Order Number 1333248

Real-time process control and simulation for chemical mix facility

Liu, Pi-Shien, M.S.

The University of Arizona, 1988

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __✓__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages __✓__
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Dissertation contains pages with print at a slant, filmed as received _____
16. Other _____

_____

_____

REAL-TIME PROCESS CONTROL AND SIMULATION FOR CHEMICAL MIX FACILITY

By

Pi-Shien Liu

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN ELECTRICAL ENGINEERING

In The Graduate College

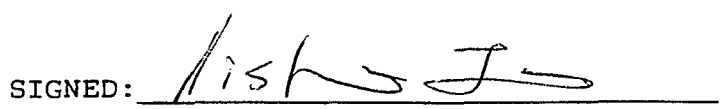
THE UNIVERSITY OF ARIZONA

1 9 8 8

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotation from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction or this manuscript in whole or in part may be granted by the head of the major department of the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: 

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Theodore L Williams
Theodore L Williams
Professor
Electrical and Computer Engineering

April 30, 1988
DATE

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to his advisor and committee chairman, Professors Theodore L Williams, for his guidance, helpful comments and support and professor F. J. Hill and Dr. Sy-Yen Kuo, for their suggestions and evaluation during the course of this study. I would also like to extend my appreciation to all my family members for their constant support, and to my friends for their constructive comments during the preparation of this thesis.

TABLE OF CONTENTS

TABLE OF CONTENTS--Continued

LIST OF TABLES

LIST OF ILLUSTRATIONS

ABSTRACT

The purpose of this study is to design a real-time control and simulation system for a chemical mix facility. A simulation circuit board and software simulation in an IBM personal computer emulated the real-time chemical mix facility. A second personal computer controlled the plant. The parallel port in the IBM PC computer serves as a communication path between the controlled and controlling system. Results show that the simulation can assist the design of the actual system.

CHAPTER 1

INTRODUCTION

Real-Time Systems

Over the past few years the dramatic decrease in the cost of computing hardware has led to a proliferation of real-time systems in a wide range of applications. Most of these systems are now being introduced to control commercial, industrial and communications systems.

A real-time system [1, 2] has stringent time constraints. Processing must be done within these defined constraints or the system will fail. Contrast this requirement to a time-sharing system where it is desirable (but not mandatory) to respond quickly, or to a batch system where there may be no time constraints at all.

"Real-time" is used to describe two very different kinds of systems. The older usage is applied to such things as airline reservation systems and message-switching centers. These systems were considered "real-time" because there was an operator sitting at a terminal waiting for a response. These systems are now more commonly called interactive. Another older term for them was on-line.

Another real-time system is the "process control" system [3]. Before the advent of the mini-computer and the microprocessor, this was a less commonly available; however, the situation is now reversed. With the "process-control" system, the computer directs or monitors an ongoing physical process. In this report, the real-time system is applied to a process-control environment.

Most microcomputer real-time applications fall into the loose category of "process control." This simply means that the computer is controlling an ongoing process in the outside world. In most cases, the computer is "embedded" in another product, and does not appear to the user as a general-purpose computer or, indeed, as a computer at all. The computer is simply performing a function that a few years ago might have been performed by a collection of levers and switches driven by a timing motor.

A real-time system reacts so as to affect the environment in which it is operating. It is a collection of devices, controlled by a stored program of instructions. This program acts as the regulating element in a feedback loop, which then forms part of a industrial system. For convenience, the entire real-time system can divide into two parts - the controlled system and the controlling system. The controlled system consists of the hardware devices and related elements which make up the process

plant. The controlling system consists of the software element together with its associated processing hardware which does the control job.

The fact that a real-time software is working as a controlling element in a time-critical environment becomes very evident when viewing these applications. Another aspect that emerges when considering this area of application is that the software must be reliable. Large sums of money, valuable equipment and human lives often depend on the correct and reliable operation of the software controlling these systems.

Use of Simulation

As mentioned earlier, the time-critical nature of the real-time process adds new opportunities for errors. If the time constraints are violated, then unwelcome effects may occur. Therefore verification aids, such as simulation, are necessary for getting the design right before modifications are expensive and time consuming.

Simulating a system [1] means: developing a model of the system and then seeing how it responds to inputs; refining the model, then drawing conclusions to assist the design of the actual system. The simulation technique can be used at several levels in the development of a real-time system.

(1) To simulate the complete operational policy to check and improve the design.

(2) To simulate the chosen computer equipment on another machine so that programs can be developed before the actual computer chosen is available. This is called emulation.

(3) To simulate the environment for the computer system in order to test the new system, once developed.

(4) To vary plant parameters to test the robustness and reliability of the control; thereby avoiding operation at the edge of disaster.

When considering simulation, the costs in manpower, elapsed time, and computer time need to be estimated in advance. These costs must be weighed against the risk of proceeding directly with the real implementation of the design.

Thesis Goal

The goal of this thesis is to control and simulate a real-time bottle filling facility. The system consists of the two parts - the controlled system which uses a computer to emulate a real-time bottle filling environment, and the controlling system which using a second computer to control the bottle filling facility. A simulation circuit board has been built for the purpose of controlling and simulating.

Figure 1.1 shows the system diagram of the bottle filling simulation. The software, which is written in C language, has been implemented to control and simulate the system.

