A system study of a computerized grocery store

Neal Joseph Martini

,66

A SYSTEM STUDY OF A COMPUTERIZED GROCERY STORE

BY

NEAL JOSEPH MARTINI, 1946-

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

1971

Approved by

Jarvin M. Taylor (Advisor) Paul D. Stigall

James W. Joiner

ABSTRACT

This thesis is a system study of a computer controlled grocery store. Previous work has been done in the area of computerized inventory control, and computerized checkout lines in grocery stores, but the system proposed in this thesis is to control the movement and handling of the products as orders are automatically filled.

In the preliminary part of the system study, a comparison between a time shared computer service and a minicomputer is done, which results in the selection of a minicomputer for this control system. Then the general control system is presented, and the system interfacing and control for the various peripherals is configured, assuming a minicomputer with a bus structure similar to the PDP-8 machine. Finally, a set of workable minicomputer specifications is developed, and a simulation of the proposed control system is done on the SCC-650 minicomputer.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Dr. J. M. Taylor for his guidance and help throughout this project.

Appreciation is also extended to Patricia Johnson for her typing efforts.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Computer misapplications to control systems are probably as numerous and widespread as their valid applications. Many times the computer is sought merely to appeal to the stock exchange status seekers, or to appease individuals who want to make sure the company doesn't lose pace with the times. In many instances, computer control cannot only turn out to insignificantly benefit a firm, but it can even lessen the overall productivity. Therefore, before performing a system study of a partially computerized grocery store, it is desirable to examine the application by discussing some of the effects it will have.

There are many degrees of computer control - from simple monitoring to complete automation. The following discussion will be concerned with those effects which will be evidenced if the system proposed in the ensueing thesis is physically realized. (Incidentally, many of the following points are also used by Perlman[1] to justify the application of a computer to a warehouse materials handling system.)

A. Why Computerize?

What are the advantages of computer application to the small grocery store?

1. Labor Saving
2. Space Saving
3. Tighter Inventory
4. Improved Customer Service

5. Decreased Breakage and Pilferage
6. Decreased Utilities Cost
7. Better Management Control

Labor - As stated by Perlman[1], the typical cost per man-year for labor ranges from about $8,000 to $10,000, and is steadily increasing. The labor force referred to here is typists, clerks and stock boys. Application of the computer allows most of the paper work to be done automatically.

Space - Space saving results from the fact that no elaborate displays are necessary, since the customer doesn't actually view the products on the shelves. This allows compactness and, therefore, space savings. Also, since goods are handled by automatic dispensers, they can be stacked higher per square foot of floor space since it is no longer necessary for a man to reach the goods directly from floor level.

Inventory - Tighter inventory refers to the fact that the shelf time and overall waste of the goods is reduced to a minimum. The fact that shelf time can be reduced is very important in a store that handles perishable goods. Overall inventory reduction is a result of two things. First, since the computer keeps current inventory status summaries, the amount of reserve stock is controlled without counting goods on the shelf. Secondly, with the application of statistical programming methods, future demands can be forecast for various product lines, thus allowing more accurate merchandise ordering. A good example of inventory control and future demand forecasting is summarized by Hipwell[2], Here a small steel firm, with a desk-sized computer, handles routine stock control and performs accurate demand prediction using exponential smoothing. The application has apparently been very successful.

Service - Customer service is a very attractive feature of the computerized store. As will be discussed later, the possibility of placing an order by telephone in this system is feasible. This type of order input means that the customer just picks up a telephone and dials in his order, and then picks up the order after a short period of time. Since the order is automatically filled by the computerized system, and since billing by mail is easily incorporated (also under computer control), the customer is therefore alleviated from the drudgery of grocery shopping.

Pilferage and Breakage - Decreased pilferage and breakage is obviously realized with a computerized system since the consumer only comes in contact with the goods after purchasing the items. Petty thievery is also minimized for the same reason. The handling system will have to be designed to dispense such fragile things as eggs, etc., but careful design can payoff in less breakage. Also, since billing is done automatically by mail, the need to have cash in the store is eliminated, thus eliminating the threat of robberies.

Utilities - Decreased utilities cost can be realized for several reasons. First, an automated store does not require very much lighting, since machine - not man - roams its quarters. The display freezers which exist in most grocery stores today are also made more efficient by minimizing the warm air interface. This results from the fact that customer viewing is no longer necessary. There is, of course, the additional loading due to the conveyor drive motors, but typically these motors can be quite small and still handle big loads.

Management - Better management control, as pointed out by Perlman[1], may seem to be an intangible, but it has been one of the major factors in many successful computer installations. By making records immediately available in summary form, time is saved. Also, as mentioned earlier, reliable forecasting and trend prediction aids an individual in management decisions concerning product acquisitions or deletions.

Of course, there are some disadvantages to incorporating a computer into the system. Such disadvantages, as the need for highly technical maintenance, and the total dependence on the operation of the computer, can be softened by such concepts as the one discussed by Mueller[3]. In the article the author eludes to the possibility of incorporating an "equipment failure history" information system. This would allow prediction of failure and more reasonable spare parts ordering.

Before going on, it is necessary to mention an important assumption which is held throughout the remainder of this thesis. Although the system is being studied and customized to the limitations of the small businessman, it is necessary to assume that the initial design and development must be taken on by an organization with sufficient funds, such as a supermarket chain. Design and development of such things as conveying equipment, product dispensers, and software routines can easily amount to from 2 to 3 million dollars, which is far beyond the budget of a small businessman. After this initial work, it is then conceivable that a single unit store could be built for about 1/2 a million dollars.

Another point that should be brought out at this point is the bounds of this thesis. In general, the thesis will not concern itself with discussion of the design or development of any mechanical equipment used in the system. Also, the topics that are included are treated in a systems approach rather than a detailed hardware design approach. Cost of individual devices is omitted in general, unless it is sighted as a determining factor in the selection between alternatives.

II. REVIEW OF LITERATURE

Up to this point, the advantages and some disadvantages of the proposal to computerize a grocery store have been considered. Assuming that this discussion is sufficient to persuade one to seriously consider the proposal, the next step is to choose the type of computer service to employ. The three main alternatives are:

1. Large in-house computer
2. Time-sharing terminal
3. Minicomputer

The large in-house computer can be immediately ruled out for a business whose annual profit is about \$100,000 to \$200,000, for obvious reasons. Rental fees for these units runs in the thousands of dollars per month (System 360 at University of Missouri-Rolla campus costs approximately \$25,000/month for rental). Therefore, the latter two alternatives will be discussed. Before considering these alternatives, it is necessary to define briefly what is meant by a time-sharing system and a standalone minicomputer.

A. Minicomputer - What Is It?

A generally accepted definition of a minicomputer is that it is a small general purpose digital computer, with less than about 16K of memory and costing less than \$25,000[4]. Minicomputers are very much like the large computers of today in architecture and operation, but have smaller word sizes, slower memory cycle times, and smaller memories. Minicomputers are programmable, like large computers, but have smaller instruction sets due to the smaller word lengths. They can also be adequately interfaced with various I/O devices such as high speed printers, tape punches and magnetic tape units.

To help further characterize what a typical minicomputer is, general specifications for a typical unit are given in Table 1. The computer is the Hewlett-Packard 2114A, a 16 bit minicomputer. The data for Table 1 was taken from a survey chart compiled by Theis & Cobbs[4].

MEMORY	CPU
Memory Cycle Time - 2u sec Memory Word Length - 16 Bits Minimum Memory Size - 4K	Instruction Word Length - 16 Bits Number of Bits for Operation Code - 4 Number of Bits for Address Mode - 2 Number of Addressing Modes - 4 Number of Bits for Address - 10
ARITHMETIC OPERATION (Microseconds)	**SOFTWARE**
Store Time for Full Word - 4.00 Add Time for Full Word - 4.00	Assembler - 2 pass Relocatable - Yes Assemblies Available - FORTRAN II, ALGOL

TABLE 1 - Typical Minicomputer Specifications

I/O CAPABILITY	BASIC MAINFRAME COSTS
Data Path Width - 16 Bits Number of Priority Interrupt Levels - 8 Maximum Number of Interrupts - 56 Response Time to Save Registers of Interrupted Program and Initiate New Program Execution - 10u sec	Basic System Price With 4K Memory and ASR-33 Teletype and Power Supplies - $11,950 Same As Above Only With 8K Memory - $13,950
PERIPHERALS AVAILABLE	
Magnetic Tape Unit - $12,500-$15,000 High Speed Paper Tape Reader (300 Characters/sec) - $2,100 High Speed Paper Tape Punch (50 Characters/sec) - $4,100	

TABLE 1 - Typical Minicomputer Specifications (continued)

So it can be seen that the minicomputer is a very flexible machine with fairly fast operating speeds, various I/O interfacing capabilities, and most important of all, priced within the means of a small businessman.

B. Time Sharing - What Is It?

The following brief analogy quoted from a paper written by Professor F. M. Woodworth of the University of Detroit[5], adequately depicts, for the purpose of definition in this thesis, what a time shared system consists of:

> "Time-sharing might be compared to a chess game between an amateur and a professional player. Naturally the professional, who can play at a much more rapid rate, would spend a great deal of time waiting for the amateur to make his move, and in most cases he would respond immediately with a counter move. If the professional player were to play several games at once with a group of amateurs, he could rapidly move from one board to the next; thus he would avoid wasting time while the necessary 'think' time of each amateur would still be available...
>
> ...In order to make this example closer to the actual time-sharing concept, imagine that the various amateur players did not all play at the same rate...Now suppose that after each one had made his move, he raised his hand...He [the professional] would then not go in automatic sequence from one player to the next, but rather would switch around as each player became ready for the next move."

The correspondences to an actual time shared system are, of course, computer/professional player, time sharing terminal/amateur player, and time for I/O or off time/amateur's "think" time. A typical time sharing system that multiplexes signals to and from several remote user terminals is shown in Figure 1. The multiplexer solves the problem of handling the data processing of many users communicating with the computer system simultaneously, and the external memory for paging provides temporary storage for programs while one is being run.

This perfunctory examination of time sharing and minicomputers provides sufficient background for a comparison between the two types of computer services. This comparison will now be made, keeping in mind the limitations and sentiments of the small businessman.

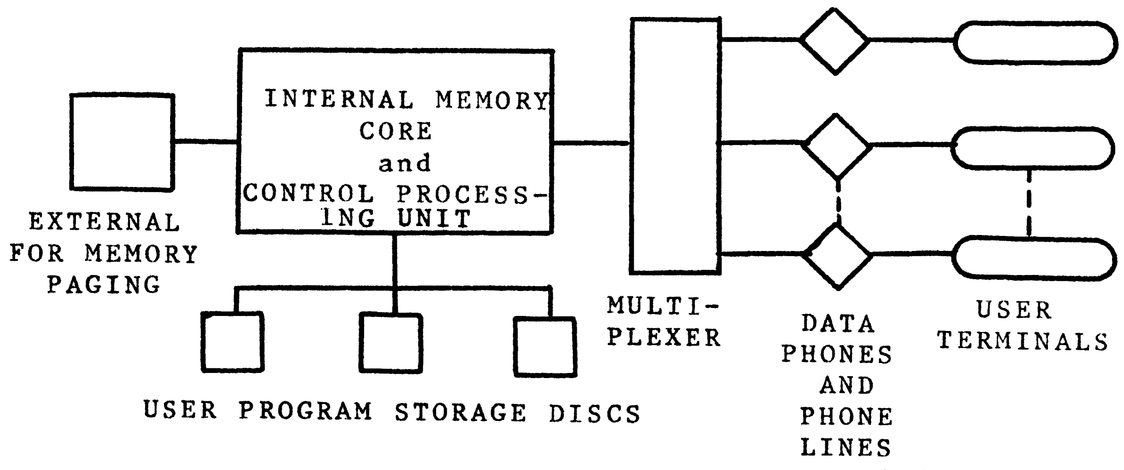

INTERNAL MEMORY CORE and CONTROL PROCESSING UNIT
EXTERNAL FOR MEMORY PAGING
MULTIPLEXER
DATA PHONES AND PHONE LINES
USER TERMINALS
USER PROGRAM STORAGE DISCS

FIGURE 1 - Basic Time Sharing System[6]

C. <u>Time Sharing vs Minicomputer</u>

Edward Yourdon[7], in an article discussing computer applications to small businesses, points out that there are three major fears that the small businessman appears to have associated with computer application:

1. Cost
2. Security
3. Reliability

Time sharing and minicomputers will be compared with respect to these three items.

Cost must necessarily be the first concern to a small businessman. For this analysis, it is assumed that the interfacing, conveying equipment, and the various control devices cost approximately the same, regardless of which system is used. It is also assumed that the tradeoff of semi-skilled employees for skilled employees is about the same.

As will be justified in Section VI, a basic minicomputer, with a 4K memory and a 16 bit word length is necessary for the grocery store application. An ASR-33 teletype is also included to facilitate I/O communication. A very reasonable approximation of this type of system is about $15,000[8]. This includes all power supplies and cabling. Adding another $3,000 to have the unit installed brings the cost to $18,000. If approximately $7,000 is included for initial software assistance, the total cost is approximately $25,000.

For the cost estimate of a time shared system (assuming proximity to a large computer so as to avoid long distance charges on telephone service) three guidelines will be used as stated by McDaniel[6].

> "Averages for initial estimates of cost are roughly $10 to $25 per hour of connect time, including CPU and storage costs. Terminal costs are usually in the range of $60 to $250 per month..."
>
> "... This activity [time sharing] can be accomplished by allowing each user to communicate with the computer via terminal devices having relatively slow input/output rates (10 to 30 characters per second)."

The following assumptions will be made:

1. Average throughput of 30 orders/hour (best case in favor of time shared system)
2. Average order length of 20 items (maximum number specified in Section III is 40 items)
3. Five characters necessary for each product's item code number (results because 10,000 possible different item codes; more details in Section III)
4. Each item code handled twice a day (to include

automatic billing operations)

5. Terminal speed of 20 characters/second, connect time charges of $15/hour and terminal rental charge of $150/month (base on above quote from McDaniel)

6. A twelve hour working day

Thus, based on assumptions 1 through 6, the connect time is 3600 seconds/day. The cost of the terminal connect time per year is therefore $5,500 and the cost of the terminal rental per year is $1,800. Therefore, the total cost per year for the time shared terminal is approximately $7,300. This does not include any cost for initial installation charge and initial programming assistance which can easily run about the same as that of the minicomputer software assistance, i.e., $7,000. Grant it, more conversational programming languages can be used if a time shared system is employed, but remembering the earlier assumption of initial software design by a large chain, any additional programming required can be considered minimal.

It can therefore be seen that an in-house minicomputer can pay for itself in approximately 3 years, which makes it the more attractive alternative to the small businessman. If, however, the savings are not substantial when such things as obsolescence, maintainability, etc., are considered, possibly the next point to be covered - the inherent security in the two systems - will offset some of the advantages of the time shared system.

