time but running over budgeted costs, and

altering the costs of work activities for incomplete structural increments to bring the costs of completion into closer agreement with their budgeted construction costs.

31. Project construction process according to claim **30**, including reducing work crews to decrease their contribution to construction costs.

32. In accomplishing a building construction project wherein discrete work activities are performed to construct increments of structure of the project according to drawings of such structural increments and according to a construction schedule, the steps of

depicting as of a given time the existing extent of the structural increments under construction, shaded with more or less boldness to indicate more or less discrepancy between their actual extent and scheduled extent when behind schedule, and

takin appropriate steps to reduce such degree of discrepancy of the most boldly shaded structural increments.

33. Building construction method according to claim **32**, wherein such appropriate steps include reducing the overall time to construct the most boldly shaded structural increments.

34. Building construction method according to claim **33**, wherein such appropriate steps include increasing the actual man-hours beyond the man-hours scheduled for the work activities to construct the most boldly shaded structural increments.

* * * * *