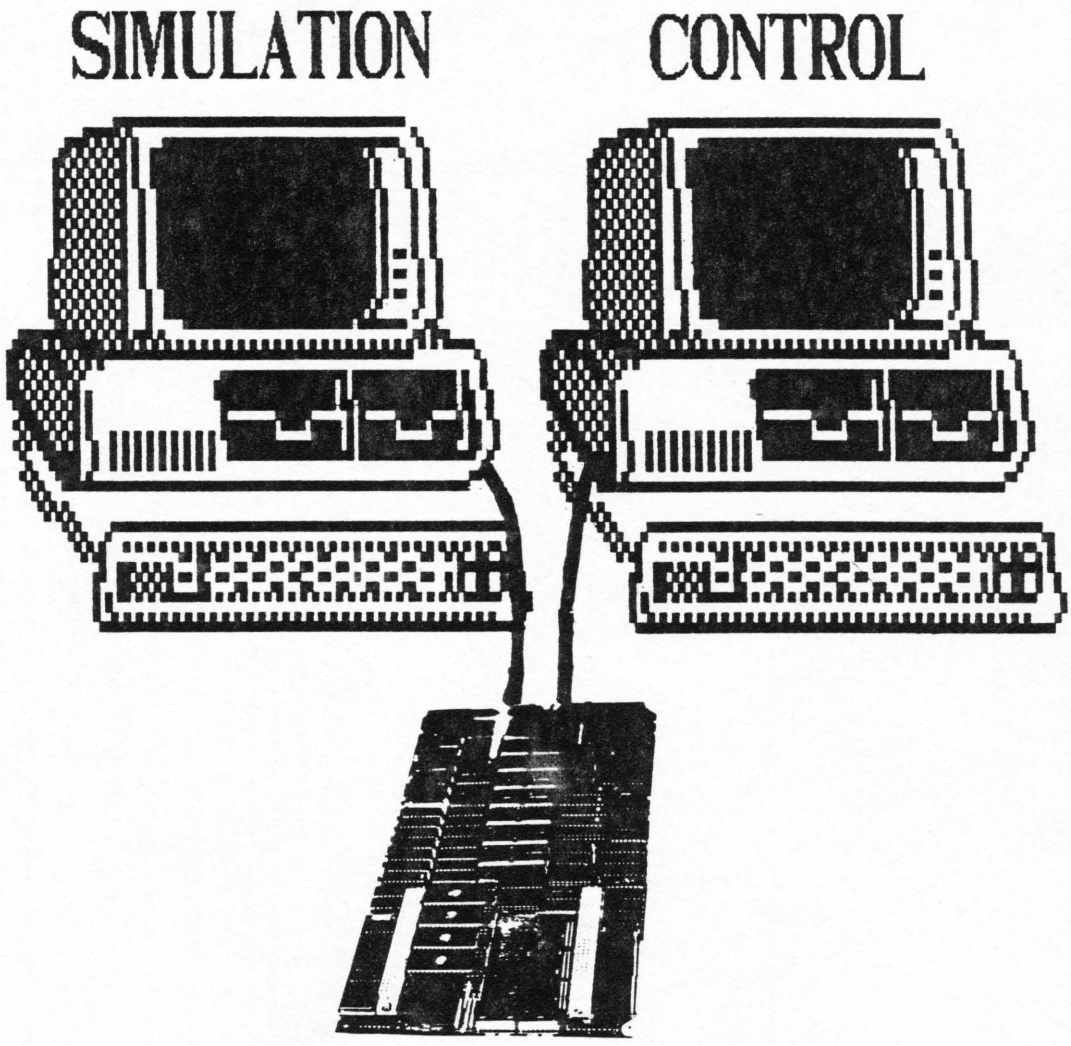


Figure 1.1 Simulation System Diagram

CHAPTER 2

DESCRIPTION OF BOTTLE FILLING FACILITY

The automated bottle filling facility has been designed for the mixing and bottling of several chemical ingredients.

Up to six chemical ingredients flow from bulk tanks to batch tanks to hold tanks (Figure 2.1). The chemical ingredients are mixed in the hold tanks, and then the operator is asked for a quality assurance code. Finally, the product is bottled by the bottle filler.

A similar facility has been installed in Phoenix, Arizona for Motorola. In this project, the control and simulation program for the bottle filling facility has been designed to assist and improve the real-time process control environment.

Elements in Bottle Filling Facility

Bulk Tanks

There are six bulk tanks, each with a storage capacity of 5000 gallons, which stores the various chemical ingredients. Each bulk tank has a fill valve and a dump valve. The fill valve opens when the truck tank is filling

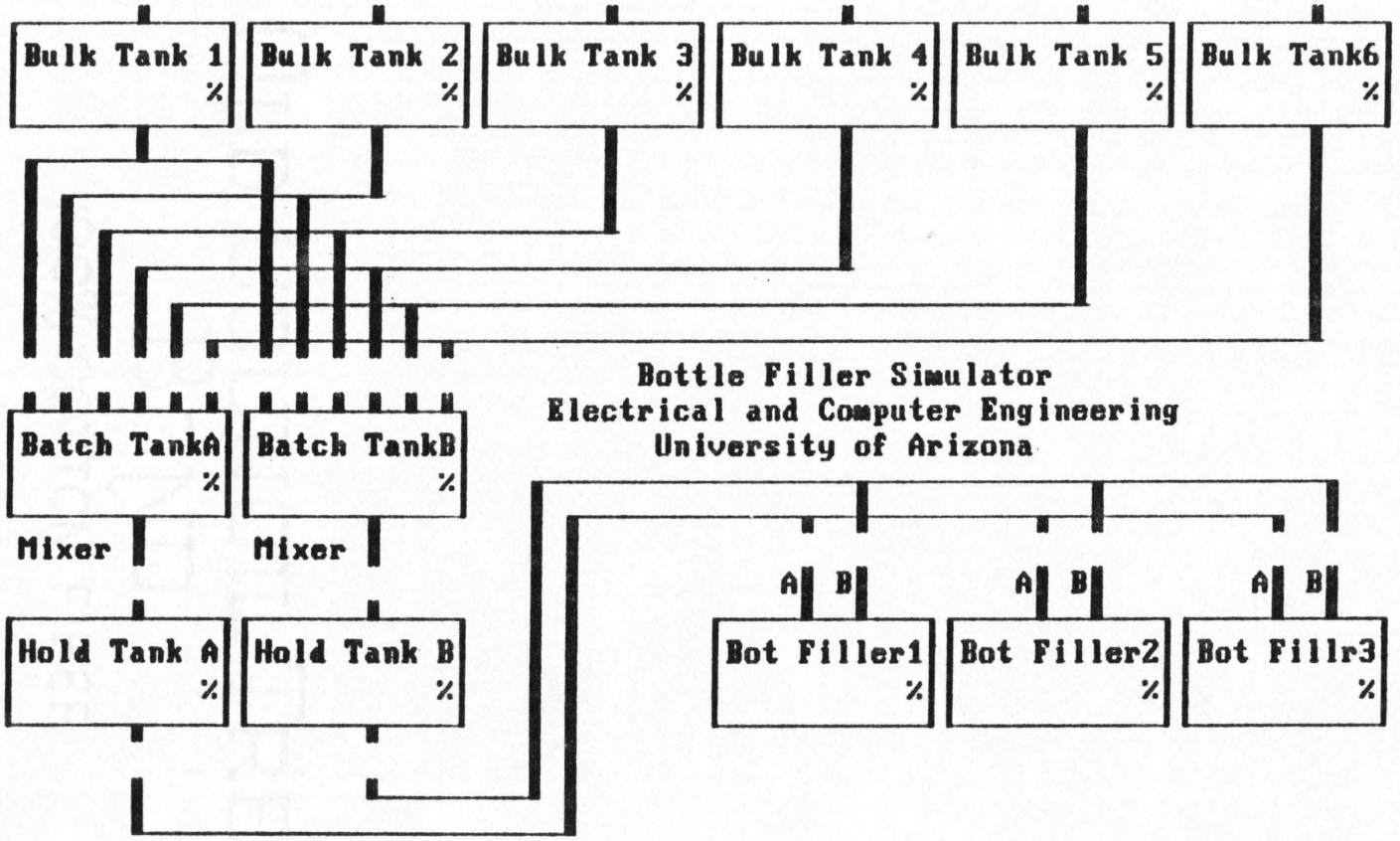


Figure 2.1 Display Screen of Bottle Filler Simulator

the bulk tank, and the dump valve opens when the bulk tank is dumping its contents into a batch tank.

Batch Tanks

There are two batch tanks in the bottling filling facility (Figure 2.1). Each batch tank has the capacity of 100 gallons. Each batch tank has six fill valves and one dump valve. The six batch tank fill valves are connected to six bulk tank dump valves. The batch tank fill valves open when the bulk tank is dumping its chemical contents. When the batch tank reaches its specified level, the fill valve closes and the dump valve opens, dumping the chemical contents to the hold tank.

Hold Tanks

Two hold tanks are used to collect various of chemical ingredients from the batch tank. The capacity of hold tank is 500 gallons. Each has one fill valve and one dump valve. The fill valve opens when the batch tank dumps its chemical contents. The dump valve opens when the bottle filler start bottling the final product.

Bottle Fillers

Filling the bottles is the final stage in the bottle filling facility. Three bottle fillers are employed. Each has two fill valves, one for each hold tank.

Mix Motors

Each hold tank has a mix motor which stirs the chemical contents in the hold tank to assure a uniform mixture.

Pipes

The valve systems are connected by piping through which the chemical contents flow from one tank to the next.