The problem of security is discussed by E. Yourdon[7] as follows:

> "Whether rational or irrational, a good many businessmen are extremely concerned about the idea of somebody else gaining access to their files. The current approach of providing passwords to each time sharing user does not seem to provide much psychological reassurance to the businessman, who worries that his competitors may somehow discover his password, that employees of the time sharing bureau might permit unauthorized access to the files, and so forth."

Some situations of how the insecurity of a time shared system can cause problems are enumerated by Robert P. Bigelow[9]. He mentions the fact that records are often used in court as evidence, and that their validity would be questionable if the existing time sharing systems were used. Another problem he mentions is when a change is requested by a user but never incorporated.

With a minicomputer everything is right at the fingertips of the businessman. The fact that all of his orders and records are enclosed in a desktop enclosure right in his store, provides great psychological reassurance. Failure to update files, programming changes, and product line changes will be the responsibility of a store employee and not the responsibility of a less concerned third party.

The third point of comparison is reliability. Once again a quote from Yourdon[7]:

> "We can identify a number of common causes of system [time-shared] failures, memory parity errors, input-output errors, program bugs, power failures and so on. Because of these hazards, the typical time-sharing service bureau 'crashes' between one and five times a day, which, while annoying to the

> scientific/engineer user, is accepted as a necessary evil. For the businessman, who is being asked to place his vital business records on a computer, he can't see or touch with his own hands, the prospect of system failure is enough to make the blood run cold."

In a business such as a grocery store, the "crash" of the system is feared not only for the amount of information that is lost during the crash, but also for the length of time the system is down, since the store deals in many perishable goods, as well as "perishable" customers. The advantage of the minicomputer service is that it is a less complicated computer service and, therefore, there are less things to go wrong. There are not such things as multiplexed interfacing, huge memory bank peripherals, and telephone line data paths.

With these few points considered, it appears that a minicomputer is more suited for the grocery store application. If it were necessary to store large amounts of data and perform operations at a very high rate, the time shared system would become more competitive since the cost of additional memory banks and faster cycle times would significantly increase the total cost of the minicomputer system, but would not significantly affect the total cost of the time shared service.

Before going into the actual system employing a minicomputer in an automated grocery store, a few existing minicomputer applications will be described to illustrate some of the potential breadth and depth of minicomputers in control systems.

D. Computer Controlled Mill - Minicomputer Application

The first minicomputer application is the computer controlled rolling mill. Figure 2 shows the system diagram. The system was designed by Carpenter Technology Corporation, Reading, Pennsylvania. The minicomputer used is a PDP-8/1. The job of the computer is to compare set point thickness to actual input and output thicknesses of the strip. The computer then compares the error with the stored allowable tolerances, and generates the necessary correction signals where out of tolerance conditions are detected. Lapidus[10] points out that without the computerized system, production rates of 500 feet per minute and tolerances as fine as 50 millionths of an inch are almost impossible. The system also has the capability of reproducing the profile of a pass, and can print out such things as total number of feet run, and number of feet within - and outside of - tolerance.

E. Numerically Controlled Hole Drilling - Minicomputer Application

The second minicomputer application is concerned with numerical control. The system is a printed circuit board hole drilling unit that was built by Digital Systems Incorporated of Covina, California[10]. It employs a SPC-12 minicomputer, which provides for programs to be developed for drilling any format, and for frequently repeated patterns to be stored to reduce programming. The system is also flexible enough to allow various offsets due to drill size changes used in multiple spindle machines. Naturally,

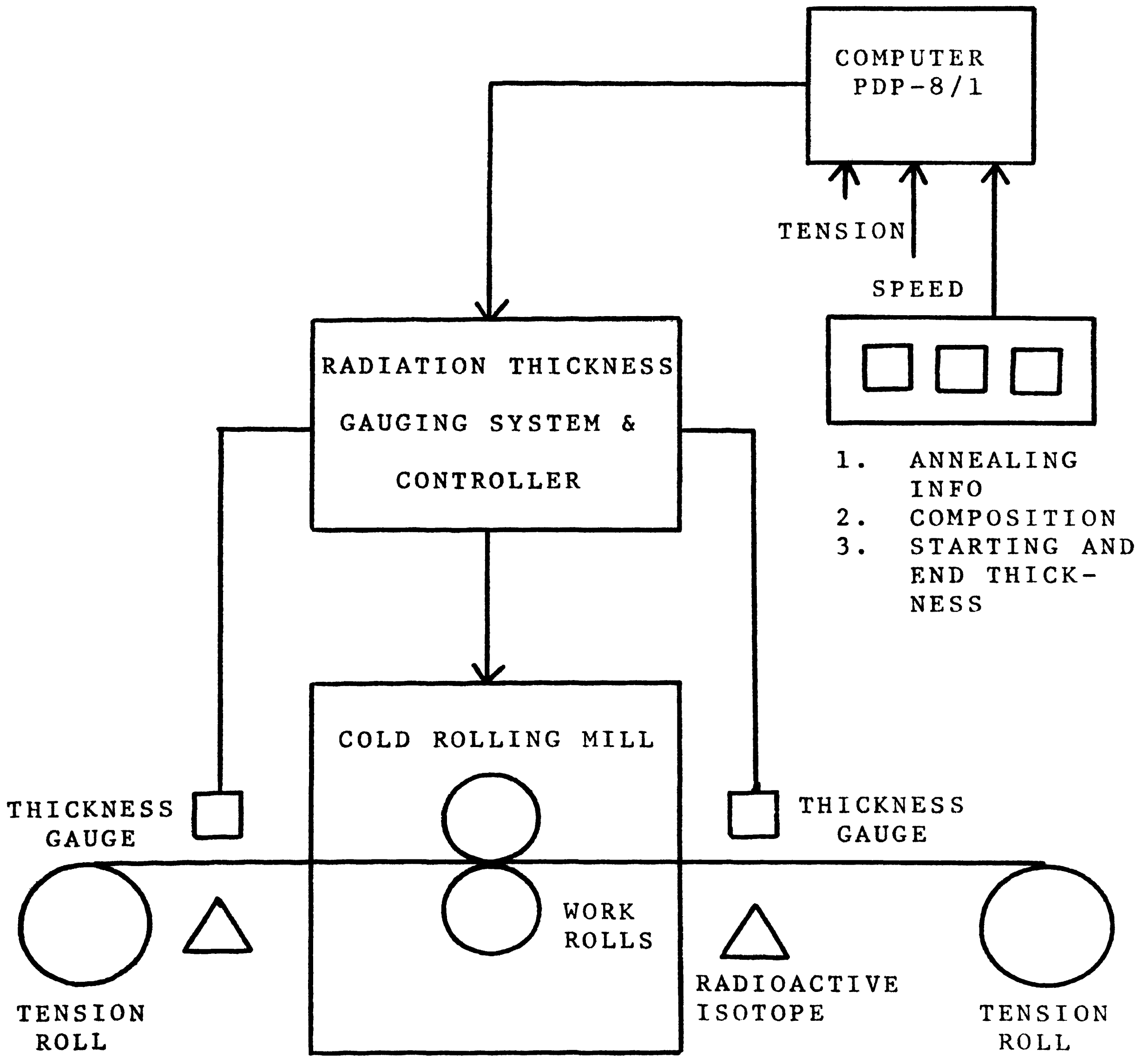


FIGURE 2 - Minicomputer Controlled Rolling Mill[10]

no hardware changes are necessary to facilitate these pattern and offset modifications. Lapidus[10] also points out that users are reporting reductions in cost from about $0.08 to $0.02 per hole, and he eludes to the possibility of further savings by using the computer to optimize the sequence by which the holes are drilled.

F. Other Minicomputer Applications

The American Hoist Company, Oakland, California, has manufactured a batching control system for concrete they call the MARK II Slectron. The system employs Digital Equipment Company's model PDP-8 minicomputer. All the operator has to do in this system, is type out the code of the desired mixture formula and specify how much concrete is to be mixed, and the computer controls the rate and proportion of sand, gravel, water, etc., that goes into the batch. Printouts of total amounts of material used (or remaining), calculations for truck utilization, and demand prediction for various formulae, can all be obtained upon request[10].

Another interesting application of a minicomputer is the one designed by Educational Data Systems (EDS), Newport Beach, California, which is used in secondary school curricula. Here is a combination of the two sources of computer service discussed earlier, i.e., time sharing and minicomputers, into a single time shared minicomputer system. It provides the students with multiple access to calculating packages and is used as a teaching aid. The system uses the General Corporation "NOVA" minicomputer.

A final application of the minicomputer is that of the Food Fair Service in Baldwin Hills, California. This system uses the Honeywell DDP-516 minicomputer and is mainly an inventory control system. At the checkout stations, the checkers input the color code associated with each item and the code number of the particular item. The computer will then display at the checkout, the cost of the item and will keep track of totals, tax, and coupons. Meats are also automatically weighed and priced. Human error is minimized, records are readily available, and such things as ideal product mix for a particular store can be determined[12].

III. PROPOSED SYSTEM FOR SUPERMARKET COMPUTERIZATION

A. Basic Premises

The grocery store that this computer system is to automate is assumed to have 10,000 individually stocked items. If, for example, a certain type of cereal comes in 3 different sizes, this accounts for 3 of the 10,000 items. It is also assumed that the maximum number of items on each order is 40. This constraint doesn't present any limitations. If a customer desires more than 40 items, he can simply make two consecutive orders.

A typical present day supermarket has about 5 active checkout lines, each of which tally and pack an order about once every 5 minutes, or 12 orders per hour per checkout line. This results in a throughput of 60 orders per hour. To help justify the cost of the computer system, the minimum computerized supermarket throughput is required to be twice that of the conventional supermarket, or 120 orders per hour.

The following section describes how orders are placed, automatically filled, and picked up by customers, and will illustrate the general operation of the proposed system.

B. From Customer to System To Customer

The portion of the system that receives the orders from the customers is assumed to operate as stated in the following paragraph.

The tools that a typical customer has are a coded membership number, a product catalog, and a telephone. The coded membership number is the identification used when an order is picked up and is also the first number the customer dials into the store after his telephone has been connected to the receiving system. The product catalog is simply a listing of the 10,000 items carried by the store. It may be arranged in alphabetic order and/or some sort of product type grouping. The only thing that is important, however, is that each item be assigned an individual number. After the customer has called the grocery store, has been connected, and has dialed in his membership number, he dials in the numbers from the catalog that correspond to the desired items. Figure 3 shows the basic diagram of the information receiving system. As the various item numbers are dialed in, they are multiplexed into the interpretation system, and the various pulse trains are converted to appropriate binary levels compatible with the paper tape punch

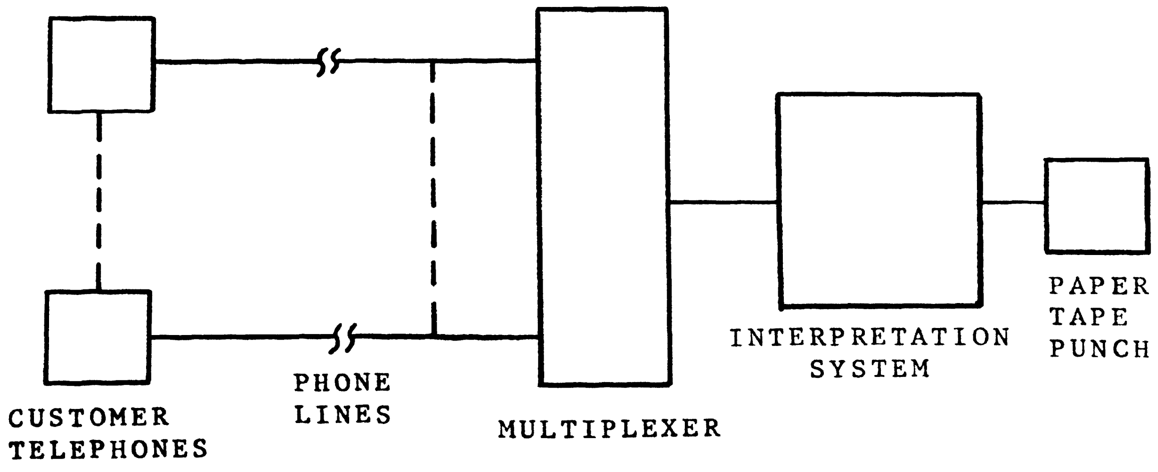


FIGURE 3 - Information Receiving System

control. The orders are then transferred to the tape, in a format compatible with the minicomputer software (see Section VI). This, of course, is a very general description of the possible data receiving system.

The bounds of this thesis include the store's order filling control system, and exclude the particular method that places the orders on paper tape. Such details as what is done if the customer dials in the wrong item by mistake, or what is done if a customer wants to cancel an order, are therefore not considered in this paper. It will therefore be assumed from this point on that the orders have been put on paper tape by some receiving system, and are available for input to the minicomputer used to fill the orders.

The question may arise as to why the input system was chosen to be off line, i.e., why have the intermediate transfer of orders to paper tape instead of going directly to the minicomputer. The major reason for this is to make a permanent record of orders available. This permanent record is used as input to billing calculation programs, and other software routines such as demand prediction, and also provides security to the store owner. Another reason for this off line approach is to avoid the situation where a customer has to wait a long time to get on line.

Now that the orders are assumed to be available on paper tape, the next point of interest is the movement of goods in the grocery store. Figure 4 shows the general layout of the store.