Definitions

Recipe

A specified proportion of each chemical ingredient is needed to produce a final chemical product.

Menu

The menu consists of recipes for the chemical products which the system is capable of producing through various combinations of the six bulk tank chemicals. The operator selects a recipe from the menu to start a "loop" (loop is defined next).

Loop

Two loops are available in the simulation: loop A and loop B. Each loop consists of a batch tank, hold tank, and user selected bottle filler.

Batch

All of the ingredients in a recipe is called a batch, each batch flows from bulk tank to the batch tank, and collects in the hold tank.

Quality Assurance Code

After the motor turns on for a while duration, the quality assurance code must be entered. To assure the purity and correct composition of a batch, each batch is analyzed chemically. The inspector enters a quality assurance code to certify that the test is complete before the hold tank content is transferred to the bottle filler.

Limitations and Precautions

(1) No tank can be filled over its maximum capacity.

(2) Bulk tanks cannot be filled unless the bulk tank dump valve is closed.

(3) Only one bulk tank can be filled at one time.

(4) A batch tank cannot be filled unless its dump valve is closed.

(5) A hold tank cannot be filled unless its dump valve is closed.

(6) No new ingredients can be added if batch tank is not empty.

(7) No batch can be started if there are not enough

ingredients available.

(8) No bottle filler can be used by another loop system if it is already in use.

(9) Each ingredient in the recipe cannot exceed the hold tank capacity.

(10) Total sum of ingredients in a recipe cannot exceed the hold tank capacity.

(11) Mix motor should be on for one minute before inspection can start.

(12) No bottling can take place before the quality assurance code is entered.

Operator Input

The operator can enter the following commands:

(1) Enter the bulk tank number and ingredient amount to fill a bulk tank.

(2) Enter a recipe, or change the ingredient amount in an existing recipe.

(3) Select a recipe for bottling.

(4) Select a bottle filler for a loop.

(5) Inspect a batch and enter the Quality Assurance code.

Operator Display

The display provides the following information:

(1) All of the tank capacities in gallons.

(2) All of the current tank levels in gallons and as a percentage.

(3) All of the valve settings, and pump conditions.

(4) The mix motor conditions.

(5) The Quality Assurance code if entered.

(6) The name of the recipe if a loop is running.

(7) The total of bottled products.

CHAPTER 3

SIMULATION OF BOTTLE FILLING ENVIRONMENT

The bottle filling environment is simulated by using a simulation circuit board and software simulation in a IBM PC computer. The controlling system consists of another IBM PC computer and the software which does the control. A hardware interface is used to connect the controlled system to the controlling system.

Hardware Interface

In this design, the parallel port in the IBM PC computer serves as a communication path between the controlling and controlled system.

IBM Printer Adapter

The IBM printer adapter [4] is specifically designed to attach printers with a parallel port interface, but it can also be used as general input/output for other applications that match its input/output capabilities.

The input/output signals are made available at the back of the adapter through a right-angle, printed-circuit-board-mounted, 25-pin, D-shell connector. This connector protrudes through the rear panel of the system unit, where

a cable may be attached.

The IBM printer adapter provides 8-bit data output, 5-bit status input, and 4-bit control outputs using which are using as the communication path between the simulation computer and control computer.

Bottle Filler Simulation Circuits

The bottle filler simulation circuit, as shown in Figure 3.1, is designed to fulfil the simulation.

The simulation circuit board is connected between the controlling computer and the simulation computer. Figure 3.2 shows the pin assignments on the mass termination connector.

There are three control bytes sent from the control computer to the simulation system. The assignment of the control bits is shown in Table 3.1, 3.2, 3.3 and 3.4.

One byte controls the bulk tank filling valves and selects the tank levels returned by the simulator to the controller. The second byte is used to control loop A activity which includes the batch tank fill valves, dump valves, the mix motor in the hold tank, the dump valve in the hold tank, and the bottle filler valves for the loop. The third byte controls loop B.