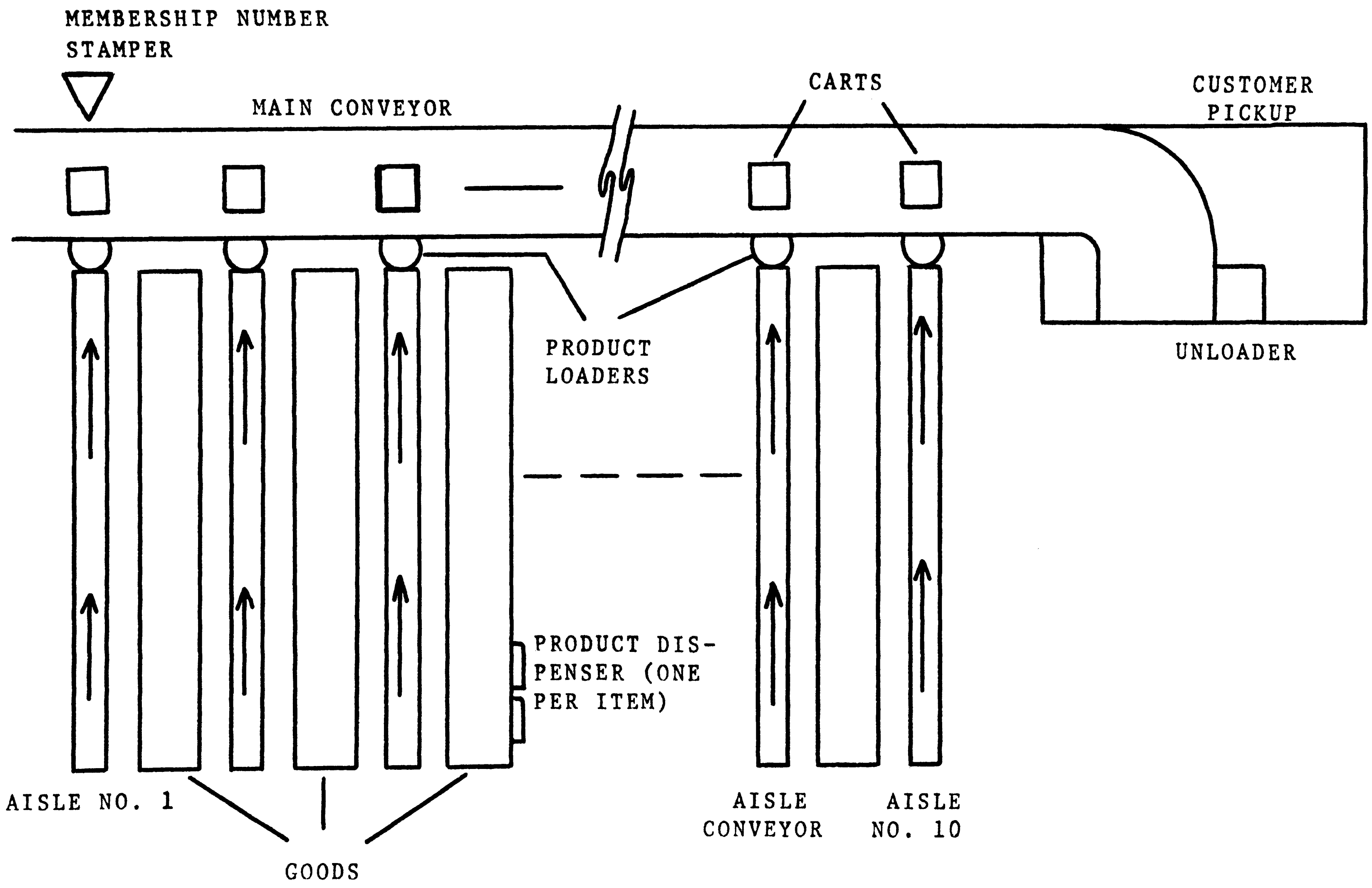


FIGURE 4 - Store Layout

The main conveyor carries the carts, each of which are loaded with the contents of an individual order. Under computer control, the process begins when the membership number stamper prints the customer's coded membership number on the box contained in the cart located at aisle No. 1. Each of the ten carts are lined up with one of the ten product loaders. Each of the ten aisles contains 1000 of the stores 10,000 items. Once the main conveyor has stopped, the computer compares the items in the individual orders with that of the items in the various aisles, and where matches occur, appropriate product dispensers are actuated. Then the items are loaded onto the aisle conveyors. These conveyors run continuously, and transport the items (maximum of 40 items on any one conveyor at one time) to the product loaders, from where they are loaded into the appropriate carts. The main conveyor then steps all the carts to the next aisle and the matching, dispensing, and loading process takes place again, with the computer keeping track of which cart is aligned with which aisle. In this manner, ten orders can be serviced simultaneously.

In the event that a product is out of stock, the computer automatically detects this condition and punches on paper tape both the code number of the item and the membership number of the customer who requested the item. This tape is used as input to the billing calculations program, and therefore prevents a customer from being charged for an item he never received.

Once a cart has stopped at all ten aisles and has been loaded with the requested items, the cart moves to the customer pick-up section, where the box in the cart is either manually or mechanically unloaded. The cart, with another empty box, is then conveyed back to the starting point in the process. When a customer comes to claim his order, a simple check between his membership number and the number on the particular processed order is all that is necessary.

Some mention should be made here regarding the choice of 10 aisles as opposed to some other number. The reasonableness of the layout becomes apparent when it is considered that it takes approximately 15 seconds for the comparison process at one stop of the main conveyor (see Section VI). As will become apparent when the programming philosophy is discussed in Section VI, the comparison process and the conveying of goods down the aisles, take place simultaneously. It is therefore desirable to have the conveying time from the farthest point on the aisle conveyor to the cart be about 15 seconds. Therefore, if it is assumed that a maximum of 4 feet/second aisle conveying speed is acceptable, then the maximum length of the aisle can be 60 feet, which appears reasonable for holding 1000 of the store's items. Therefore, 10 aisles, 60 feet long is a sufficient layout to keep conveying time about 15 seconds. Decreasing the number of aisles would increase the necessary operating speed of the aisle conveyors because the aisles would be longer. Increasing the number of aisles would make it possible to convey in less than 15 seconds, but at the expense of additional handling equipment (conveyor, product loaders, etc.).

Finally, at the end of approximately a twelve hour day, the programs for billing calculations and mailing label printing can be input to the computer. The paper tape order records are used as input data to these programs, and may also be used, as mentioned earlier, in conjunction with demand or trend prediction.

C. Functions of Minicomputer and Peripherals

Summing up what the computer has to control, the following list contains all the operations which need to be monitored or controlled in the system:

1. Input orders from paper tape
2. Stamp code number on box in cart
3. Initiate main conveyor stepping
4. Perform comparison between items in various aisles with items on various orders
5. Actuate appropriate dispenser when a match is found
6. Keep track of where the carts are in the process
7. Sound general alarm if necessary (as of yet, not discussed)
8. Punch tape when out of stock condition encountered

The block diagram of the computer and the peripherals to control these operations is shown in Figure 5. The following will describe the functions of the various peripherals

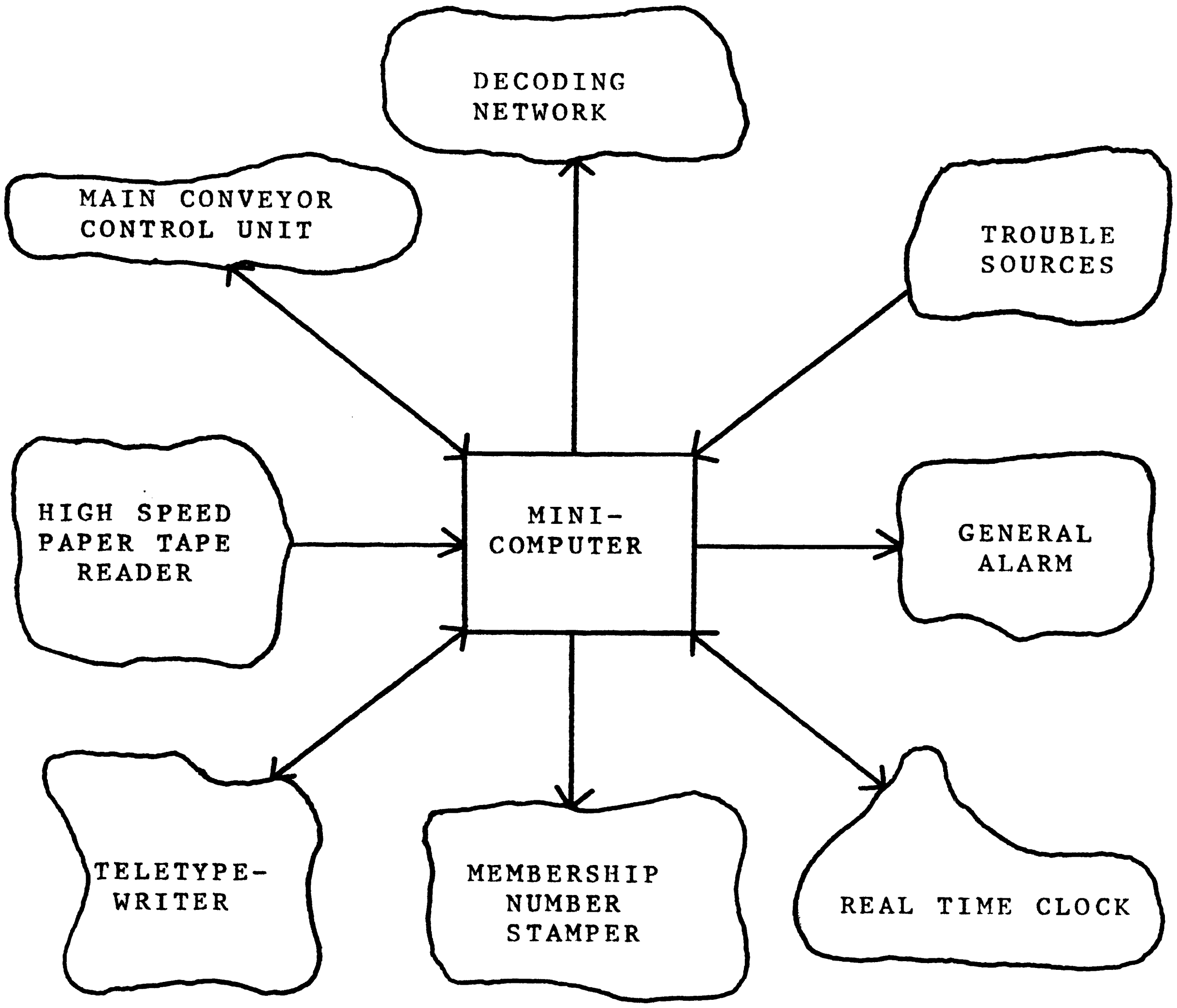


FIGURE 5 - System Diagram of Computer With Peripherals

shown in Figure 5, and also the rationale behind their selection.

The membership number stamper has for its sole purpose to automatically identify the particular order in each cart by stamping the customer's membership number on the box contained in the cart.

The teletypewriter is included for general I/O operations. Once the system is operating as an order processor, the teletype will not normally be used. It will be used, however, in the evenings when billing calculations and mailing label printing are done. Also, the paper tape punch, integral to the teletype, is needed to punch out the customer's membership number, etc., when an out of stock condition is detected.

Paper tape has information arranged in bytes of 8 bits each as shown in Figure 6, the presence or absence of a hole corresponding to a binary "1" and "0", respectively.

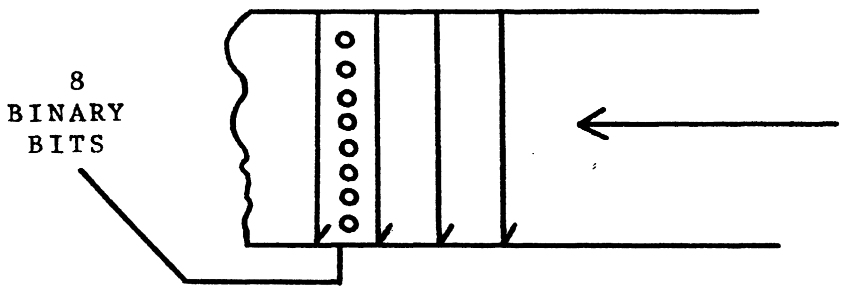


FIGURE 6 - Paper Tape Data Format

Assuming a minicomputer with a 16 bit word size, two 8 bit bytes of paper tape have to be input for every desired word of input. Typically, the operating rates of a high speed paper tape reader, and a low speed tape reader (integral to the teletypewriter) are 300 bytes/second and 5 bytes/second, respectively. Since in the proposed grocery store, at the beginning of the day, approximately 2K of memory (see Section VI) has to be loaded, and at night replaced by programs for billing calculations, etc., the length of input time is a very important consideration. The time to input 2K (2000 words) using the two types of readers is as follows:

Low Speed Tape Reader - 13.3 Minutes

High Speed Tape Reader - 13.3 Seconds

So it can be seen that approximately 1/2 hour of input time can be saved per day using a high speed tape reader. This precludes that its cost of $2000 pays for itself in a short time when it is considered that the order processing day is lengthened 1/2 hour.

The main conveyor control unit is simply the means that the computer uses to communicate with the main conveyor drive system. Since the main conveyor's motion (stepping) is initiated by the computer, this control unit has to be able to respond to the start signal given under program control.

The decoding network has for its purpose communication between the computer and the various product dispensers. It responds to commands from the computer by causing the appropriate product dispenser to place an item on an aisle conveyor when a match is detected in the product-order comparison process.

The trouble sources are in general any conditions which warrant that the system stop processing orders until the condition is remedied, or until the operator resets the system. They cause the computer to trigger a general alarm. Some of the conditions that actuate this alarm could be a conveyor or product dispenser hangup, an out of tape condition on the high speed tape reader, or an out of paper condition on the teletypewriter. When one of these conditions arises, besides triggering the alarm unit, a simple coded message is also printed out on the teletypewriter. Each trouble condition has its own coded message, thus expediting finding the problem. Since the trouble signals enter the computer in parallel on a common interrupt bus, as will be described in Section IV, the only limitation to the maximum number of trouble conditions is the maximum number of device codes remaining after the main peripherals have their codes assigned. As many as 64 interrupts is not uncommon[4].

The final peripheral is the real time clock. Its purpose is to tell the computer when to initiate the main conveyor's stepping motion. It allows sufficient time for worst case aisle loading conditions. The purpose of computer control instead of direct timer control of the conveyor, is to keep the system sychronized. For example, if the timer is directly connected to the conveyor control, and one of the trouble conditions occurs, the main conveyor goes right on stepping at the predetermined interval and, as a result, causes the computer to lose track of where the various orders and carts are.

Now that the functions of the minicomputer and its peripherals have been discussed, the next section is concerned with the general system interfacing and control necessary to allow computer-peripheral communication.

IV. SYSTEM INTERFACING AND CONTROL

A. Interfacing Features of Minicomputer

For the purpose of discussion in this section, the minicomputer used in this control system will be assumed to have the bus structure shown in Figure 7[13], which is the bus structure used in Digital Equipment Corporation's PDP-8 minicomputer.

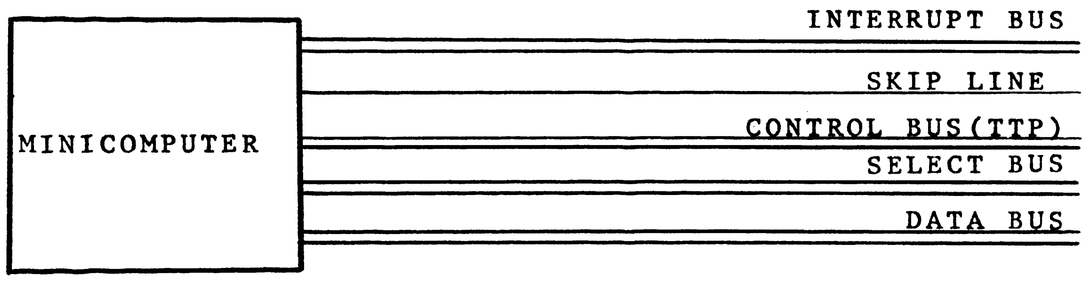


FIGURE 7 - Minicomputer Bus Structure

The following is included as explanation for the workings of the bus network for input-output operations[14]. The various operations will be described first, and will be illustrated diagramatically in Section IVB.

The data bus originates in the accumulator register of the minicomputer, and has as many individual lines as there are bits in the accumulator. All I/O devices that deal with data transfers are connected in parallel to the data bus. The bus is used for transfers in both directions. Its purpose is therefore to make data simultaneously available to all units connected to it for an output operation, and to be available to all units simultaneously for an input operation.

As a result of this simultaneous availability, it is necessary to be able to distinguish which device is to be taking part in the particular I/O operation. The device selection bus handles this differentiation by means of device code transmissions. The desired device code is transmitted over this bus to all units simultaneously, but only the requested device will respond to a particular code, by means of a decoder integral to each device. Typically, the device selection bus has from 6 to 8 lines, allowing 64 to 256 device selection codes. This bus has its origin in the instruction register of the minicomputer, and is the address portion of the input-output instruction.

The purpose of the control bus (TTP) is to tell the selected device what to do, and when to do it. The TTP stands for "transfer timing pulse". These pulses are available to all devices simultaneously, but again only the device that is enabled by a selection bus transmission will respond to the TTP's. The TTP bus originates in the control section of the minicomputer. This control section is programmed, in a sense, by the address portion of the input-output instruction. In other words, the presence or absence of a "1" in certain bit positions of the instruction causes the desired sequence of TTP's to be generated. In the PDP-8, for example, there are 3 TTP's available upon execution of an I/O instruction. Which of these pulses is actually output is determined by whether or not a "1" is present in each of the last three bit positions of the instruction word.

Therefore, it can be seen that to output data to a particular device the data word must first be loaded into the accumulator, then the device must be activated over the device selection bus, and then the transfer of the word from the accumulator to the device takes place over the data bus, under the control of the TTP bus. For input of data the process is similar, but can take place only when the particular device is ready to input. Since the minicomputer executes instructions internally at such a high rate compared to its peripherals, there must be some means of synchronization between the computer and the much slower peripheral devices. The following discussion of these methods of synchronization will illustrate the purposes of the interrupt bus and the skip line, yet to be examined.

Two methods of synchronization appropriate for this control system are the status request method, and the interrupt method.

The status request method is a scheme that checks the state of a device flip-flop. This flip-flop, when set to "1" by the device may indicate, for example, that data is ready to be input. The setting of this flip-flop by the device is referred to as "raising its flag". The computer checks for the condition of a raised flag by an executed instruction which selects the desired device and gates the output of the status flip-flop to the "skip line". If a flag is raised, a signal appears on the skip line and causes the computer to skip the next instruction in its sequence. If no flag is present, the next instruction is executed. The following instruction sequence illustrates its synchronization capabilities:

Instruction No. 1	Test flag of device xx
Instruction No. 2	Jump back to instruction 1
Instruction No. 3	Perform desired operation

It can be seen that until the particular device is ready (flag raised), the computer loops between instruction No. 1 and 2. But once the flag is raised, instruction No. 1 causes the computer to skip to instruction No. 3 and the desired operation takes place.

The interrupt method is a scheme by which a particular device tells the computer when it's ready for action, unlike the status scheme where the computer asks the device. Employing this method allows the computer to continue in program execution while a device is getting ready for an I/O operation. A summary of the sequence of events that occur once a device causes an interrupt are enumerated by Flores[13] as follows:

1. Suspend all interrupts (this prevents other devices from interrupting after the first interrupt occurs)
2. Save registers for future return to main program
3. Go to an interrupt service routine, and perform desired operations
4. Restore registers and return to main program, when

possible

5. Restore interrupt capability so as to allow other device interrupts to occur

Figure 8 shows the breakdown of the interrupt bus shown in Figure 7 into the individual lines that carry out the steps enumerated by Flores. This figure shows control for one typical device, and must therefore be duplicated for each interrupting device.

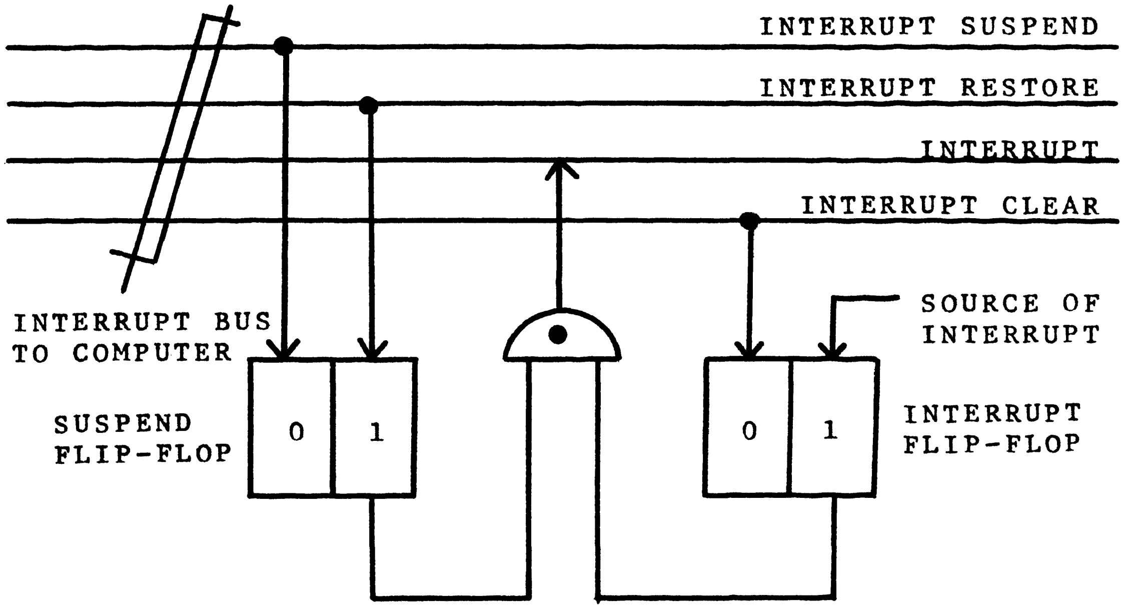


FIGURE 8 - Interrupt Control[13]

It can be seen that a signal on the interrupt suspend line inhibits any other interrupt from putting a signal on the interrupt line after the first interrupt is received. The interrupt restore line naturally restores the suspended interrupt capability. It can also be seen that once the source of interrupt is no longer present, the interrupt flip-flop can be reset by means of the interrupt clear line.

B. Grocery Store System Interfacing

Now that the various bus networks and synchronization schemes have been discussed, an illustration of the interfacing for the grocery store peripherals will be presented, employing these features.

Teletypewriter and High Speed Tape Reader - The teletypewriter and the high speed paper tape reader have been integrated into many minicomputer systems, and the standard interfacing as illustrated by Flores is shown in Figure 9. It should be noted that the teletypewriter consists of four of the control systems shown in Figure 9. This is so because it has four independent units in it, i.e., printer, keyboard, paper tape reader, and paper tape punch. Each has its own selection code, decoder, and data buffer. To illustrate the operation of the system, the function of the printer will now be discussed.

The execution of a selection instruction causes the decoding network of the printer to output an enable signal. Assuming now that the printer has been selected, and it is desired to check the status of the printer's "flag", the execution of the instruction to check the flag causes the gate network to output TTP 1. This in turn causes a signal to be placed on the skip line if the flag is raised. Suppose it is now desired to output to the printer the contents

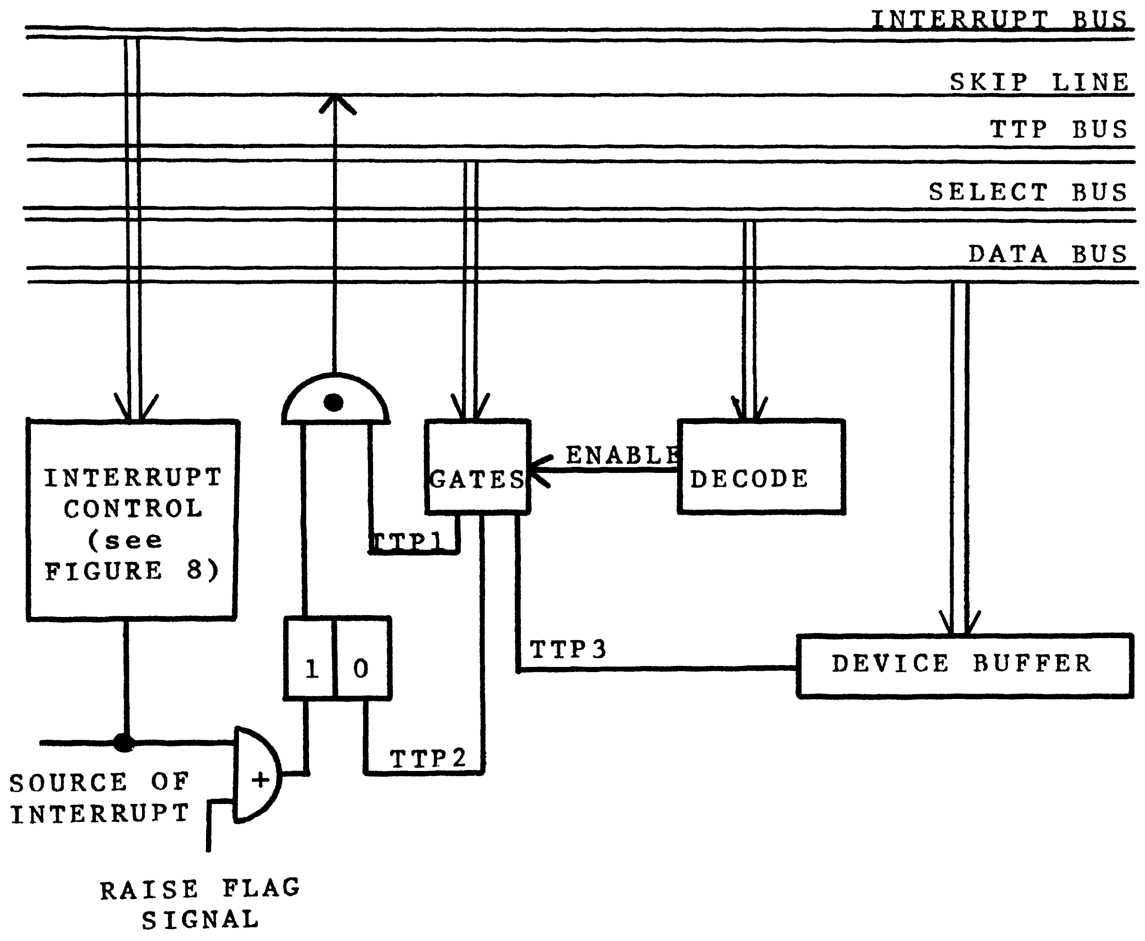


FIGURE 9 - Teletypewriter and High Speed Paper Tape Reader Interfacing[13]

of the accumulator. Again, the device having been selected, the execution of the output instruction causes the gate network to output TTP 3, which causes the loading of the printer's buffer with the desired data. In like manner, TTP 2 clears the raised flag when the appropriate instruction is executed.

The operation of the interrupt control, shown in Figure 9, is the same as discussed previously, and it should be noted that the device flag is raised if an interrupt occurs, so as to allow detection of which device is interrupting.

Main Conveyor Control - The main conveyor control interfacing is shown in Figure 10.

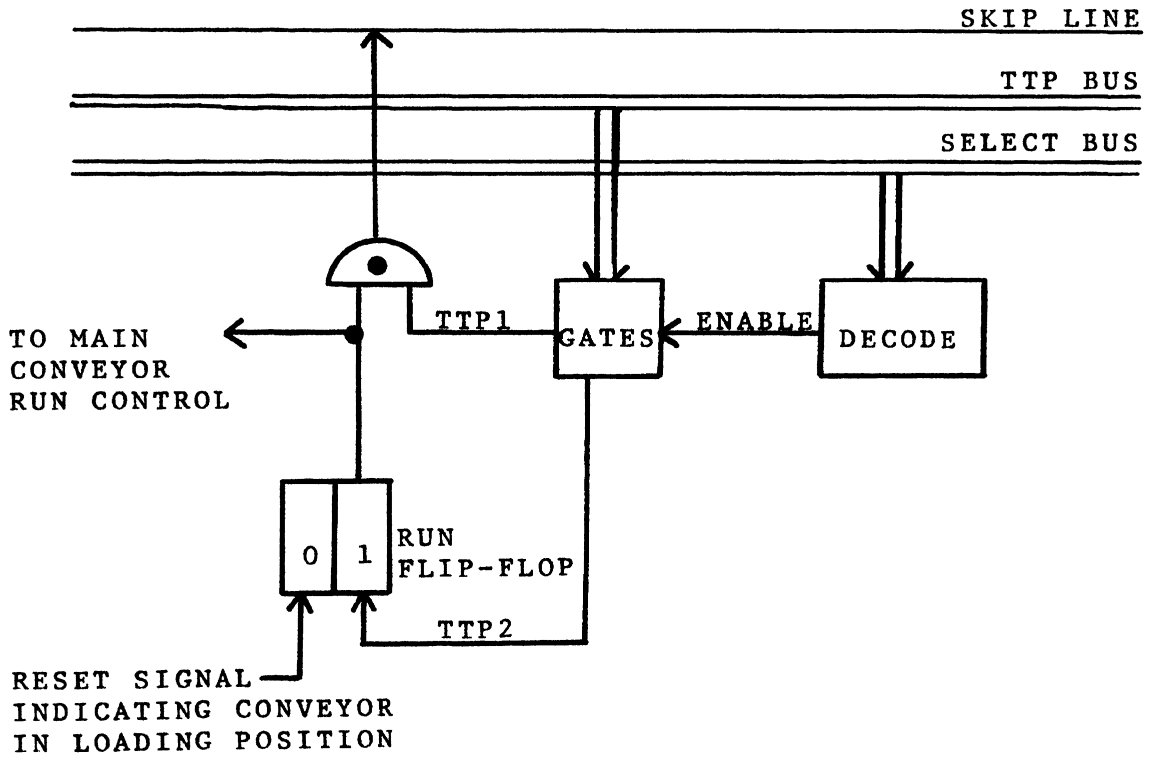

SKIP LINE
TTP BUS
SELECT BUS
TTP1
GATES
ENABLE
DECODE
TO MAIN CONVEYOR RUN CONTROL
0
1
RUN FLIP-FLOP
TTP2
RESET SIGNAL INDICATING CONVEYOR IN LOADING POSITION

FIGURE 10 - Main Conveyor Control

The TTP 1 allows the computer to check the status of the main conveyor in a similar manner to the flag test described earlier. The TTP 2 results from the instruction which desires that the main conveyor be set in motion, by causing the run flip-flop to be set. Once this flip-flop is set, the conveyor runs continuously, and the computer is free to go on about its business. The signal to reset the run flip-flop, and therefore stop the main conveyor, is independent of computer control, and comes from a device which senses that the conveyor is in proper position for aisle loading.