The control system writes the three control bytes

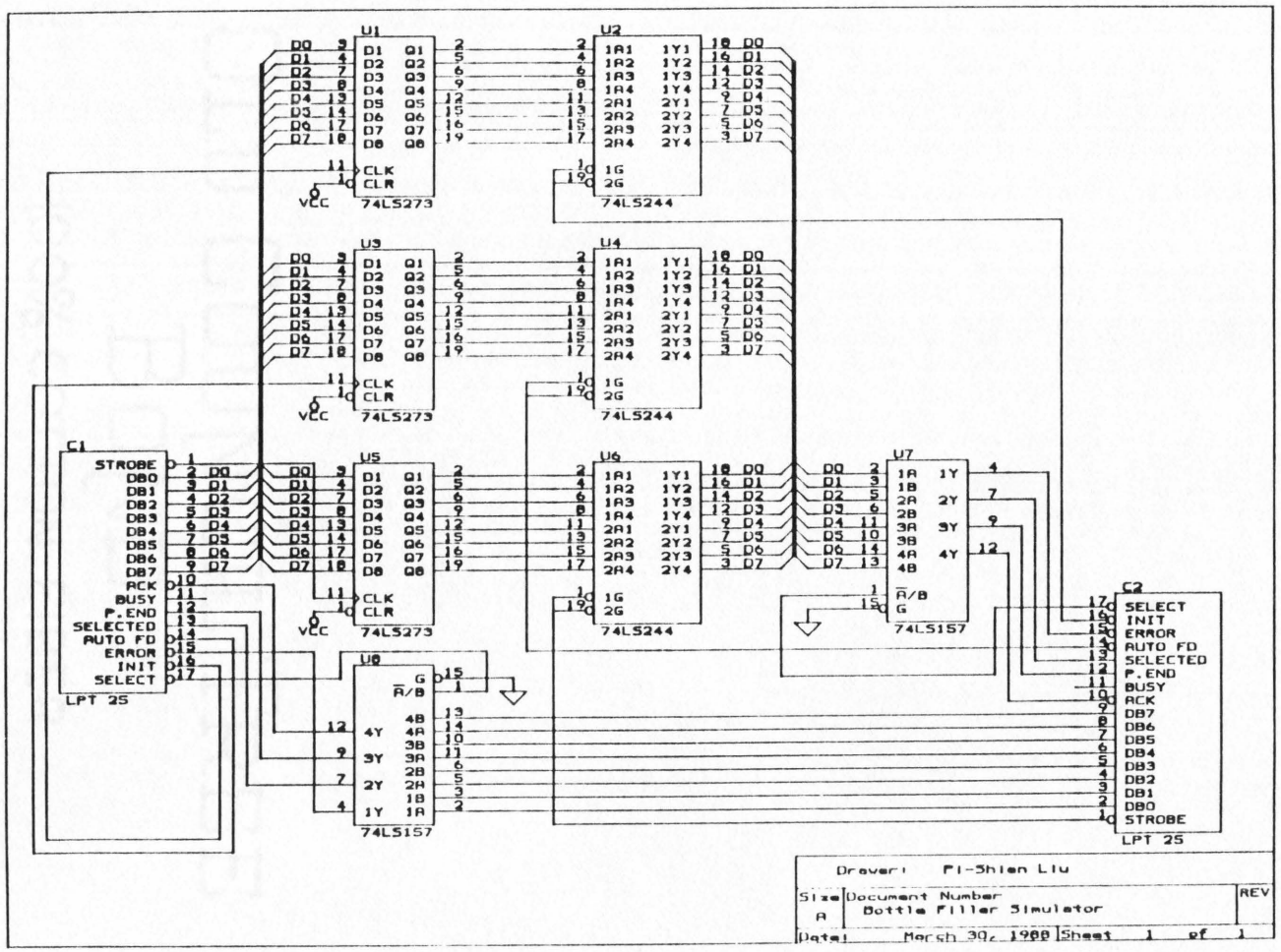


Figure 3.1 Bottle Filler Simulation Circuit Layout

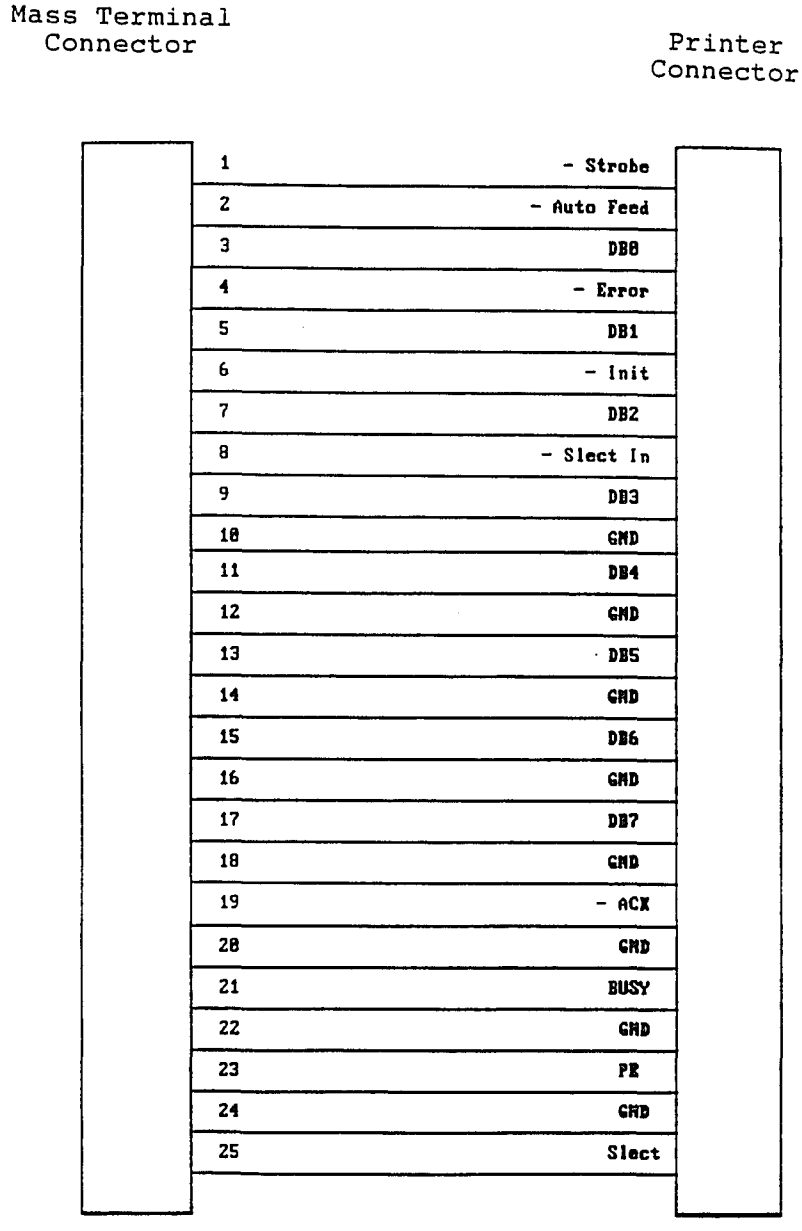


Figure 3.2 Pin Assignments on Mass Termination Connector

Table 3.1 Bit Assignments for Controlling Bulk Tank Filling

Control Bits (Hex)	Bulk Tank Number
1x xx xx	1
2x xx xx	2
3x xx xx	3
4x xx xx	4
5x xx xx	5
6x xx xx	6

Table 3.2 Bit Assignments for Reading Tank Levels

Control Bits (Hex)	Batch Tank Status
xx xx ex	Batch Tank A Dumping Ing.
xx xx cx	Batch Tank A Fill Ing. 6
xx xx ax	Batch Tank A Fill Ing. 5
xx xx 8x	Batch Tank A Fill Ing. 4
xx xx 6x	Batch Tank A Fill Ing. 3
xx xx 4x	Batch Tank A Fill Ing. 2
xx xx 2x	Batch Tank A Fill Ing. 1
xx ex xx	Batch Tank B Dumping Ing.
xx cx xx	Batch Tank B Fill Ing. 6
xx ax xx	Batch Tank B Fill Ing. 5
xx 8x xx	Batch Tank B Fill Ing. 4
xx 6x xx	Batch Tank B Fill Ing. 3
xx 4x xx	Batch Tank B Fill Ing. 2
xx 2x xx	Batch Tank B Fill Ing. 1

Table 3.3 Bit Assignments for Reading Tank Levels

Control Bits (Hex)	Tanks
xc xx xx	Bulk Tank Ingredient 6
xb xx xx	Bulk Tank Ingredient 5
xa xx xx	Bulk Tank Ingredient 4
x9 xx xx	Bulk Tank Ingredient 3
x8 xx xx	Bulk Tank Ingredient 2
x7 xx xx	Bulk Tank Ingredient 1
x6 xx xx	Batch Tank B
x5 xx xx	Hold Tank B
x4 xx xx	Batch Tank A
x3 xx xx	Hold Tank A
x2 xx xx	Bottle Filler 3
x1 xx xx	Bottle Filler 2
x0 xx xx	Bottle Filler 1

Table 3.4 Bit Assignments for Controlling Mix Motors and Selecting Bottle Fillers

	Description	Control Bits (Hex)
MMA	Loop A Mix Motor Control	xx xx 1x
MMB	Loop B Mix Motor Control	xx 1x xx
BTLA3	Loop A Select Bottle Filler 3	xx xx x4
BTLA2	Loop A Select Bottle Filler 2	xx xx x2
BTLA1	Loop A Select Bottle Filler 1	xx xx x1
BTLB3	Loop B Select Bottle Filler 3	xx x4 xx
BTLB2	Loop B Select Bottle Filler 2	xx x2 xx
BTLB1	Loop B Select Bottle Filler 1	xx x1 xx

on the simulator board via the data register of the printer port. The Strobe, Autofeed, and Init control outputs of the printer adapter provide the clock signals of the three 74LS273 [5] latches on the simulator board. The three control bytes are read by the simulator via three 74LS244 octal buffers. The simulation computer enables these octal buffers to read their outputs via 74LS157 quad data selectors/multiplexers, using Strobe, Autofeed, and Init.

Bottle Filler Simulator Software

The bottle filling environment is simulated by using the software simulation running in an IBM PC computer. First, the simulation initializes the display to show a diagram of the bottle filling system (as shown in Figure 2.1), and then reads the control bits from the control system through the simulation circuit board.

By reading the control bit pattern, the tank levels will increase if the fill valves on the tanks are open, and tank levels will decrease if the dump valves are open. An error message will be printed if a dump valve and a fill valve in a tank are open at the same time or if a tank reaches its maximum capacity.

The simulation updates the display every dt seconds where dt is nominally one. It should be noted that the dt can be varied, it can be decreased so that simulation runs

faster than real time or increased to run slower than real time.

In every dt seconds the simulation:

(1) Updates the levels of each tank and bottle filler.

(2) Displays the levels, valves, and mix motor condition.