Decoding Network - The decoding network interfacing is shown in Figure 11. A typical output operation in this

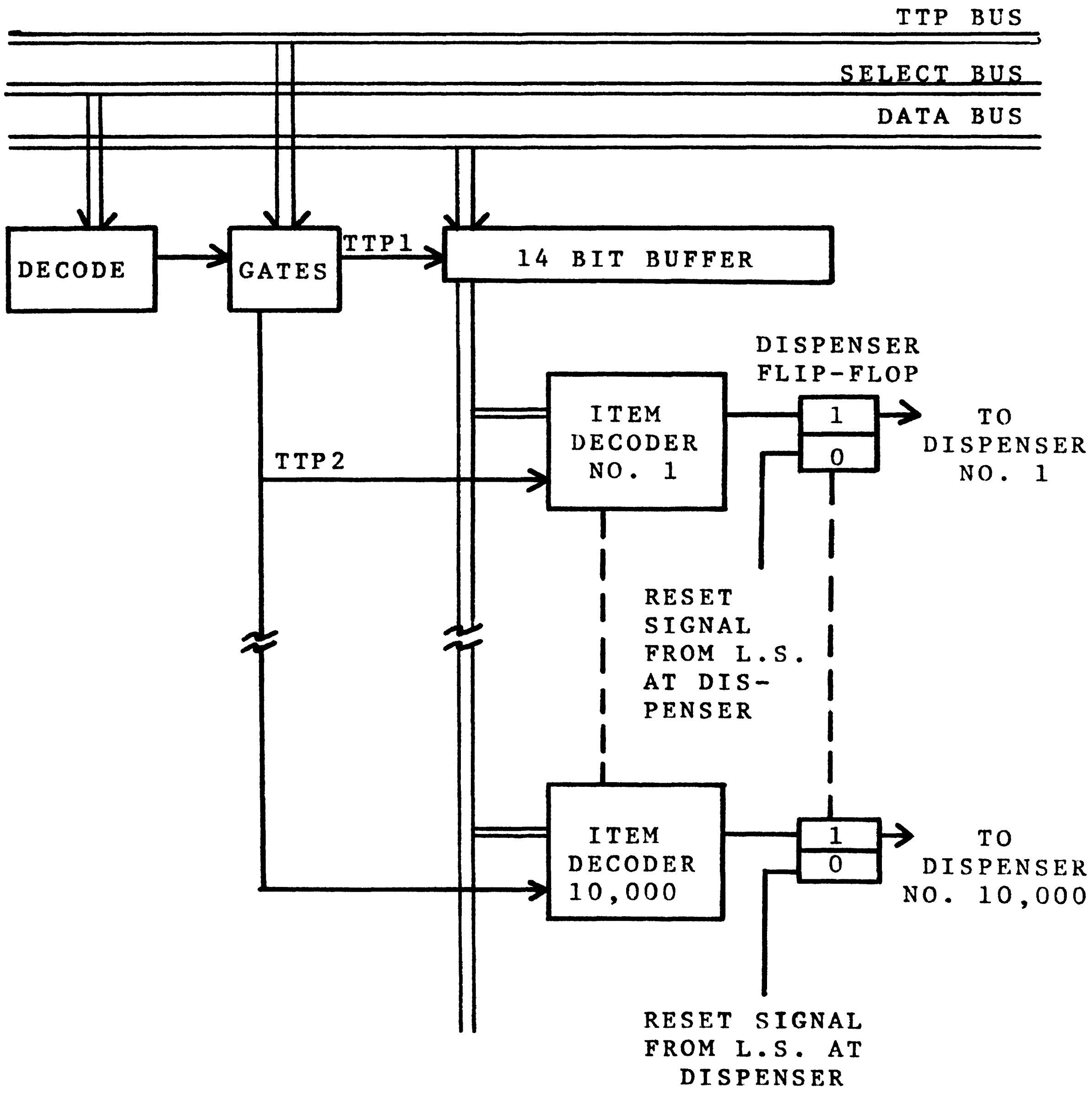


FIGURE 11 - Decoding Network Interfacing

network starts with the selection of the decoding unit, thus enabling the TTP gates. Upon execution of the output instruction, the 14 bit data word (representing one of 10,000 item codes) is loaded into the 14 bit buffer at TTP 1. The individual item decoders are then enabled at TTP 2 and the appropriate flip-flop is set depending on which item code was output. The setting of this flip-flop causes the particular dispenser to go into motion and loads the desired product on the aisle conveyor. Once the flip-flop is set, the computer is free to output the next item code number, and the cycle repeats. The individual dispenser flip-flops are reset by a limit switch which is activated by the dispenser's motion.

As mentioned earlier, if an item runs out of stock, the computer is to punch on paper tape the customer's membership number and the code number of the particular item. To allow for the capability of detecting this out of stock condition, a modification has to be made to each item decoder shown in Figure 11. The modification to one typical item decoder is shown in Figure 12. This modification allows a flag check to be done on the dispensers so that an "out of stock" condition or a "dispenser in motion" condition can be detected. The detection of the dispenser in motion is necessary in order to cause the system to stay in a wait loop when two of the same item are ordered by a single customer (see Section VI).

Trouble Sources - The trouble source system is a

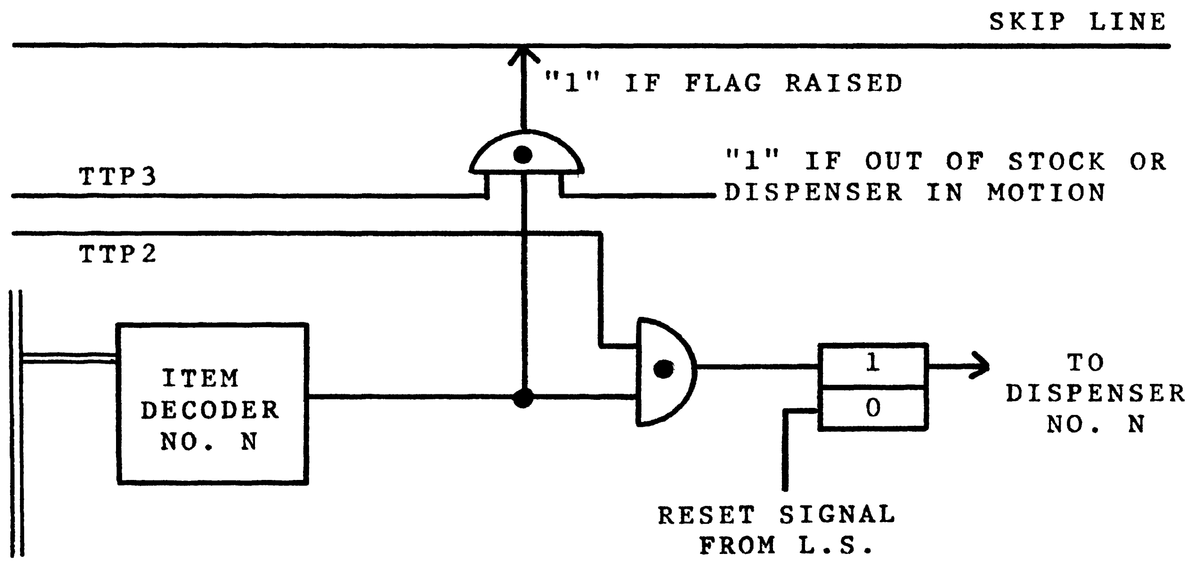


FIGURE 12 - Decoding Network Modification

combination of the status and interrupt synchronization schemes discussed earlier. Each source of trouble is provided with the capability of interrupting the computer, causing it to jump to an interrupt service routine, and also has the capability of having its flag tested. The latter is provided to allow the computer to find out which condition caused the interrupt in the event one occurs. Figure 13 shows the interfacing for a typical trouble source controller. This control must be duplicated for every trouble source which causes the system to stop order processing. It can be seen that on TTP 1 the selected trouble source controller has its interrupt flip-flop gated to the skip line, and as a result communicates its status to the computer.

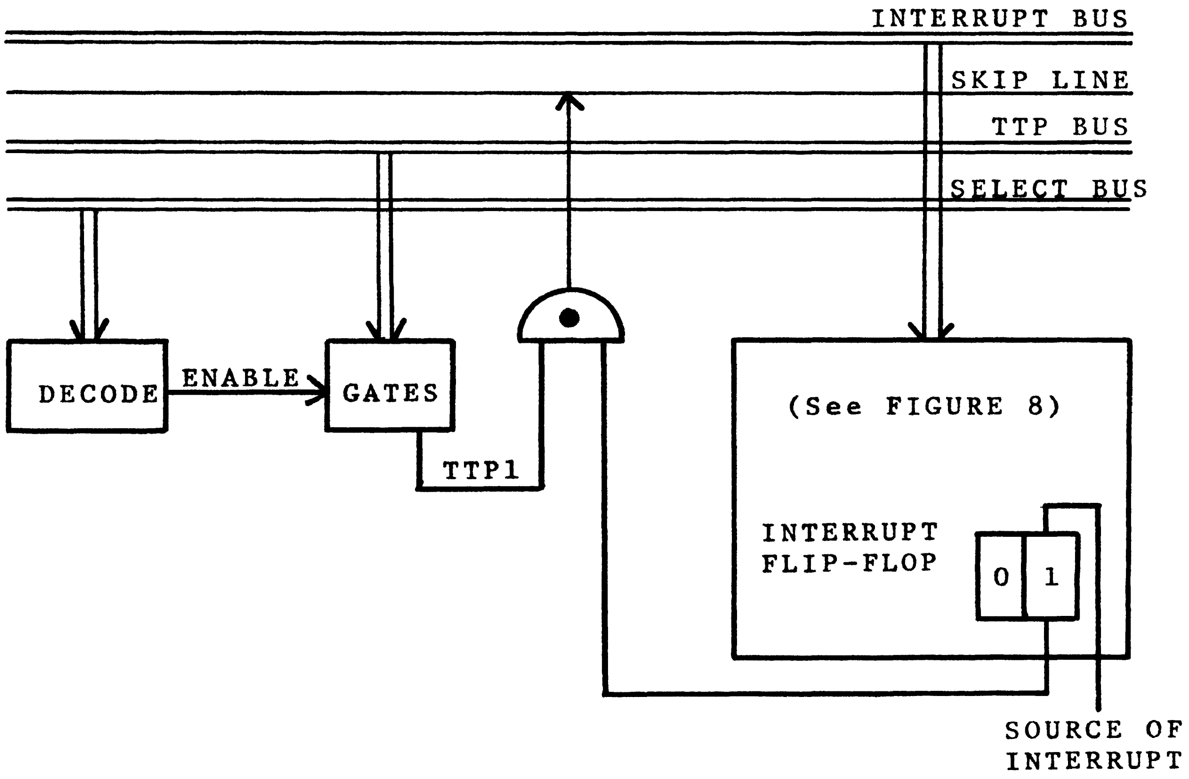


FIGURE 13 - Trouble Source Control

Real Time Clock - The real time clock control is identical to that shown in Figure 13, only TTP 2 is added to allow the clock to be reset under computer control. Therefore, an additional line labeled "TTP 2" goes between the gates and the real time clock. Also, the source of the interrupt is the real time clock output pulse which occurs at the end of a predetermined interval, allowing for worst case loading conditions before the main conveyor is set in motion. It should be noted that this particular interrupt does not cause the general alarm to be activated, but rather causes the conveyor control system to initiate movement by setting the run flip-flop (see Figure 10). Again the interrupt approach is used to allow the computer to go on processing while waiting for the next move of the main conveyor to take place.

General Alarm - The control and interfacing for the general alarm is identical to the main conveyor control (Figure 10) with the exceptions that there is no need for the status check gating into the skip line. Also, the reset signal for the flip-flop is supplied by an operator's pushbutton. This pushbutton allows the operator to acknowledge the alarm and perform the necessary task with the system still being in the halt mode.

Membership Number Stamper - It is recalled that the task of the membership number stamper is to print an identifying number on the various boxes contained in the carts on the main conveyor. To save hardware, the stamping of the customer's membership number is in binary form. The system is shown in Figure 14. Once the stamper has been selected (gates enabled), the execution of the output instruction causes the stamper buffer to be loaded at TTP 1 and then causes the binary printers to stamp the appropriate 1's and 0's at TTP 2. Of course, if desired, each set of 3 binary bits can be treated as an octal digit (maximum value 7), and octal numbers could be output with the proper decoding included.

Now that the general system has been described and the control and interfacing of the peripherals has been discussed, it is necessary to examine the specifications of the

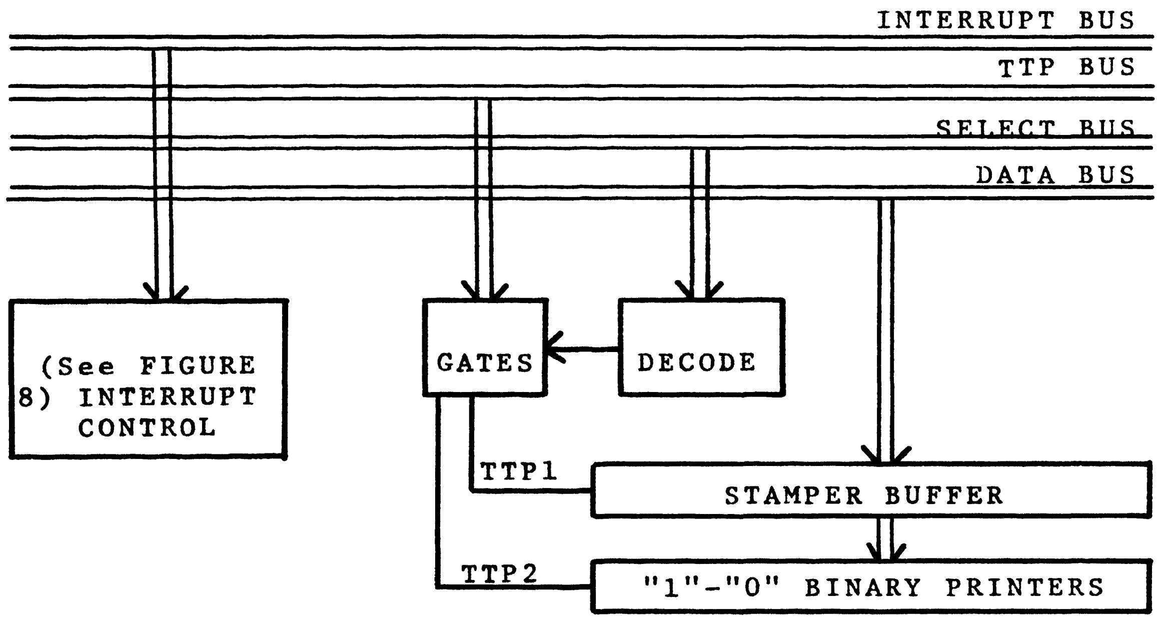


FIGURE 14 - Membership Number Stamper Control

minicomputer itself.

V. MINICOMPUTER SPECIFICATIONS

The purpose of this section is to enumerate some of the basic specifications that the minicomputer used in the grocery store system must have. It is not an attempt to choose any particular manufacturer's minicomputer, nor is it a unique set of specifications. It is simply "a" set of specifications that can efficiently do the job required in lieu of the interfacing, control and general system operation previously discussed and, therefore, a compatible set of specifications would also suffice. The discussion will be divided into three main sections, i.e., memory, CPU and I/O specifications.