(3) Outputs the requested level to the printer port.

CHAPTER 4

CONTROL SYSTEM

The control system is divided into three tasks. First, it displays a menu of choices. From the menu, the operator composes one or more recipes. The recipes are stored in a linked list. The operator may also fill bulk tanks or begin a batch to be bottled. The second task sends control information to the simulation circuit. The final task reads back the tank levels from the simulation system, and displays the tank levels. The block diagram of the control system is shown in Figure 4.1.

Digital Hardware

In this system, an IBM personal computer controls the operation of the bottle filling facility. The computer is equipped with a parallel port which interfaces with the simulation circuits board enabling it to measure the state of the system and impose control over its operation. The control algorithms require each of the 13 tank levels to be sampled on a regular basis.

In addition to interfacing with the bottle filling facility, the computer also interfaces with its operators

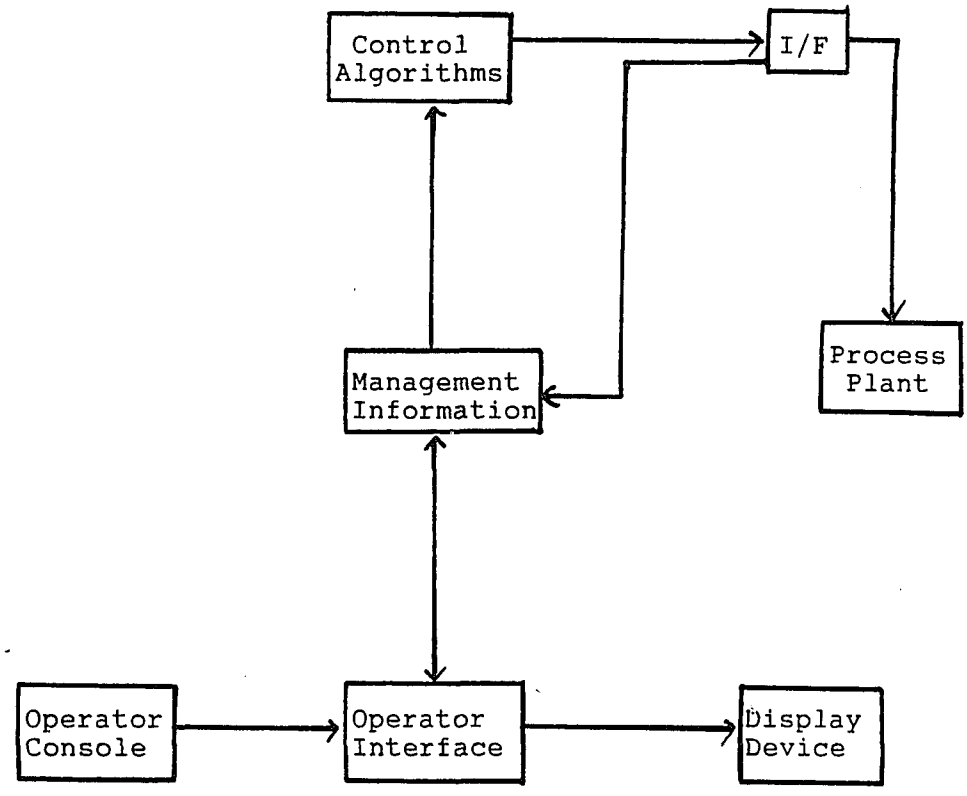


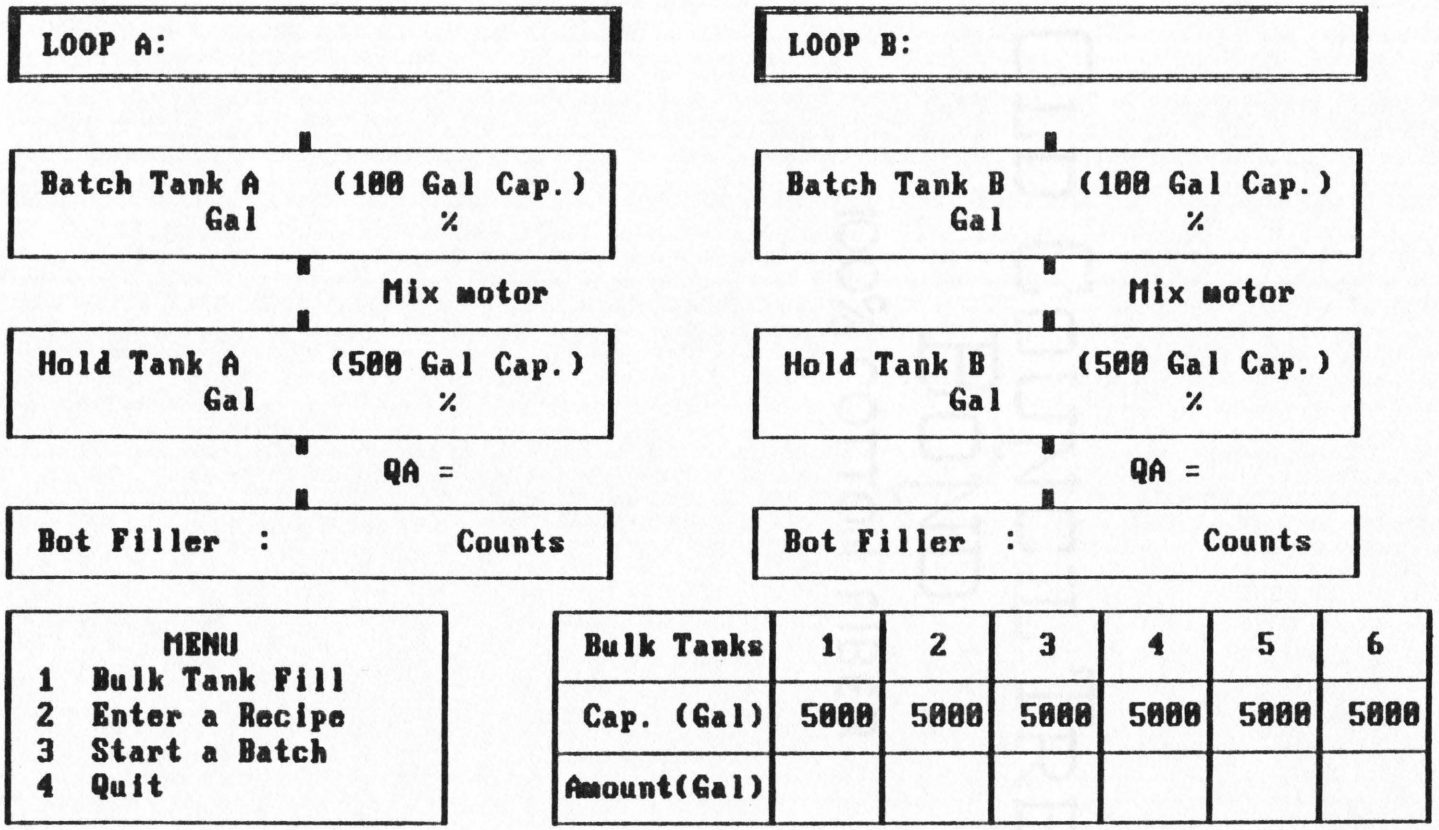
Figure 4.1 Function Block Diagram for Control System

in order to: (1) initiate bulk tank filling; (2) enter recipes; (3) start a batch to run; (4) enter a quality assurance code, and (5) perform some emergency operation. Additionally, the tank levels, valve conditions, and process activity are displayed on a monitor. Figure 4.2 shows the display screen of the control system.