A. Memory

In order to represent the 10,000 items in the grocery store by individual binary codes, at least 14 bit binary words are required. In a machine with a word size less than 14 bits, the storage and handling of one of these codes requires two memory locations to store the full code word, and twice the associated register transfers when operating on the code word. This necessarily implies twice the computer processing time per order. The use of a 14 bit or greater word size, however, allows storage and handling of item code numbers in single operations. Since the standard word lengths available today are 8, 12, and 16, the most efficient word size is the 16 bit length, which inherently allows for expansion of the total number of products handled in the grocery store.

The memory cycle time is another memory specification to be determined. It is the total time it takes to read out the current value of a specified memory location and then to write in a word (possibly same one read out) into the specified location. This time is fixed for random access memories like in minicomputers, since it takes the same amount of time to access any of the basic 4K memory locations.

To give an actual value to the cycle time for the computer to be used in the grocery store system, the simulation done on the SCC-650 (see Section VI) is used as a benchmark. Since the simulation performs the critical comparison routine fast enough to maintain the desired throughput of 120 orders/hour, the SCC-650's cycle time of 1.75 microseconds[8] is used as a minimum cycle time specification.

To determine the size (number of words) of the memory, it is noted that very few minicomputers are provided with memories with less than 4K words. Therefore, it is reasonable to assume a lower bound of 4K on memory size. As will be shown in Section VI, 4K memory suffices also as an upper bound for this application. The 4K memory should, however, have the capability of being expanded to at least 8K, in the plug in fashion typical to most minicomputers, thus allowing for future needs.

B. CPU

The size of the accumulator and associated arithmetic registers is now to be chosen. These registers handle the same data that is stored in memory, and also the instruction words. To allow the data words and instruction words to be stored interchangeably in memory, the size of the accumulator and arithmetic registers is fixed at 16 bits.

Before discussing the format of the 16 bit instruction word, it is necessary to briefly describe three basic modes of addressing found in minicomputers. They are here referred to as direct, indirect and relative, but each manufacturer typically has different terminology. Stated briefly, direct addressing causes the operation, specified by the op-code portion of the instruction, to be done on the data in the memory location specified by the address portion of the instruction. Using indirect addressing causes the operation to be done on the data contained in the memory location specified by the contents of the memory location specified by the address portion of the instruction. Using relative addressing causes, the operation to be done on the data contained in the memory location which is "+n" locations away from the location which contains the instruction being executed. An instruction using direct or relative addressing typically takes 2 cycle times (2T) for execution, while the powerful form of indirect addressing typically takes 3T[15].

Since almost all minicomputers come with the direct addressing feature, only the need for indirect and relative addressing is established. At least one of the latter is included because the capability of directly addressing 4K requires 12 bits, which only leaves 4 bits for the op-code portion of the instruction. Besides this, there isn't any allowance for future memory expansion. Indirect addressing solves these problems. Another reason for including indirect addressing is that the program described in Section VI, makes use of pointers and counters to keep track of where the various carts are, and which items are to be compared, and indirect addressing is essential for this type of program control. Relative addressing is also included as a specification because it is faster than indirect addressing and is used whenever possible. Again, the constraint of 3 addressing modes is not very severe as evidenced by the fact that 19 of the 23 sixteen bit machines covered in the survey by Hobbs and Theis[4] had at least 3 addressing modes.

Assuming these 3 addressing modes are provided, two bits of the instruction word format are required to indicate which of the addressing modes is being employed (allows for combination of two modes). The remaining 14 bits are used by the op-code and the address itself. It is naturally desired to be able to directly or relatively address as much of the memory as possible since they are the fastest modes of addressing. It is therefore desired to make the address portion of the instruction as long as possible. As described by Flores[13], the IBM 1130 is a 16 bit minicomputer with a proficient 32 instruction repertoire. If this set of instructions is assumed sufficient for the grocery store control computer, it requires 5 binary bits of the instruction word for specification of these 32 instructions. That leaves 9 binary bits for the address portion of the instruction word, which in turn allows 512 locations to be directly addressed, and 512 possibly different locations to be relatively addressed (forward reference). The resulting instruction word format is as follows:

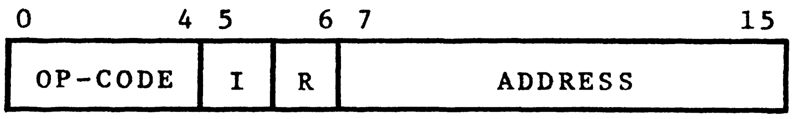


FIGURE 15 - Minicomputer Word Format

The I and R represent the addressing mode bits. It is again emphasized that this is just one possible word format, and that there are certainly other compatible formats that are workable in this application.

C. Input-Output

Since the data bus, which communicates information to and from the various peripherals, originates in the accumulator (16 bit register), it follows that the bus contains 16 lines. This allows an item code number to be transmitted to the decoding buffer in a single I/O operation, and avoids two stage output operations.

The number of lines in the select bus determines the number of uniquely addressable I/O devices allowed (including interrupt sources). This bus originates in the address portion of the instruction register and is therefore limited to a maximum of 9 lines. The maximum number of devices that can typically be paralleled on the bus is 15-20[14], beyond which extenders (signal amplifiers) must be used. Therefore, more than four lines (16 codes) of the 9 lines available can be used if it is desired to have more than 6 interrupt sources, but only with the addition of extenders. Only 6 interrupt sources are allowed using four lines because there are 10 I/O devices in the grocery store system shown in Figure 5.

Since all the sources of interrupts are paralleled onto the interrupt bus, and since the interrupt service program determines the source of the interrupt, only one interrupt level is required. Also, since the time to save registers and initiate execution of the service routine is not critical (process wise), a typical value of 14u sec[14] is assumed sufficient.

The requirements for the TTP's are determined by their applications in Figures 9 thru Figures 14. It is necessary that the computer have at least 3 TTP's available, and that they be programmable by the address portion of the instruction word, as discussed earlier in Section IVA.

Finally the minicomputer must have a skip line to provide a means of synchronization between the computer and its peripherals. When a signal is placed on this line, it causes the location counter of the computer to count two, instead of the normal one, and thus skip the next instruction to be executed.

A summary of the discussed specifications appears in Table 2.

MEMORY	CPU
Memory Cycle Time - 1.75u sec	Instruction Word Length - 16 Bits
Memory Word Length - 16 Bits	Number of Bits for Op-Code - 5 Bits
Memory Size - 4K	Number of Bits for Address Mode - 2 Bits
I/O	Number of Addressing Modes - 2 Bits
Data Bus - 16 Lines	Number of Bits for Address - 9 Bits
Select Bus - 4 ≤ lines ≤ 9	
Interrupt Levels - 1	
Response Time of Interrupt - 14u sec	
Number of IOT Pulses - 3	
Skip Line - 1	

TABLE 2 - Set of Specifications for Grocery Store Minicomputer

The next section of the thesis is concerned with the philosophy of the programming approach. Memory organization, interrupt servicing, input-output software, etc., will be examined assuming a 16 bit machine.

VI. PROGRAMMING PHILOSOPHY

This section describes the general programming, memory organization and timing considerations for the proposed grocery store control system. A simulation of the proposed system using the SCC-650 minicomputer, a 12 bit machine, is also described. The SCC-650 is used because of its availability on the University of Missouri-Rolla campus, and also because of the author's familiarity with its operation. The simulation assumes 4000 items in the store (12 bits allow only 4096 different item codes). Since the criterion of 10,000 items is a constraint only in the comparison routine (described in Section VIC), and since the comparison process is "linear", the time for comparison on the SCC-650 multiplied by 2 1/2 approximates closely the comparison time required for the specified 16 bit machine. The actual programming for the SCC-650 is done using the symbolic programming language SPL-650, and a summary of the mnemonics and their functions appears in Appendix A.

It is also assumed that the 16 bit machine to be used in the actual control system has at least a functionally equivalent instruction set as the SCC-650, which has only a 4 bit operation code field. Also, communications between the ASR-35 teletype and the SCC-650 are used to simulate any peripheral-minicomputer data transfers that take place in the actual grocery store system. Unless otherwise specified, the following discussion refers to the actual

16 bit minicomputer control system.

A. <u>Sequence of Events</u>

The following discussion briefly describes the events that occur in the actual system with respect to the interval diagram shown in Figure 16.

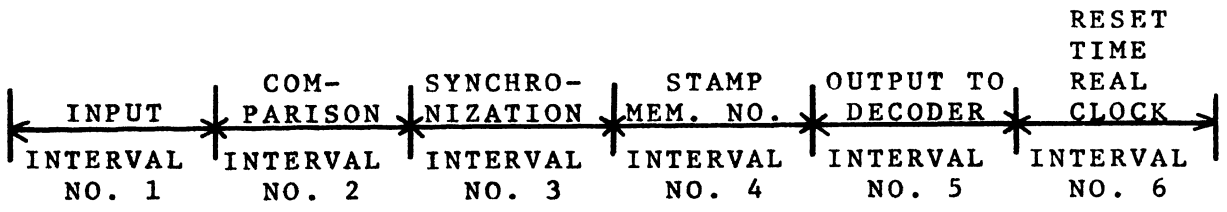


FIGURE 16 - Interval Diagram

<u>Interval No. 1</u> - The customer's order to be handled next by the system is now input from paper tape. The paper tape contains the orders in the following format - membership number, individual items ordered (maximum of 40), and an all zero row indicating the end of the order. In the actual system, each row of punched holes of paper tape contains only 8 bits; therefore, each 16 bit word must be input in two operations, and then the total word must be reconstructed in the computer. In the actual system, the input of one order takes about 1/4 second since a high speed tape reader is employed.

<u>Interval No. 2</u> - The next event to take place is the comparison routine. To explain this process, assume that initially there are no orders being processed. Also assume that the main conveyor is positioned so that the first cart to be serviced lines up with aisle no. 1, as shown in

Figure 17.

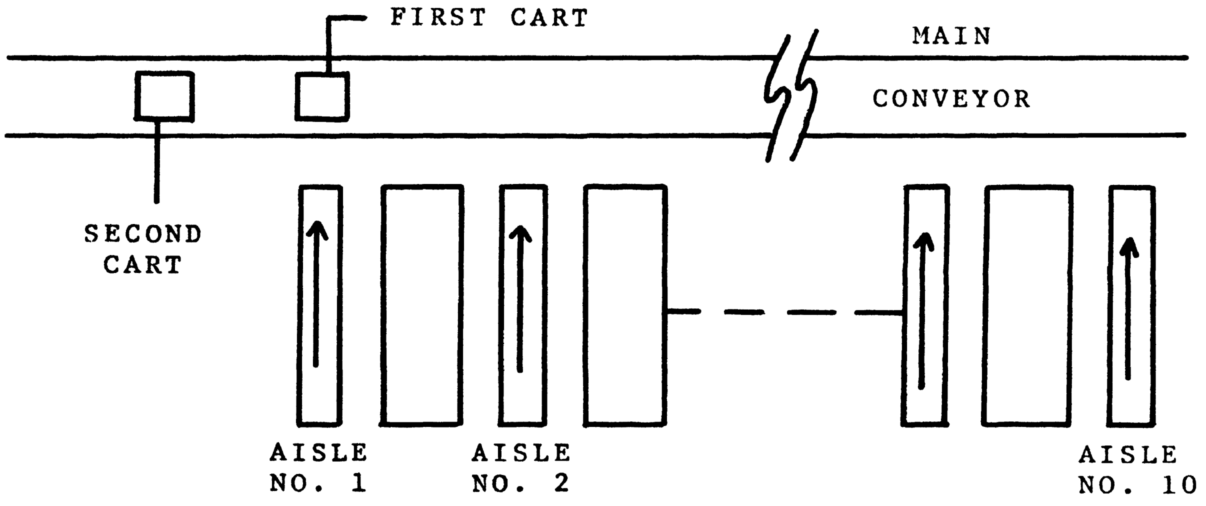


FIGURE 17 - Store Layout Showing Location of Incoming Cart

The item codes on the goods in aisle no. 1 range from 0 to 1750 octal (0-1000 decimal). Therefore, during the first comparison process, each of the possible 40 items in the first order (already input to memory) is compared to the octal numbers 0 thru 1750. When a match is detected, the particular item code is stored in a section of memory termed a "match table". Note that each of a block's 40 memory words is compared regardless if the order had a full 40 items specified or not. In this sense the comparison process is "linear".

Since, for this example, no other orders are being serviced, the item codes stored in the match table are then output to the decoding unit. If other orders are being serviced, their contents is also compared to the items in the various aisles. As a result, the items specified by order no. 1 are dispensed onto the aisle no. 1 conveyor, and are then loaded into the first cart. The main conveyor is then moved so that cart no. 2 aligns with aisle no. 1 and cart no. 1 aligns with aisle no. 2. The next comparison routine therefore compares the items specified by order no. 2 (input during conveyor move) with 0-1750 octal, and the items specified by order no. 1 with 1751-3720 octal (1001-2000 decimal). Therefore, it is seen that each order has its items compared only to the goods in the aisle with which its cart is aligned. In this manner, ten separate orders can be serviced simultaneously.

It should be noted that mention of the output to the dispensers and the inputting of order no. 2 were only made for clarification, and do not occur during Interval No. 2. Their actual position in the sequence of events will become clear as the other intervals are described.

On the SCC-650, the comparison process for a single stop takes approximately 5 seconds. This is assuming 400 items per aisle instead of the actual 1000 items per aisle. Since the comparison process is linear, it can therefore be assumed that the actual comparison process on the 16 bit machine takes approximately 2 1/2 times as long or 12.5 seconds.

<u>Interval No. 3</u> - At this time the system must check and see first if the real time clock has interrupted starting the main conveyor's movement, and second, if the main conveyor

was started, to make sure that it is in loading position. Normally the two events take place during the comparison routine, but this checking section is included to guarantee synchronization in case a trouble source interrupt occurs.
<u>Interval No. 4</u> - At this point, output of data from the computer to its peripherals begins. During this interval, the membership number stamper, stamps the customer's number on the cart aligned with aisle no. 1.
<u>Interval No. 5</u> - During this interval, the various matches detected in the comparison routine are output to the decoding unit. This outputting is done at microsecond rates since the decoding unit simply initiates motion in a dispenser and is then free to output to another dispenser, with two exceptions.

The first exception is if an out of stock condition is detected (raised flag) for a particular item. In this case, the computer punches on paper tape the membership number of the customer who requested the item, and also punches the code number of the item itself. The outputting is necessarily done in split format because there are only 8 bits maximum in one row of paper tape. This tape is then used by the software routine which performs billing and inventory calculations.

The second exception is when a single order specifies more than one of a certain item. In this case, the output process to the decoding unit must pause and wait for the first item to be dispensed before calling for an additional dispensing. Since this is time consuming, individual item codes and dispensers are provided whenever possible for different quantities of a certain item; for example, one dozen of eggs and one-half dozen of eggs should have different call number item codes. This, of course, can only be done up to a point, and a tradeoff between speed of order processing and necessary hardware to implement individual item codes is apparent.

Interval No. 6 - At this point the real time clock, responsible for initiating the main conveyor's motion, is reset. The time setting of the clock is determined by worst case conveying time from the dispensers to the carts.

From here the system recycles back to Interval No. 1, and the next order is input, and the whole process is repeated.

Intervals No. 1-6 - An interrupt can occur at any time during the process. The first thing the interrupt service routine does is to determine which device is interrupting. This is done by checking the device flags of the possible sources of interrupt. If the real time clock is the source of the interrupt, the service routine initiates the main conveyor's motion by setting the run flip-flop shown in Figure 10. If the interrupt is other than the real time clock, the general alarm is sounded, and the device code number is output to the teletype. This number tells the operator which device interrupted.

After servicing a real time clock interrupt, control is automatically returned to the main program at the point where it was interrupted. After servicing any other interrupt, the system is halted and waits for the operator to return control to the main program.

One of the main criteria for the grocery store control system is a throughput of at least 120 orders per hour. If the following best case conditions are assumed,

1. No multiple entries
2. No out of stock items
3. Comparison time for one stop of main conveyor with system fully loaded = 12.5 seconds (based on SCC-650 simulation)
4. Items can be conveyed down aisles and into carts before the end of the comparison process (once dispensers are actuated, conveying down aisles and comparison occur independently and simultaneously)
5. High speed tape reader (300 lines/sec.) can input a 40 item order in approximately .25 seconds.

the resulting processing for 120 orders is approximately 25.5 minutes. The 120 comparison processes take 25 minutes and the 120 inputting processes take 30 seconds.

Worst case throughput analysis is omitted here since it must take into account the dispensing and conveying times and the maximum number of items that are out of stock at any one time. Both of these areas are outside the scope of this thesis.

B. Memory Organization

Figure 18 shows the basic memory organization, and an explanation of each section follows.

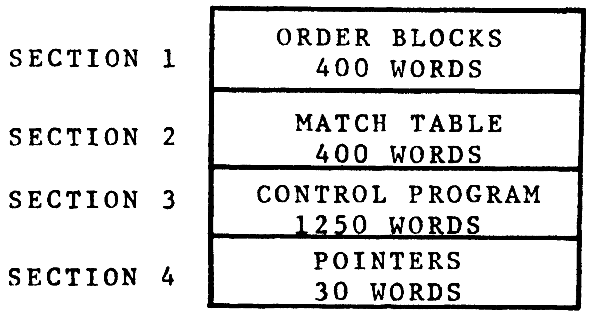


FIGURE 18 - Memory Organization

<u>Section 1</u> - Since each order has a maximum of 40 items, and there are 10 orders maximum being serviced at any one time, 400 words of memory must be reserved for storing 10 orders. This section of memory is hereafter referred to as the "order blocks".

<u>Section 2</u> - This section of memory contains the matches detected in the comparison routine. At any one stop of the main conveyor, it is possible that 400 matches can be entered in this section of memory - thus the requirement of 400 words. This memory section will hereafter be referred to as the "match table".

<u>Section 3</u> - This section of memory is space allowed for the control program, including the interrupt service routine. The number of words required is based on the SCC-650 simulation requirement of 2334_8 or about 1250 decimal. An explanation of the program appears in Section VIC.

<u>Section 4</u> - The 30 words allowed in this section contain such pointers as the starting locations for the 10 order blocks, the membership numbers corresponding to each

of the order blocks, and pointers that keep track of the next location in the match table to be output. Putting these 30 pointers in the first 30 locations of memory allows them to be used in a very efficient indirect addressing mode. For example, assume that a pointer is stored in M(0000), i.e., memory location $(0000)_8$. Assume also that it contains the address of the next location of memory to be output to the decoding unit (next item code to be output). A program employing this pointer which accomplishes the desired output follows:

```
Start   LDA*    00      Loads accumulator indirectly,
                        i.e., loads A with contents of
                        address specified by M(0000)

        TFA     '23     Transmits from accumulator to
        JMF     *+2     output device, when device
        JMB     *-2     ready to receive

        MIN     00      Adds a "1" to contents of
                        M(0000)

        JMP     START   Recycles and outputs next item
                        code
```

Of course, relative plus indirect addressing can also be used here instead of employing the first 30 locations of memory, but it requires more time.

The total memory required for the four sections of Figure 18 is therefore approximately 2.1K, which is acceptable since a standard sized core of 4K is readily available.

C. Program Description

A flow diagram of the main program appears in Appendix B. Each block will now be explained.

<u>Initialization</u> - This section of the program clears the appropriate order block into which the next order is to be input from paper tape. This is necessary because of the linear comparison approach wherein all 40 words of an order block are compared regardless if the order is 40 items long or not. This section also clears (initializes) the match table.

<u>Input</u> - As stated earlier, the data on paper tape has each 16 bit word split in half, i.e., in two rows of eight bits. Therefore, this input program is a loader which essentially inputs the data in two parts and then reassembles the complete word in the computer. The basic program is as follows:

```
CLA              Clears Accumulator

TTA     '03      Transmits most significant portion of
JMF     *+2      data word into least significant por-
JMB     *-2      tion of accumulator

XHA              Exchanges Halves of the Accumulator

CAX              Stores contents of accumulator in "X"
                 register

CLA

TTA     '03      Transmits least significant portion of
JMF     *+2      data word to least significant portion
JMB     *-2      of the accumulator

AOX              Performs logical OR between contents of
                 accumulator with "X" register, thus
                 reassembling the whole data word
```

The input section of the program is also responsible for keeping track of where to store the incoming order, and also must detect when the input is complete by checking for an all zero row.

Comparison - The comparison routine is the heart of the control system software. A general flow diagram is shown in Figure 19. There are three loops in the comparison program. The innermost loop compares each of the 40 items in the order at aisle no. 1 with the store item no. 1. Loop 2 controls the incrementing to store item no. 2 and then invokes Loop 1 again. Loop 3 increments from the order at aisle no. 1 to the order at aisle no. 2, and then invokes the inner two loops.

Assuming the following pointer designations,

M(PTR1) -	contains location of next item from a particular order to be compared
M(PTR2) -	contains the code for the next item in a particular aisle to be compared
M(PTR3) -	contains location where next match is to be stored in match table

the basic comparison and storage-if-match process can be illustrated as follows:

```
LDX*      PTR1      Loads "X" register with item from some
                    order

LDA       PTR2      Loads "A" register with next item in
                    aisle to be compared

SRA       3,Z       Performs logical exclusive OR between
                    "A" and "X", and skips next instruc-
                    tion if result is 0.

JMP       *+2       Jumps past storage step if not a match

STX*      PTR3      Store match in match table, indirectly
                    specified by M(PTR3)
```

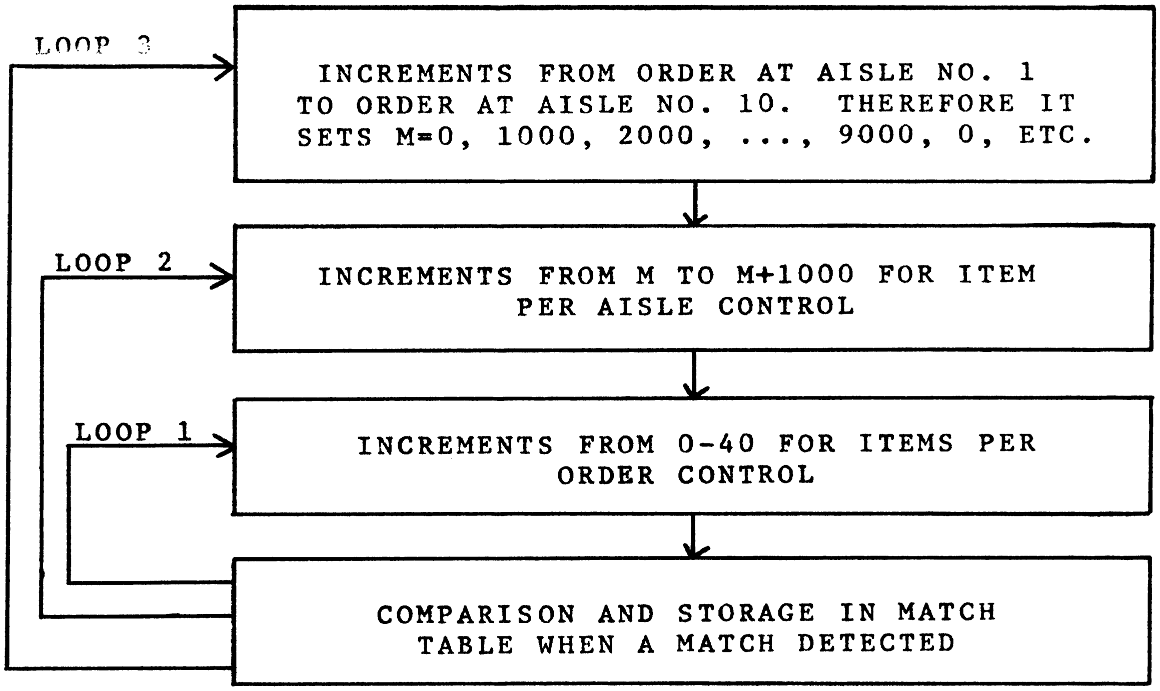

LOOP 3
INCREMENTS FROM ORDER AT AISLE NO. 1 TO ORDER AT AISLE NO. 10. THEREFORE IT SETS M=0, 1000, 2000, ..., 9000, 0, ETC.
LOOP 2
INCREMENTS FROM M TO M+1000 FOR ITEM PER AISLE CONTROL
LOOP 1
INCREMENTS FROM 0-40 FOR ITEMS PER ORDER CONTROL
COMPARISON AND STORAGE IN MATCH TABLE WHEN A MATCH DETECTED

FIGURE 19 - Comparison Routine Flow Diagram

<u>WAIT</u> - The next two blocks of the flow diagram shown in Appendix B are the synchronization schemes that occur during Interval No. 3 described in Section VIA.

The first wait loop circulates until the real time clock interrupt occurs. It then checks a preassigned memory location (BUF) for the presence of a "1" or a"0", indicating that the interrupt has or has not occurred, respectively. When the interrupt occurs, the interrupt service routine sets M(BUF) to a "1". Anytime after this, when the wait loop checks and finds M(BUF) = 1, it clears M(BUF) and goes on. The basic program to accomplish this checking scheme is as follows:

```
LDX    BUF    Load "X" register with contents of
```

```
                M(BUF)

LDL     1       Load "A" register with a "1"

SRA     3,Z     Exclusive OR performed between "X" and
                "A"; if result is 0, the next instruc-
                tion is skipped

JMP     *-3     Recycles if M(BUF) ≠ 1
```

The other wait loop in the flow diagram is similar to the above, only the device flag flip-flop of the main conveyor (see Figure 10) is checked instead of M(BUF).

STAMP - This block of the program simply takes the membership number corresponding to the cart located at aisle no. 1 and outputs it to the stamping unit.

OUTPUT TO DECODER - This block is actually made up of 3 subroutines: the output to the decoder, the output to paper tape punch when an out of stock condition is detected, and the detection/wait loop routine to handle multiple entries of the same item by an order.

The first subroutine simply takes the appropriate entries in the match table and transmits the whole 16 bit word at once to the decoding unit. Before each attempt to output, however, a flag test is made. If a dispenser's flag is raised, an out of stock condition is present. If this happens, the computer outputs the membership number and the item code number to the paper tape punch. This outputting is again in half words at a time, and is therefore very similar to the input program for paper tape shown earlier.

The third subroutine, which causes a wait loop to be entered if a multiple entry is detected, compares the last item code output to the next item code to be output, by means of a logical exclusive OR operation. If they are the same, the loop repeatedly checks the device flag (flag set indicates dispenser in motion), and exits the loop allowing normal output only after the dispenser returns to a suitable position.

RESET - This section of the flow diagram causes a pulse to reset the real time clock, once the output to the decoding unit is complete.

INTERRUPT SERVICE ROUTINE - The general flow diagram for the interrupt service routine appears in Appendix C. The first function of the interrupt service routine is to find out which device is interrupting. A scheme for doing this is shown below:

```
LDA      01    Gets device number into accumulator

SDF      01    Tests device flag; skips next instruc-
               tion if flag raised

JMF     *+3    Jump to check if next device has
               interrupted

STA    TEMP    If this is interrupting device, pre-
               serve its device code number

JMP    SEU1    Jump to service routine number 1

LDA      02
SDF      02    Same as above only checks device num-
JMF     *+3    ber 2
STA    TEMP
JMP    SEU2
```

All the trouble sources, except the real time clock, cause a jump to the same service routine. The real time clock necessarily has a separate service routine.

The service routine for the real time clock first stores a "1" in the memory location discussed earlier called BUF, for the purpose of synchronization. Secondly, the service routine causes the run flip-flop in the main conveyor control unit to be set "on" (see Figure 10). As mentioned in Section VIA, the main program is then reentered at the point where it was left when the interrupt occurred.

The scheme for outputting a coded word to the teletype in case of an interrupt other than the real time clock, makes use of the memory location called TEMP in the program just illustrated. The contents of this location must be output to the teletype in the same piecewise fashion that the paper tape data is handled. This is so because the teletype requires a 7 bit binary number to cause it to print a single octal digit, and each device code stored in M(TEMP) is two octal digits long (to allow for more than 7 interrupt sources).

A duplication of the printout obtained from the SCC-650 simulation appears in Appendix D. Comments are included to indicate the block of the flow diagram (Appendices B and C) corresponding to the various lines of the printout. Table 3 lists the sample orders used in the simulation (octal numbers), where the first entry of each order is the customer's membership number. Table 4 lists the range of octal codes of the items contained in each of the 10 aisles. Also, in the simulation, the SCC-650's external switch register is used to indicate interrupt numbers and derive flag status, since the actual peripheral control units are not available.

Order No. 1	Order No. 2	Order No. 3
0001	0002	0003
0201	0202	0203
0761	0762	0203
1201	1202	1203
1701	1702	1703
2601	2602	2603

TABLE 3 - Sample Orders for SCC-650 Simulation

Aisle	Octal Range of Item Codes
1	140-760
2	761-1600
3	1601-2420
4	2421-3240
5	3241-4060
6	4061-4700
7	4701-5520
8	5521-6340
9	6341-7160
10	7161-7777

TABLE 4 - Octal Code Ranges for Items in Various Aisles for SCC-650 Simulation

VII. CONCLUSION AND RECOMMENDATIONS FOR FUTURE STUDY

The proposed system employing a 16 bit minicomputer and the various peripherals previously described, forms a workable grocery store order filling system. The minimum desired throughput of 120 orders/hour is attainable with the system, based on the SCC-650 simulation. All of the computer devices (or elements of the devices) described are presently available, making the control system realizable.

Since this thesis is mainly concerned with the minicomputer control phase of the grocery store system, a major shortcoming is not covered. The shortcoming is the design and development costs of the system. It would be necessary that a large organization, such as a chain store, take on this cost to make it at all feasible. A very strong factor that could be used in an economic study of this cost is that a throughput of more than 120 orders/hour is attainable, under favorable conditions. An economic study is, however, a topic for further research.