Software

The software for the system is naturally categorized by its distinct tasks. In this system several tasks have been identified: displaying process activity, allowing the operator to enter control information, sending control information, and reading tank levels from the simulation. These tasks are represented in the software by processes which ideally should be executed in parallel, interacting where necessary to synchronize certain events and to transfer information.

It should be noted that, in order to meet the response time requirements of the system, some tasks, such as some emergency controls or reading tank levels from the simulation system, should be assigned higher priority than less urgent task. Also, the parallel construct of the system implies that a multi-tasking operating system may be appropriate. In this project, because of the time constraint and the personal limitations, a sequential



Bulk Tanks	1	2	3	4	5	6
Cap. (Gal)	5000	5000	5000	5000	5000	5000
Amount(Gal)						

Figure 4.2 Display Screen of Control System

programming technique is used.

Process control software is large and complex, such as this one, may be dealt with by breaking the task into modules. The software is broken into small self-contained units, with each unit being separately compiled and tested. Modularization is important for many reasons: it aids program design, and it simplifies modification and maintenance; furthermore, it allows the software to be developed by teams of programmers since each member of the team can work on a separate module.

The software developed for this system has been implemented in the C language, except one module which is written in assembly language for fast screen updates. The function of each module is briefly described in Table 4.1.

The C program consists of eight modules, each independently compiled by the Microsoft C compiler version 5.0 [6], using the large model. The assembly language program is compiled by Microsoft Macro Assembler version 5.0 [7].

Write-String Assembly Routine

Program arguments in the module "wrtstr.asm" are listed in Table 4.2. This assembly routine writes the string, with attributes, to the desired screen position. It writes the string directly to the video RAM for maximum

Table 4.1 Function of Each Module in Control System

Module	Description
Wrtstr.asm	Writes string directly to video RAM.
Control.c	Main function does the control.
Menu.c	Function handles the user interface.
Recipe.c	Function saves the user input recipes.
Delay.c	Function provides specified duration delay.
Sound.c	Function sounds for a specific duration.
Savescrn.c	Function saves and restores screen image.

Table 4.2 Argument Descriptions for Module Wrtstr.asm

Argument	Description
X	Column position to print the string.
Y	Row position to print the string.
Far *	Starting address of the string.
Length	Length of the string to print.
Attribute	Attribute of the string.

speed. The C programs use this routine for all screen updates.

Machine Dependent Modules

When a key is pressed, the 09H interrupt service routine reads the keyboard input port, which returns a scan code. Each key on the keyboard has a scan code. Function Scan_Code detects and stores the operator input data and uses it in other modules.

The delay function provides a delay of approximately the specified duration (resolution is about 0.055 second). It used the computer's timer to generate delays that are consistent with the entire family of IBM PCs and compatible machines.

Function "sound" produces a constant tone for a specified duration. The sound system is built upon channel two of the timer subsystem and a couple of I/O ports in the 8255 programmable peripheral interface (PPI) device. The timer/counter produces square-wave signals that are amplified and filtered, then passed to the speaker. The AND gate ahead of the driver is controlled by bit 1 of the PPI port 61 hex. In addition, bit 0 of the same port controls the gate lead of timer channel two. Thus, the PPI can be used to turn sound on/off.

Function save_screen moves the video RAM to a

buffer and refreshes it as needed. It first calls function 0FH of interrupt 10H service routine to determine the mode of video state so the address of video RAM can be known. It then moves the whole video page to a temporary buffer for re-display later.

User Interface

The program Menu.c is used mainly for the user interface. Figure 4.3 shows the flow chart of the menu system. If a valid choice has been entered, the switch statement branches to the appropriate routine.

To begin filling a bulk tank, the operator is asked to enter a bulk tank number and the amount. If either the bulk tank number or the amount entered are out of range, the operator will be asked to correct the entries. The valid bulk tank numbers are between one and six, and the valid ingredient amount is between one and 5000. The operator can stop the process at any time by pressing the escape key.

When entering an ingredient for the recipe, the operator is asked to enter the name of the recipe. If the name is already in the menu, the amount of each chemical ingredient is displayed and the operator can make any necessary changes. If the name is not in the menu, a new recipe can be entered. The recipe is stored in a linked