There are several other areas that can be examined in further detail. For example, the system that receives orders from customers over the telephone lines is complex enough to require a minicomputer control system of its own. Another area for further investigation is the design and development of the hardware to implement the decoding network which causes the appropriate product dispensers to actuate. Still another area for future research is the application of disc storage units and cassette magnetic tape storage units especially made for minicomputer interfacing. These units can allow for the inventory control and order processing to take place simultaneously.

The age when all material handling in commercial stores is done under computer control is not likely to be far off. With the advances in technology, fabrication of minicomputers is becoming more and more inexpensive. As a result, their application to control systems such as the grocery store system will undoubtedly multiply.

BIBLIOGRAPHY

1. Perlman, J. A., (May 1970) "Materials Handling: New Market for Computer Control", Datamation, Vol. 16, No. 5, pp. 133-137.

2. Hipwell, J. C., (March 1969) "Inventory Management on a Desk-Sized Computer", Data Processing, Vol. 11, No. 2, pp. 118-122.

3. Mueller, D. M., (January 1970) "Applying Computers to Warehousing", Automation, Vol. 17, No. 1, p. 47.

4. Theis, D. J., Hobbs, L. C., (March 1969) "Minicomputers for Real-Time Applications", Datamation, Vol. 15, No. 3, pp. 48-53.

5. Woodworth, F. M., (1967) "Time Sharing", Personal Correspondence.

6. McDaniel, J. R., (June 1970) "ABC's of Time-Sharing", Automation, Vol. 17, No. 6, p. 46.

7. Yourdon, E., (October 1969) "Time Sharing for Small Businesses", Computers and Automation, Vol. 17, No. 6, p. 46.

8. Lapidus, G., (September 1968) "Minicomputers - What All the Noise is About", Control Engineering, Vol. 15, No. 9, pp. 73-80.

9. Bigelow, R. P., (August 1969) "Some Legal Aspects of Commercial Remote Access Computer Services", Datamation, Vol. 15, No. 8, pp. 48-52.

10. Lapidus, G., (November 1969) "A Look at Mini-Computer Applications", Control Engineering, Vol. 16, No. 11, pp. 85-86.

11. Kluckman, A. Z., (December 1969) "Minicomputers on the Move", Computers and Automation, Vol. 18, No. 13, pp. 25-26.

12. Jurgen, R. K., (August 1970) "Minicomputer Applications in the Seventies", IEEE Spectrum, Vol. 7, No. 8, pp. 50-51.

13. Flores, I., (1969) Computer Organization, New Jersey, Prentice-Hall, pp. 112-132.

14. Kintner, P. M., (November 1969) "Interfacing A Control Computer with Control Devices", Control Engineering, Vol. 16, No. 11, pp. 97-101.

15. Pottinger, H. J., Tracey, J. H., (January 1968) Formal Description of the SCC-650 Computer, Technical Report CRL 68.1, pp. 12-16.

VITA

Neal Joseph Martini was born on October 9, 1946 in Buffalo, New York. He received his Bachelor of Science degree in Electrical Engineering from the University of Detroit in June 1969. He enrolled in graduate school at the University of Missouri-Rolla (St. Louis Extension - evening division) in August 1969, while working at Monsanto Company in St. Louis. He came to the University of Missouri-Rolla campus in August 1970 to complete the requirements for a Master of Science degree in Electrical Engineering.

Appendix A

CONDENSED LIST OF INSTRUCTIONS FOR SCC-650

NOTE: *Indicates indirect addressing; *+ or - indicates relative addressing

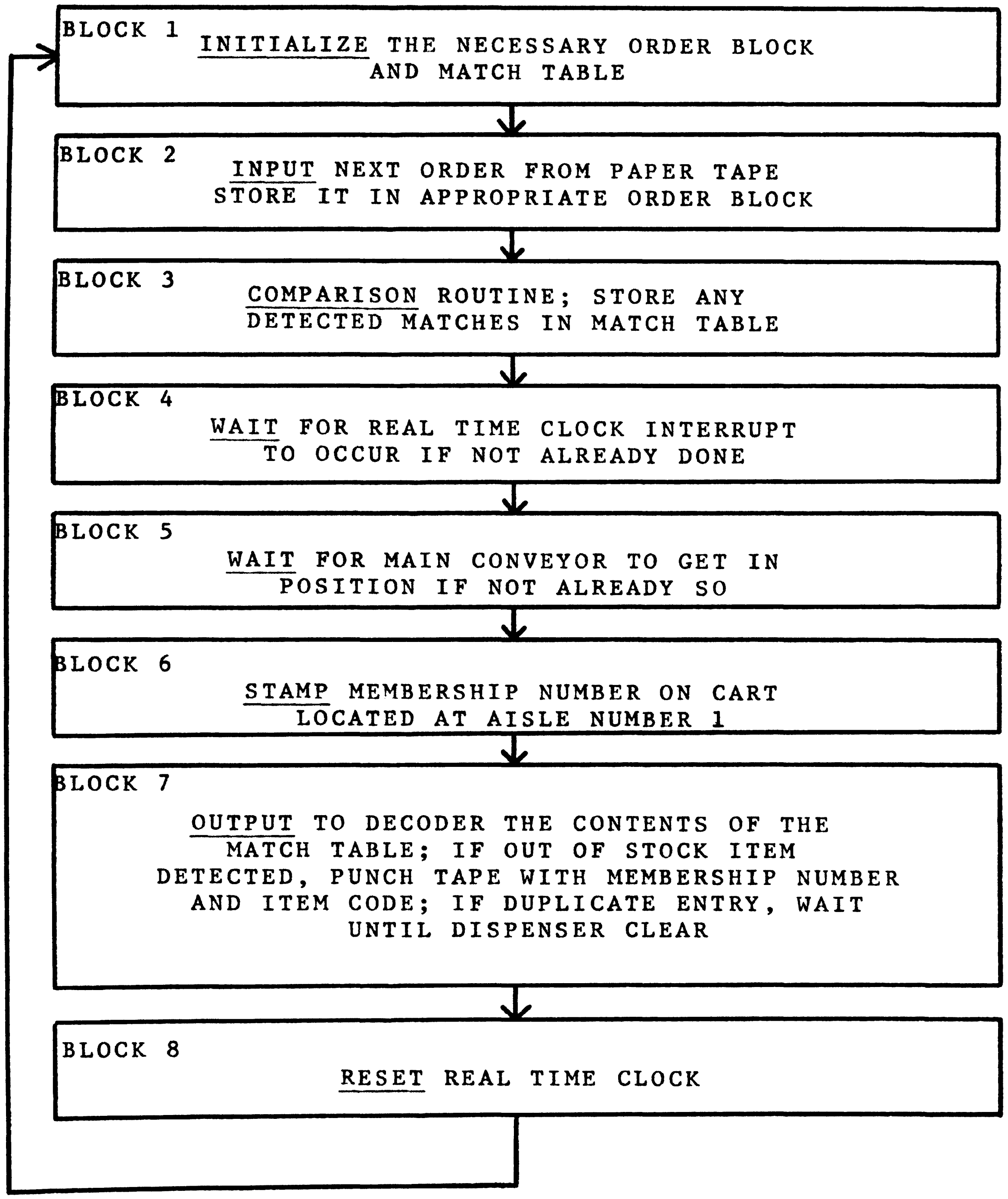
MNEMONIC	NAME
LOAD/STORE	
LDA	Load accumulator
STA	Store accumulator
LDX	Load index
LDL	Load accumulator, literal
STX	Store index
LDS	Load status register
LAS	Load accumulator from switches
STS	Store status register
LDI	Load IB from accumulator
LDM	Load IB from memory
ARITHMETIC	
ADD	Add
SUB	Subtract
MIN	Memory increment and skip on zero
ADL	Add, literal
ADC	Add carry
LOGICAL	
XOR	Exclusive or
AND	Logical and
ANL	And, literal
XOL	Exclusive or, literal
AOX	Or accumulator with index
AAX	And accumulator with index
JUMP	
JMF	Jump forward
JMB	Jump backward
JSL	Jump and store location
JRT	Return jump

APPENDIX A (continued)

MNEMONIC	NAME
REGISTER	
CLA	Clear AC
CLX	Clear index
CAX	Copy accumulator into index
CXA	Copy index into accumulator
XAX	Exchange accumulator and index
XHA	Exchange halves of accumulator
CONTROL	
HLT	Halt
NOP	No operation
CLI	Clear interrupt
SCN	Set carry ON
SCF	Set carry OFF
ENA	Enable interrupts
DIS	Disable interrupts
XSS	Index state set
XSR	Index state reset
MICROINSTRUCTIONS	
SRA	Select register A
SRAC	Select register A and clear
SRX	Select register X
SRXC	Select register X and clear
INPUT/OUTPUT	
SEL	Select
IOC	Input/Output control
TTA	Transmit to accumulator
TFA	Transmit from accumulator
DST	Input device status
SDF	Skip on device flag
EXU	Execute command in accumulator
TMR	Terminate

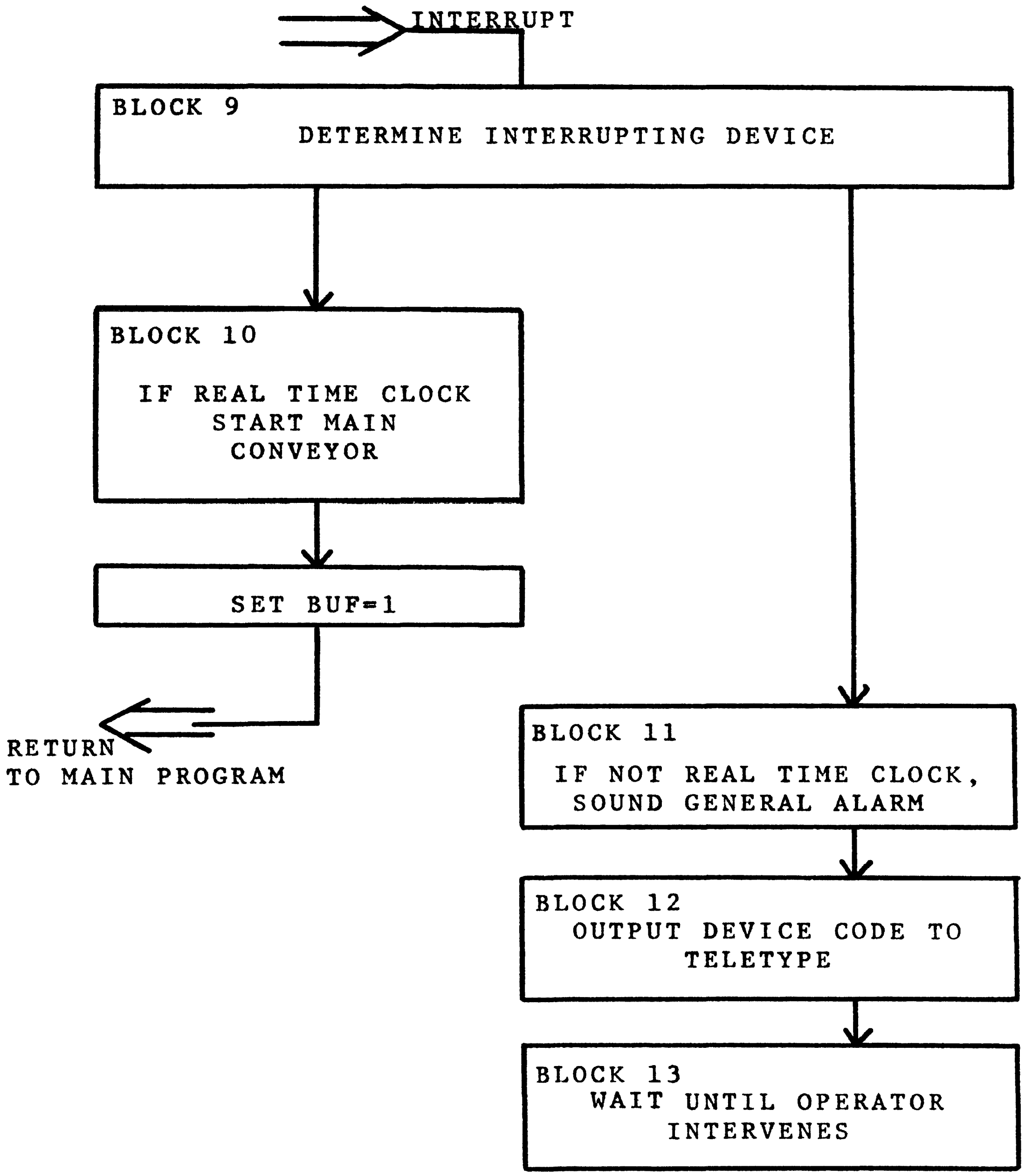
Appendix B

MAIN CONTROL PROGRAM FLOW DIAGRAM


BLOCK 1
INITIALIZE THE NECESSARY ORDER BLOCK AND MATCH TABLE
BLOCK 2
INPUT NEXT ORDER FROM PAPER TAPE STORE IT IN APPROPRIATE ORDER BLOCK
BLOCK 3
COMPARISON ROUTINE; STORE ANY DETECTED MATCHES IN MATCH TABLE
BLOCK 4
WAIT FOR REAL TIME CLOCK INTERRUPT TO OCCUR IF NOT ALREADY DONE
BLOCK 5
WAIT FOR MAIN CONVEYOR TO GET IN POSITION IF NOT ALREADY SO
BLOCK 6
STAMP MEMBERSHIP NUMBER ON CART LOCATED AT AISLE NUMBER 1
BLOCK 7
OUTPUT TO DECODER THE CONTENTS OF THE MATCH TABLE; IF OUT OF STOCK ITEM DETECTED, PUNCH TAPE WITH MEMBERSHIP NUMBER AND ITEM CODE; IF DUPLICATE ENTRY, WAIT UNTIL DISPENSER CLEAR
BLOCK 8
RESET REAL TIME CLOCK

Appendix C

INTERRUPT SERVICE FLOW DIAGRAM



Appendix D

SIMULATION PRINTOUT

Printout	Comments (See Appendices A&B)
Conveyor in Position	Block 5
The Membership # is 0001	Block 6
0201	Block 7
Reset Real Time Clock	Block 8, 1, 2, 3
Interrupt # 01	Block 9
Start Main Conveyor	Block 10
Conveyor in Position	Block 5
The Membership # is 0002	Block 6
0202	Block 7
0761	
1201	
Reset Real Time Clock	Block 8, 1, 2, 3
Interrupt # 76	Block 9
Alarm	Block 11, 13
Interrupt No. 01	Block 9
Start Main Conveyor	Block 10
Conveyor in Position	Block 5
The Membership # is 0003	Block 6
0203	Block 7
Duplicate Entry, Wait Until Clear	
0203	

Appendix D (continued)

	Comments (See Appendices A&B)
0762	
1202	
1701	
Reset Real Time Clock	Block 8, 1, 2, 3
Interrupt # 01	Block 9
Start Main Conveyor	Block 10
Conveyor in Position	Block 5
'	
'	
'	
ETC.	