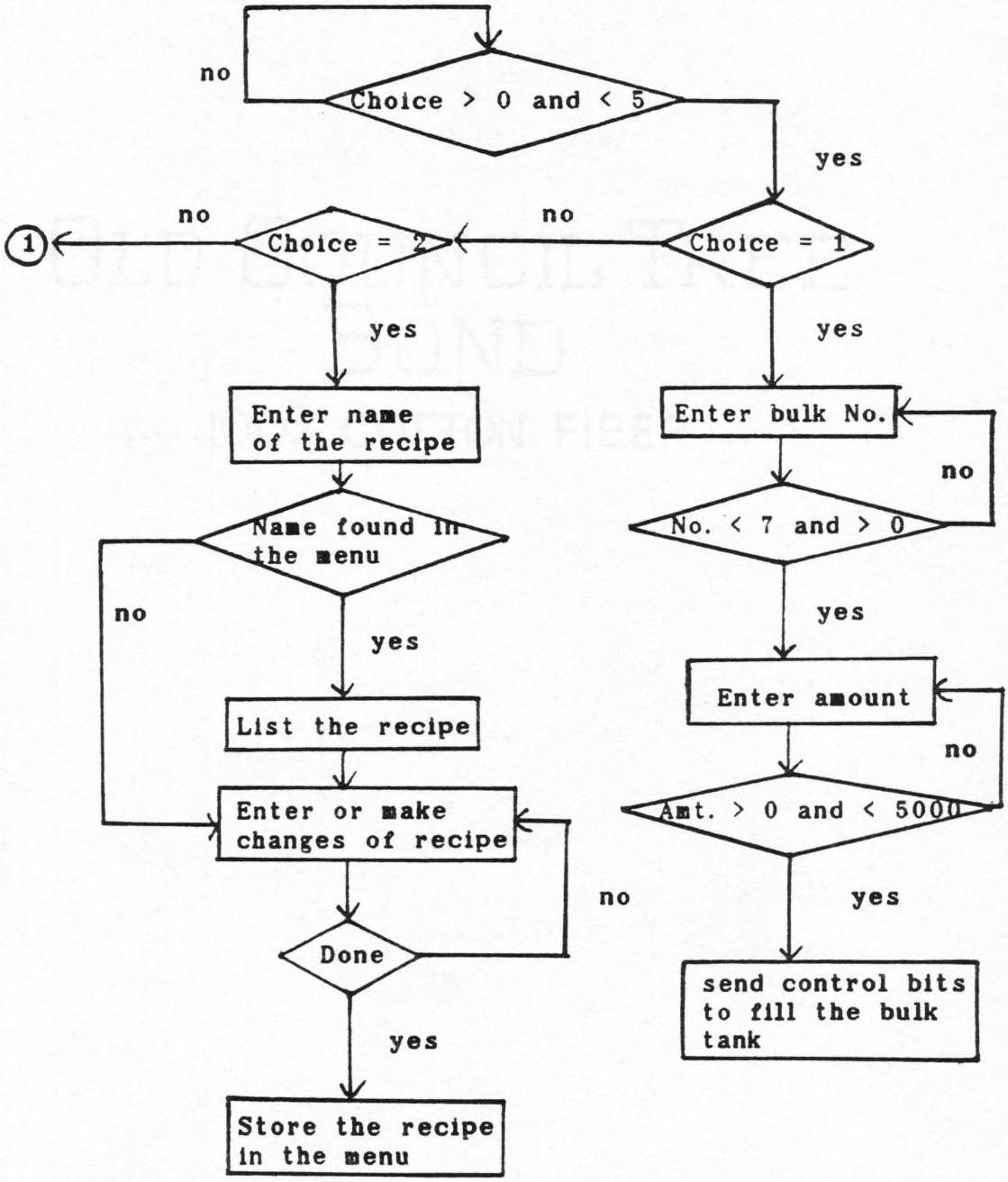


Figure 4.3 Flow Chart for Module Menu

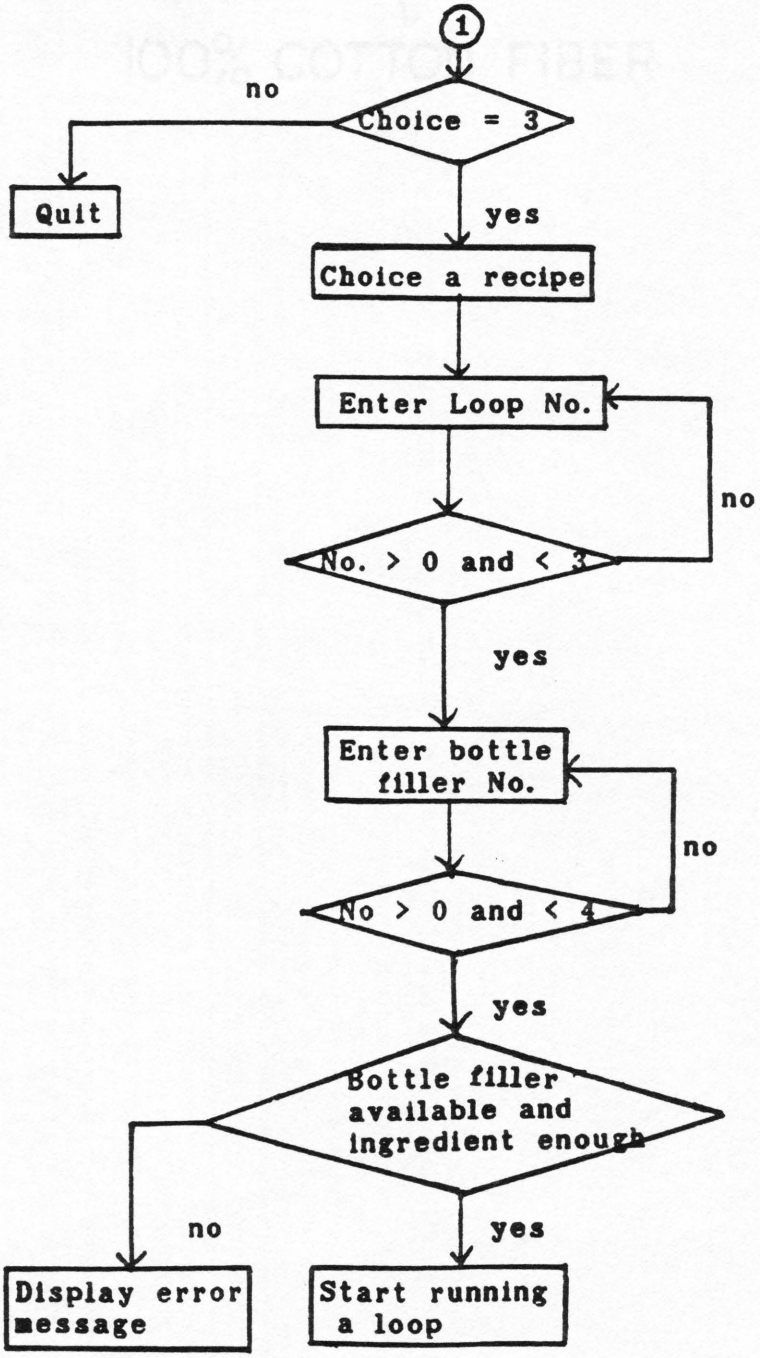


Figure 4.3 Flow Chart for Module Menu--Continued

list with the name as the key. The total quantity of ingredients cannot exceed 500 gallons. The command can be aborted at any time by pressing the escape key (the recipe is not saved).

When starting a loop, the operator is asked to select a recipe from the recipe list, and to choose the loop and bottle filler. If the chemical ingredients are not available in the bulk tanks or the loop and/or bottle filler are not available, the request will be denied.

When a loop is running and the mix motor is turned on for the specified period, the inspector is asked to enter the quality assurance code, which indicates the result from the chemical analysis. In this simulation, the bottle filler will start bottling after the quality assurance code is entered even if the batch fails the analysis. Presumably, the bottles are disaided.

Control System

Two data structures TankLvl, and VlvCtrl are used in the control system (as shows in Table 4.3 and Table 4.4). The first records the tank levels and the other controls and reports on the valve conditions.

Ctrl_Bits is a global variable which is sent to the simulation system. It is modified when filling is started or stopped.

Table 4.3 Data Structure for Tank Levels Control

Variable	Description
RLvlMask	Nibble indicates next tank level should return from simulation system.
Cur_Lvl	Current level in the tank.
Max_Lvl	Maximum tank level allowed.
Req_Lvl	The requested amount to fill a tank.
Delta	The safety amount to stop a tank.
Lvl_X	Column position to print the tank level.
Lvl_Y	Row position to print the tank level.
Pcnt_X	Column position to print the % of tank level.
Pcnt_Y	Row position to print the % of tank level.
Vlv_Cond	Indicates tank condition busy or idle.

Table 4.4 Data Structure for Tank Valves Control

Variable	Description
Fill_Mask	Bit Masks for control a valve.
Fill_Bits	Bits send to control open or close a valves.
Offset_1	Offset to corresponding fill tank.
Offset_2	Offset to corresponding dump tank.
Vlv_Cond	Indicates valve is busy or not.

Routine Fill_Tank is called by other routines whenever a tank needs to be filled. When filling a tank, the variable Req_Lvl in the structure TankLvl is set equal to the requested amount. The Ctrl_Bits are updated and sent to the simulation system by calling the routine Send_Bits. The corresponding valves are set to "Busy" (Vlv_Cond) if a tank is in the process of filling.

In addition to sending the Ctrl_Bits to the simulation system, routine Send_bits has two more functions. First, it reads the tank level returned from the simulation system and updates the variable Cur_Lvl in the structure TankLvl to the amount that it has just read. Second, it changes the Ctrl_Bits, in order to select the tank level returned by the simulation system.

Routine Stop_Tank compares the variable Cur_Lvl with Req_Lvl and Max_Lvl in the structure TankLvl if the result of Cur_Lvl minus delta is larger than Req_Lvl or Max_Lvl. The Ctrl_Bits are updated and sent to the simulation system to halt the tank filling. Also, the variable Vlv_Cond in structure VlvCtrl is set to "Idle."

The method used to fill a tank is called "dribbling." A tank stops filling when its current ingredient amount minus delta is larger than Req_Lvl or Max_Lvl. Filling is stopped before the requested amount is reached in order to let the ingredient in the pipe flow to

the tank in the actual system, and also to let the control system catch up in the simulation system. This was one of the more difficult system requirements to implement correctly.

Routine Vlv_Rpt reads the variable Vlv_Cond, and Cur_Lvl and updates the tank levels and valve conditions on the display screen. It will open or close a valve depending on the Vlv_Cond reading.

When initiating a loop run, the routine Init sets the variables "loop" and "bottle filler" to "Busy". When the loop is finished, those variables are set at "Idle".

The routine Run_Loop is the heart of the control system. The finite state machine auto-transition technique is used in the control algorithm. The state transition table for the module Run_Loop is shown in Table 4.5 and Table 4.6.

There are five states in the Run_Loop. If the sum of the chemical ingredients needed to fill the hold tank is larger than zero, and the batch tank condition is idle, and also the batch tank chemical ingredient is equal to zero the state will be one which is dumping the bulk tank chemical ingredient to the batch tank.

If the batch tank level is larger than zero and the batch tank condition is idle, the state is two. In this state the batch tank filled valve is closed the chemical

Table 4.5 Function of Each State in Module RunLoop

State	Description
0	Idle
1	Bulk tank ingredient dumps to batch tank.
2	Batch tank ingredient dumps to hold tank.
3	Turn timer on.
4	Enter quality assurance code.
5	Hold tank ingredient dumps to bottle filler.

Table 4.6 State Transition Table for Module RunLoop

Current State	Condition	Next State
0	Idle.	0
	Start a batch.	1
1	More ingredient to fill batch tank.	1
	Batch tank reaches its request amount.	2
2	Batch tank empty, more bulk tank ingredient to fill.	1
	Batch tank is not empty.	2
	Batch tank is empty, no bulk tank ingredient to fill.	3
3	Timer time is not expired.	3
	Timer time is expired.	4
4	Quality assurance code is not entered.	4
	Quality assurance code is entered.	5
5	Hold tank is not empty.	5
	Hold tank is empty.	0

ingredients are already collect in the batch tank and waiting for dumping to hold tank, so the batch tank dump valve will be opened and the chemical ingredient will dump to the hold tank.

During state two, if the hold tank is half full, the mix motor is turned on.

After all the chemical ingredients are collected in the hold tank it goes to state three. In this state a timer will turn on to time the mix motor turned on time.

In state four, program checks if the mix motor turn-on time has been expired, if it has expired, the timer turns off, and operator will be asked for entering a quality assurance code. After the quality assurance code has been entered, it will go to state five which starts the bottle filling. State five is the final state. When the hold tank is emptied the bottle filling process is finished and the program will call the routine Finish to do some clean up such as returning the bottle filler to the available resources list.

CHAPTER 5

CONCLUSION

Problems Discovered

An eight-bit register in the printer adapter was used to transmit the control bits to the simulation system and return the tank levels back to the control system. The maximum value of an eight-bit register can represent is 255; however the maximum tank level capacity in the bulk tank is 5000 gallons, and therefore, after some calculation has been performed, the tank levels in the simulation and the tank levels read by the control system are not the same.

When a tank is filling, the real-time tank levels and the levels read back from the simulation are different. Because of this,it is hard to control the filling of a tank to an exact value. In this design a "dribbling" method has been employed for control of the filling process.

Another problem is presented while the operator is entering data to the console. Because of the sequential nature of the design, all other processes in the controlling system are postponed. At the same time the

real-time controlled system will continue to run without proper control.

A multi-tasking programming or a foreground-background programming technique are suggested to improve the response of the control system.

Future Work

In the design of bottle filling facility, several tasks have been identified: Running Loops, sending control to simulation system, operator interface, and reading tank levels from simulation system.

Real-time Multi-tasking Operating Systems

The control of a real-time system such as a bottle filling facility, is considerably eased by the availability of real-time multi-tasking operating systems [1].

The essential features of real-time multi-tasking operating systems are:

(1) Ability to monitor and change the states of the tasks.

(2) To regulate the exchange of information between tasks.

(3) Interrupt handling.

The standard way of ensuring regulated exchange of data is to insist that information transfer between tasks

is through the operating system. The operating system can then use a monitor to ensure correct access. Real-time multi-tasking operating systems vary in the way in which inter-task communication is organized, some protected mechanism of supervisor calls, others insist that all communication between tasks is through the operating system.

Interrupt handling is crucial to the operating of real-time systems and is one of the most difficult areas of program construction and testing. The aim of handling interrupts within the operating system is to hide the details of the actual implementation from the application programmer. It is frequently adequate if the system is interrupted at preset time intervals. It is useful to allow a task is to be executed during an interrupt. This enables an alarm tasks to be programmed at the application level rather than at the operating system.

Foreground-Background Systems

In this bottle filling simulation, a division can be made between tasks which have to be synchronized with the external environment - data logging (reading tank levels from simulation system), direct digital control of the simulation, or supervisory control (emergency stop a tank filling), and tasks which perform a service - operator

communication, long term performance monitoring - which do not have to tightly synchronized. This division leads to the idea of dividing the tasks into foreground tasks i.e. tasks of high priority which are synchronized to the external environment and background tasks of lower priority.

It is possible to adapt standard sequential operating system to run a simple form of foreground-background system [1] by making the foreground task into an interrupt routine.

For many simple systems the foreground-background approach using interrupt routines is adequate: the weakness is in the methods of sharing data between the tasks. In the absence of operating system support the only data sharing mechanism available is through the use of shared memory i.e. the use of shared or common variables. This is not always apparent since data is usually passed between the two tasks by means of parameters in procedure calls, however, unless interrupts are inhibited the effect is the equivalent of having the data held in a common area.

The transfer of data between the foreground and the background task is a critical section of the program and should be an indivisible action which should be carried out by the operating system. In the absence of operating system support special routines can be provided as in Figure 5.1.

```
Procedure Modify (parameter list ...);
  Begin
        Inhibit_Interrupts;
        Transfer_Parameters;
        Enable_Interrupts;
  End;
```

Figure 5.1 Routine Handles Transfer Data Between the Foreground and Background Tasks

By using a routine of the type shown, the transfer of parameters to the control routine cannot be interrupted; however, a disturbance to the controller can still occur since the control action may be delayed and there is no way for the control task to indicate that it is a suitable time for the transfer of the information.

Summary

This project was started with the goal of designing a real-time control system and a simulation system for a chemical mix facility. Several important considerations have been identified while conducting this study.

First, the finite state auto-transition method used in the control algorithm easier the design. Second, simulation allowed us to vary system parameters easily, such as the tank filling speed, in order to test the robustness and reliability of the control. Third, by simulating the controlled system, the software can be developed and tested without using the actual plant. Fourth, problems can be discovered earlier. For example, the time delay problem in reading the tank levels was found and solved before the system was put on-line. Fifth, the simulation can help in selecting the control equipment. For example, the eight-bit analog-to-digital converter could be changed to a twelve-bit analog-to-digital converter in

order to improve the accuracy of the tank level reading.

REFERENCES

1. Peterson, J. L. and A. Silberschatz, "Operating System Concepts", Addison-Wesley, California, 2nd ed., 1984.

2. Turner, R. C., "Real-Time Programming with Microcomputers", Lexington Books, Massachusetts, 1978.

3. Allworth, S. T., "Introduction to Real-time Software Design", The Macmillan Press LTD, London, 1981.

4. Technical Reference, Personal Computer AT, IBM Corp., 1984.

5. TTL Data Book, Signetics, California, 1985.

6. Microsoft C Compiler, Microsoft Corporation, Washington, 1987.

7. Microsoft Macro Assembler, Microsoft Corporation, Washington, 1987.